aws

Developer Guide

AWS Lake Formation



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Lake Formation: Developer Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is AWS Lake Formation? 1
Lake Formation features 2
Data ingestion and management 2
Security management 3
Bring your data into the Data Catalog 4
How it works 6
Lake Formation permissions management workflow 6
Metadata permissions 8
Storage access management 11
Cross-account data sharing in Lake Formation 12
Lake Formation components 13
Lake Formation console 13
Lake Formation API and Command Line Interface 14
Other AWS services 14
Lake Formation terminology 14
Data lake 14
Data access 14
Hybrid access mode 15
Blueprint 15
Workflow 15
Data Catalog 15
Underlying data 16
Principal 16
Data lake administrator 16
AWS service integrations with Lake Formation 17
Additional Lake Formation resources 18
Blogs 19
Tech talks and webinars 19
Modern day architecture 19
Data mesh resources 19
Best practices guides 19
Getting started with Lake Formation 20
Getting started 21
Complete initial AWS configuration tasks 21

Sign up for an AWS account ... 21
Set up AWS Lake Formation ... 21
Set up Lake Formation resources using CloudFormation template ... 22
Create a data lake administrator ... 23
Change the default permission model or use hybrid access mode ... 28
Assign permissions to Lake Formation users ... 29
Configure an Amazon S3 location for your data lake ... 30
(Optional) External data filtering settings ... 31
(Optional) Grant access to the Data Catalog encryption key ... 32
(Optional) Create an IAM role for workflows ... 32
Upgrading AWS Glue data permissions to the Lake Formation model ... 34
About default permissions ... 34
List existing permissions ... 36
Set up Lake Formation permissions ... 38
Give users IAM permissions ... 39
Switch to the Lake Formation permissions model ... 39
Step 5: Secure new Data Catalog resources ... 42
Step 6: Give users a new IAM policy ... 43
Step 7: Clean up existing IAM policies ... 44
Setting up Amazon VPC endpoints (AWS PrivateLink) ... 45
Considerations for Lake Formation VPC endpoints ... 45
Creating an interface VPC endpoint for Lake Formation ... 46
Creating a VPC endpoint policy for Lake Formation ... 46
Tutorials ... 48
Creating a data lake from an AWS CloudTrail source ... 49
Intended audience ... 50
Prerequisites ... 51
Step 1: Create a data analyst user ... 51
Step 2: Add permissions to read AWS CloudTrail logs to the workflow role ... 52
Step 3: Create an Amazon S3 bucket for the data lake ... 53
Step 4: Register an Amazon S3 path ... 53
Step 5: Grant data location permissions ... 54
Step 6: Create a database in the Data Catalog ... 54
Step 7: Grant data permissions ... 54
Step 8: Use a blueprint to create a workflow ... 55
Step 9: Run the workflow ... 56

Step 10: Grant SELECT on the tables ... 57
Step 11: Query the data lake Using Amazon Athena ... 58
Creating a data lake from a JDBC source ... 59
Intended audience ... 59
Prerequisites ... 60
Step 1: Create a data analyst user ... 60
Step 2: Create a connection in AWS Glue ... 62
Step 3: Create an Amazon S3 bucket for the data lake ... 62
Step 4: Register an Amazon S3 path ... 63
Step 5: Grant data location permissions ... 63
Step 6: Create a database in the Data Catalog ... 63
Step 7: Grant data permissions ... 64
Step 8: Use a blueprint to create a workflow ... 64
Step 9: Run the workflow ... 66
Step 10: Grant SELECT on the tables ... 67
Step 11: Query the data lake using Amazon Athena ... 68
Step 12: Query the data in the data lake using Amazon Redshift Spectrum ... 68
Step 13: Grant or revoke Lake Formation permissions using Amazon Redshift Spectrum ... 72
Setting up permissions for open table formats in Lake Formation ... 72
Intended audience ... 73
Prerequisites ... 74
Step 1: Provision your resources ... 75
Step 2: Set up permissions for an Iceberg table ... 77
Step 3: Set up permissions for a Hudi table ... 83
Step 4: Set up permissions for a Delta Lake table ... 85
Step 5: Clean up AWS resources ... 88
Managing a data lake using tag-based access control ... 88
Intended audience ... 89
Prerequisites ... 90
Step 1: Provision your resources ... 91
Step 2: Register your data location, create an LF-Tag ontology, and grant permissions ... 92
Step 3: Create Lake Formation databases ... 95
Step 4: Grant table permissions ... 105
Step 5: Run a query in Amazon Athena to verify the permissions ... 107
Step 6: Clean up AWS resources ... 108
Securing data lakes with row-level access control ... 108

Intended audience ... 109
Prerequisites ... 110
Step 1: Provision your resources ... 110
Step 2: Query without data filters ... 112
Step 3: Set up data filters and grant permissions ... 113
Step 4: Query with data filters ... 115
Step 5: Clean up AWS resources ... 117
Securely share your data using Lake Formation ... 117
Intended audience ... 118
Configure Lake Formation settings ... 119
Step 1: Provision your resources using AWS CloudFormation templates ... 121
Step 2: Lake Formation cross-account sharing prerequisites ... 123
Step 3: Implement cross-account sharing using the tag-based access control method ... 126
Step 4: Implement the named resource method ... 132
Step 5: Clean up AWS resources ... 135
Sharing Data Catalog resources with external AWS accounts using fine-grained access control ... 136
Intended audience ... 137
Prerequisites ... 138
Step 1: Provide fine-grained access to another account ... 139
Step 2: Provide fine-grained access to a user in the same account ... 140
Onboarding to Lake Formation permissions ... 142
Overview of Lake Formation permissions ... 143
Methods for fine-grained access control ... 145
Metadata access control ... 148
Underlying data access control ... 152
Lake Formation personas and IAM permissions reference ... 158
AWS Lake Formation personas ... 158
AWS managed policies for Lake Formation ... 160
Personas suggested permissions ... 168
Changing the default settings for your data lake ... 179
Implicit Lake Formation permissions ... 182
Lake Formation permissions reference ... 183
Lake Formation permissions per resource type ... 184
Lake Formation grant and revoke AWS CLI commands ... 187
Lake Formation permissions ... 191

Integrating IAM Identity Center ... 206
Prerequisites for IAM Identity Center integration with Lake Formation ... 207
Connecting Lake Formation with IAM Identity Center ... 211
Updating IAM Identity Center integration ... 215
Deleting a Lake Formation connection with IAM Identity Center ... 217
Granting permissions to users and groups ... 218
Including IAM Identity Center user context in CloudTrail logs ... 221
Adding an Amazon S3 location to your data lake ... 223
Requirements for roles used to register locations ... 224
Registering an Amazon S3 location ... 231
Registering an encrypted Amazon S3 location ... 235
Registering an Amazon S3 location in another AWS account ... 240
Registering an encrypted Amazon S3 location across AWS accounts ... 242
Deregistering an Amazon S3 location ... 247
Accessing Amazon S3 locations ... 247
Hybrid access mode ... 252
Common hybrid access mode use cases ... 254
How hybrid access mode works ... 255
Setting up hybrid access mode - common scenarios ... 257
Removing principals and resources from hybrid access mode ... 276
Viewing principals and resources in hybrid access mode ... 277
Additional resources ... 278
Creating objects in the AWS Glue Data Catalog ... 279
Creating a catalog ... 280
Creating a database ... 280
Creating tables ... 281
Building Data Catalog views ... 289
Importing data using workflows ... 336
Blueprints and workflows ... 336
Creating a workflow ... 338
Running a workflow ... 341
Bringing your data into the Data Catalog ... 343
Catalog federation ... 345
Features of Catalog Federation ... 345
Limitations ... 345
Federate to Snowflake ... 346

Federate to Databricks ... 354
Bringing Amazon Redshift data into the Data Catalog ... 362
Key benefits ... 364
Roles and responsibilities ... 364
Prerequisites ... 365
Creating Amazon Redshift federated catalogs ... 371
Viewing catalog objects ... 380
Updating a federated catalog ... 382
Accessing a shared federated catalog ... 384
Deleting a federated catalog ... 388
Querying federated catalogs ... 389
Additional resources ... 390
Federating into external data sources ... 390
Workflow ... 391
Prerequisites ... 392
Creating a federated catalog ... 396
Viewing catalog objects ... 402
Deleting a federated catalog ... 404
Querying federated catalogs ... 405
Additional resources ... 406
Amazon S3 Tables integration with Data Catalog and Lake Formation ... 406
How Data Catalog and Lake Formation integration works ... 407
Prerequisites ... 409
Enabling Amazon S3 Tables integration ... 412
Creating databases and tables ... 417
Registering an Amazon S3 table bucket in another AWS account ... 420
Granting permissions ... 423
Changing access controls for S3 Tables integration ... 426
Enable Lake Formation with S3 Tables integration ... 426
Change from Lake Formation to IAM ... 428
Creating an Amazon Redshift managed catalog ... 432
Managing permissions for data in an Amazon Redshift datashare ... 436
Prerequisites ... 438
Setting up permissions for Amazon Redshift datashares ... 438
Querying federated databases ... 442
Managing permissions on datasets that use external metastores ... 443

Workflow ... 446
Prerequisites ... 446
Connecting the Data Catalog to an external Hive metastore ... 449
Additional resources ... 452
Managing Lake Formation permissions ... 453
Granting data location permissions ... 453
Granting data location permissions (same account) ... 454
Granting data location permissions (external account) ... 456
Granting permissions on a data location shared with your account ... 460
Granting data permissions ... 461
IAM permissions required to grant Lake Formation permissions ... 462
Using the named resources method ... 465
Tag-based access control ... 490
Granting data lake permissions using the LF-TBAC method ... 551
Attribute-based access control ... 558
Permissions example scenario ... 562
Data filtering and cell-level security ... 564
Data filters ... 566
PartiQL support in row filter expressions ... 570
Permissions required for querying tables with cell-level filtering ... 572
Managing data filters ... 573
Row-level security policies ... 588
RLS policy details ... 592
Managing RLS policies ... 598
Attaching and detaching RLS policies ... 603
Cross-account RLS policies ... 611
Filter expression syntax ... 614
Best practices for RLS policies ... 622
Viewing Database and Table Permissions ... 630
Revoking permissions using the console ... 633
Cross-account data sharing ... 634
Prerequisites ... 637
Updating cross-account data sharing version settings ... 641
Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts ... 647
Granting permissions on a database or table shared with your account ... 653

Granting resource link permissions ... 655
Accessing the underlying data of a shared table ... 658
Cross-account CloudTrail logging ... 660
Managing cross-account permissions using both AWS Glue and Lake Formation ... 664
Viewing all cross-account grants using the GetResourceShares API operation ... 667
Accessing and viewing shared Data Catalog tables and databases ... 669
Accepting an AWS RAM resource share invitation ... 670
Viewing shared Data Catalog tables and databases ... 672
Creating resource links ... 674
How resource links work ... 674
Creating a resource link to a shared table ... 677
Creating a resource link to a shared database ... 680
Resource link handling in AWS Glue APIs ... 684
Accessing tables across Regions ... 688
Workflows ... 689
Setting up cross-Region table access ... 693
Security ... 696
Data Protection ... 696
Encryption at Rest ... 697
Infrastructure Security ... 698
Cross-service confused deputy prevention ... 698
Security event logging in AWS Lake Formation ... 700
Integrating with Lake Formation ... 701
Using Lake Formation application integration ... 701
How Lake Formation application integration works ... 702
Roles and responsibilities in Lake Formation application integration ... 704
Lake Formation workflow for application integration API operations ... 705
Registering a third-party query engine ... 707
Enabling permissions for a third-party query engine to call application integration API operations ... 709
Application integration for full table access ... 713
Working with other AWS services ... 715
Amazon Athena ... 718
Support for transactional table formats ... 720
Additional resources ... 722
Amazon Redshift Spectrum ... 723

Support for transactional table types ... 723
Additional resources ... 725
AWS Glue ... 725
Support for transactional table types ... 726
Additional resources ... 727
Amazon EMR ... 728
Support for transactional table formats ... 728
Additional resources ... 729
Quick ... 729
Additional resources ... 730
AWS CloudTrail Lake ... 730
Logging AWS Lake Formation API calls using AWS CloudTrail ... 731
Lake Formation information in CloudTrail ... 731
Understanding Lake Formation events ... 732
Tracing data access in CloudTrail with Lake Formation ... 734
Lake Formation best practices, considerations, and limitations ... 737
Cross-account data sharing best practices and considerations ... 737
Service-linked role limitations ... 739
Cross-Region data access limitations ... 740
Data Catalog views considerations and limitations ... 741
Data filtering limitations ... 742
Notes and restrictions for column-level filtering ... 742
Cell-level filtering limitations ... 744
Hybrid access mode considerations and limitations ... 745
Limitations for bringing Amazon Redshift data warehouse data into the AWS Glue Data Catalog ... 747
S3 Tables catalog integration limitations ... 748
Hive metadata store data sharing considerations and limitations ... 749
Amazon Redshift data sharing limitations ... 751
IAM Identity Center integration limitations ... 752
Lake Formation tag-based access control best practices and considerations ... 753
Attribute-based access control considerations, limitations, and supported regions ... 756
Row-level security policy limitations ... 756
Troubleshooting Lake Formation ... 759
General troubleshooting ... 759
Error: Insufficient Lake Formation permissions on <Amazon S3 location> ... 759

Error: "Insufficient encryption key permissions for Glue API" 760
My Amazon Athena or Amazon Redshift query that uses manifests is failing 760
Error: "Insufficient Lake Formation permission(s): Required create tag on catalog" 760
Error when deleting invalid data lake administrators 760
Error: "Caller is temporarily not authorized to perform this operation on resource..." 760
Troubleshooting cross-account access 760
I granted a cross-account Lake Formation permission but the recipient can't see the resource 761
Principals in the recipient account can see the Data Catalog resource but can't access the underlying data 762
Error: "Association failed because the caller was not authorized" when accepting a AWS RAM resource share invitation 762
Error: "Not authorized to grant permissions for the resource" 763
Error: "Access denied to retrieve AWS Organization information" 763
Error: "Organization <organization-ID> not found" 763
Error: "Insufficient Lake Formation permissions: Illegal combination" 763
ConcurrentModificationException on grant/revoke requests to external accounts 763
Error when using Amazon EMR to access data shared via cross-account 763
Troubleshooting blueprints and workflows 765
My blueprint failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>" 765
My workflow failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>" 765
A crawler in my workflow failed with "Resource does not exist or requester is not authorized to access requested permissions" 766
A crawler in my workflow failed with "An error occurred (AccessDeniedException) when calling the CreateTable operation..." 766
Known issues for AWS Lake Formation 766
Limitation on filtering of table metadata 766
Issue with renaming an excluded column 767
Issue with deleting columns in CSV tables 768
Table partitions must be added under a common path 768
Issue with creating a database during workflow creation 768
Issue with deleting and then re-creating a user 768
Data Catalog API operations do not update the value for the `IsRegisteredWithLakeFormation` parameter 769

Lake Formation operations do not support AWS Glue Schema Registry 769
Updated error message 769
Lake Formation API 770
Permissions 771
— operations — 771
— data types — 771
Data Lake Settings 772
— operations — 772
— data types — 772
IAM Identity Center integration 772
— operations — 772
— data types — 772
Hybrid access mode 773
— operations — 773
— data types — 771
Credential vending 773
— operations — 773
— data types — 774
Tagging 774
— operations — 774
— data types — 775
Data filter APIs 775
— operations — 775
— data types — 775
Common data types 775
ErrorDetail 776
String patterns 776
Supported Regions 777
General availability 777
AWS GovCloud (US) 777
Transactions and storage optimization 777
Document History 780
AWS Glossary 794

What is AWS Lake Formation?

Welcome to the AWS Lake Formation Developer Guide.

AWS Lake Formation helps you centrally govern, secure, and globally share data for analytics and machine learning. With Lake Formation, you can manage fine-grained access control for your data lake data on Amazon Simple Storage Service (Amazon S3) and its metadata in AWS Glue Data Catalog.

Lake Formation provides its own permissions model that augments the IAM permissions model. Lake Formation permissions model enables fine-grained access to data stored in data lakes as well as external data sources such as Amazon Redshift data warehouses, Amazon DynamoDB databases, and third-party data sources through a simple grant or revoke mechanism, much like a relational database management system (RDBMS). Lake Formation permissions are enforced using granular controls at the column, row, and cell-levels across AWS analytics and machine learning services, including Amazon Athena, Amazon Quick, Amazon Redshift Spectrum, Amazon EMR, and AWS Glue.

With Lake Formation hybrid access mode for AWS Glue Data Catalog (Data Catalog), you can secure and access the cataloged data using both Lake Formation permissions and IAM permissions policies for Amazon S3 and AWS Glue actions. With hybrid access mode, data administrators can onboard Lake Formation permissions selectively and incrementally, focusing on one data lake use case at a time.

Lake Formation also allows you to share data internally and externally across multiple AWS accounts, AWS organizations or directly with IAM principals in another account providing fine-grained access to the Data Catalog metadata and underlying data.

Topics

- Lake Formation features
- AWS Lake Formation: How it works
- Lake Formation components
- Lake Formation terminology
- AWS service integrations with Lake Formation
- Additional Lake Formation resources

- Getting started with Lake Formation

Lake Formation features

Lake Formation helps you break down data silos and combine different types of structured and unstructured data into a centralized repository. First, identify existing data stores in Amazon S3 or relational and NoSQL databases, and move the data into your data lake. Then crawl, catalog, and prepare the data for analytics. Next, provide your users with secure self-service access to the data through their choice of analytics services.

You can use Lake Formation console to create multi-level federated catalogs in the Data Catalog, and unify data across Amazon S3 data lakes and Amazon Redshift data warehouses. You can also integrate data from your operational databases such as Amazon DynamoDB, and third-party data sources such as Google BigQuery, MySQL, among others. The Data Catalog provides a centralized metadata repository that makes managing and discovering data across disparate systems easier.

For more information, see Bringing your data into the AWS Glue Data Catalog.

Topics

- Data ingestion and management
- Security management
- Bring your data into the Data Catalog

Data ingestion and management

Import data from databases already in AWS

Once you specify where your existing databases are and provide your access credentials, Lake Formation reads the data and its metadata (schema) to understand the contents of the data source. It then imports the data to your new data lake and records the metadata in a central catalog. With Lake Formation, you can import data from MySQL, PostgreSQL, SQL Server, MariaDB, and Oracle databases running in Amazon RDS or hosted in Amazon EC2. Both bulk and incremental data loading are supported.

Import data from other external sources

You can use Lake Formation to move data from on-premises databases by connecting with Java Database Connectivity (JDBC). Identify your target sources and provide access credentials in the

console, and Lake Formation reads and loads your data into the data lake. To import data from databases other than the ones listed above, you can create custom ETL jobs with AWS Glue.

Catalog and label your data

You can use AWS Glue crawlers to read your data in Amazon S3 and extract database and table schema and store that data in a searchable Data Catalog. Then, use Lake Formation Lake Formation tag-based access control (TBAC) to manage permissions on databases, tables, and columns. For more information about adding tables to the Data Catalog, see Creating objects in the AWS Glue Data Catalog.

Security management

Define and manage access controls

Lake Formation provides a single place to manage access controls for data in your data lake. You can define security policies that restrict access to data at the database, table, column, row, and cell levels. These policies apply to IAM users and roles, and to users and groups when federating through an external identity provider. You can use fine-grained controls to access data secured by Lake Formation within Amazon Redshift Spectrum, Athena, AWS Glue ETL, and Amazon EMR for Apache Spark. Whenever you create IAM identities, make sure to follow IAM best practices. For more information, see Security best practices in the IAM User Guide.

Hybrid access mode

Lake Formation hybrid access mode provides the flexibility to selectively enable Lake Formation permissions for databases and tables in your Data Catalog. With hybrid access mode, you now have an incremental path that allows you to set Lake Formation permissions for a specific set of users without interrupting the permission policies of other existing users or workloads. For more information, see Hybrid access mode.

Implement audit logging

Lake Formation provides comprehensive audit logs with CloudTrail to monitor access and show compliance with centrally defined policies. You can audit data access history across analytics and machine learning services that read the data in your data lake via Lake Formation. This lets you see which users or roles have attempted to access what data, with which services, and when. You can access audit logs in the same way you access any other CloudTrail logs using the CloudTrail APIs and console. For more information about CloudTrail logs see Logging AWS Lake Formation API calls using AWS CloudTrail.

Row and cell-level security

Lake Formation provides data filters that allow you to restrict access to a combination of columns and rows. Use row and cell-level security to protect sensitive data like Personal Identifiable Information (PII). For more information about row-level security, see Data filtering and cell-level security in Lake Formation.

Tag-based access control

Use Lake Formation attribute-based access control to manage hundreds or even thousands of data permissions by creating custom labels called LF-Tags. You can now define LF-Tags and attach them to databases, tables, or columns. Then, share controlled access across analytic, machine learning (ML), and extract, transform, and load (ETL) services for consumption. LF-Tags make sure that data governance can be scaled easily by replacing the policy definitions of thousands of resources with a few logical tags. Lake Formation provides a text-based search over this metadata, so your users can quickly find the data they need to analyze.

Attribute-based access control

Use attribute-based access control to grant access on Data Catalog objects. Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. AWS calls these attributes tags. You can use ABAC to grant access to principals within the same account or in another account on the Data Catalog resources. Any IAM principal with matching IAM tag or session tag keys and values gains access to the resource. You must have grantable permissions on the resources to make these grants.

Cross account access

Lake Formation permission management capabilities simplify securing and managing distributed data lakes across multiple AWS accounts through a centralized approach, providing fine-grained access control to the Data Catalog and Amazon S3 locations. For more information, see Cross-account data sharing in Lake Formation.

Bring your data into the Data Catalog

The federation capability allows you to create federated catalogs and set up permissions on datasets stored in different data sources like Amazon Redshift without migrating data or metadata into Amazon S3 or AWS Glue Data Catalog. You can use the following methods to bring data and manage permissions on external datasets in Lake Formation:

For more information, see Bringing your data into the AWS Glue Data Catalog.

- **Bringing data in Amazon Redshift data warehouses into the AWS Glue Data Catalog** – Register an existing Amazon Redshift namespace or a cluster with the Data Catalog, and create a multi-level federated catalog in the Data Catalog.

  You can access your data using any query engine compatible with Apache Iceberg REST catalog OpenAPI specification, such as Amazon EMR Serverless, and Amazon Athena.

  For more information, see Bringing Amazon Redshift data into the AWS Glue Data Catalog.
- **Federating into the Data Catalog from external data sources** – Connect the Data Catalog to external data sources using AWS Glue connections, and create federated catalogs to centrally manage access permissions on datasets using Lake Formation. No migration of metadata into the Data Catalog is necessary.

  For more information, see Federating into external data sources in the AWS Glue Data Catalog.
- **Integrating Amazon S3 Table Buckets with Data Catalog** – You can publish and catalog Amazon S3 Tables as Data Catalog objects and register the catalog as a Lake Formation data location from Lake Formation console or using AWS Glue APIs.

  For more information, see Amazon S3 Tables integration with AWS Glue Data Catalog and AWS Lake Formation.
- **Create catalogs to manage Amazon Redshift tables in the Data Catalog** – You may not have an Amazon Redshift producer cluster or an Amazon Redshift datashare available today, but want to create and manage Amazon Redshift tables using Data Catalog. You can get started by creating an AWS Glue managed catalog using the `glue:CreateCatalog` API or the AWS Lake Formation console by setting the catalog type as `Managed` and `Catalog source` as **Redshift**.

  For more information, see Creating an Amazon Redshift managed catalog in the AWS Glue Data Catalog.
- **Integrating Lake Formation with Amazon Redshift data sharing** – Use Lake Formation to centrally manage database, table, column, and row-level access permissions of Amazon Redshift datashares and restrict user access to objects within a datashare.
- **Connecting Data Catalog to external metastores** – Connect AWS Glue Data Catalog to external metastores to manage access permissions on data sets in Amazon S3 using Lake Formation. No migration of metadata into the Data Catalog is necessary.

  For more information, see Managing permissions on datasets that use external metastores.

- **Integrating Lake Formation with AWS Data Exchange** – Lake Formation supports licensing access to your data through AWS Data Exchange. If you're interested in licensing your Lake Formation data, see What is AWS Data Exchange in the *AWS Data Exchange User Guide*.

AWS Lake Formation: How it works

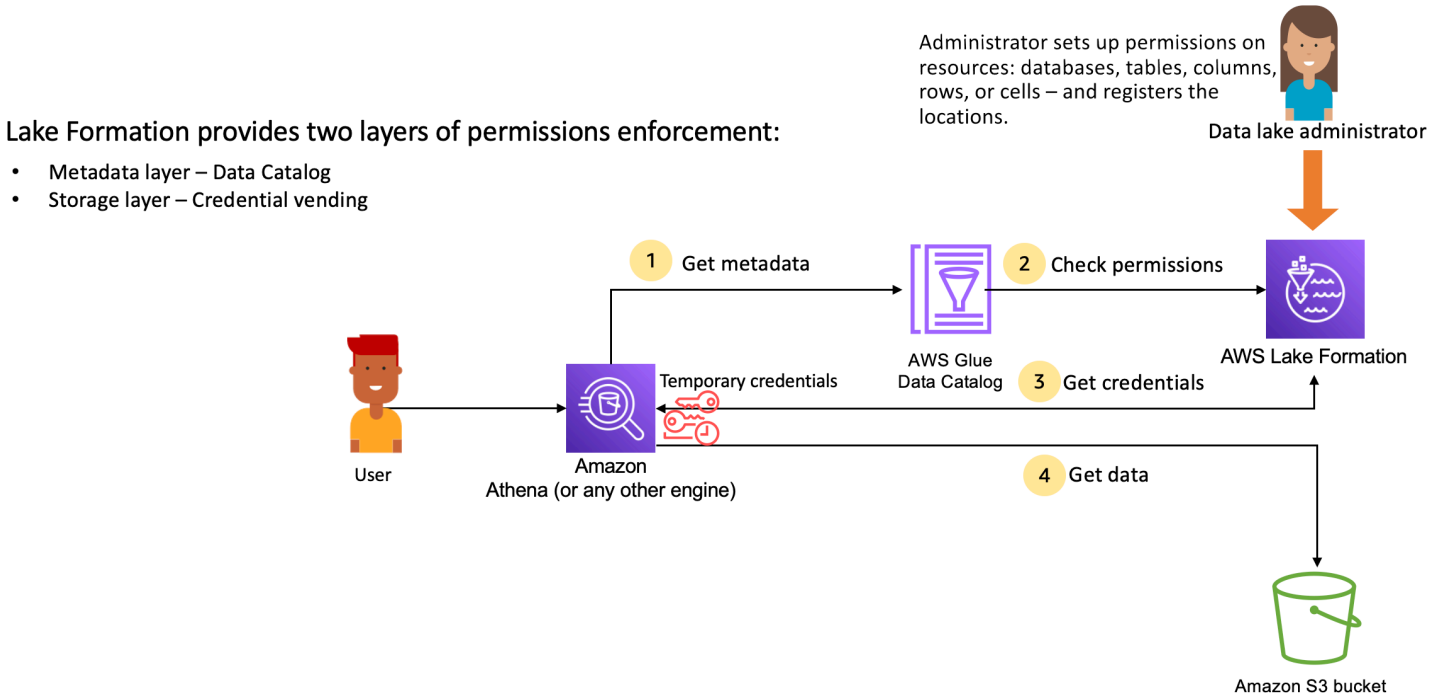
AWS Lake Formation provides a relational database management system (RDBMS) permissions model to grant or revoke access to Data Catalog resources such as databases, tables, and columns with underlying data in Amazon S3. The easy to manage Lake Formation permissions replace the complex Amazon S3 bucket policies and corresponding IAM policies.

In Lake Formation, you can implement permissions on two levels:

- Enforcing metadata-level permissions on the Data Catalog resources such as databases and tables
- Managing storage access permissions on the underlying data stored in Amazon S3 on behalf of integrated engines

Lake Formation permissions management workflow

Lake Formation integrates with analytical engines to query Amazon S3 data stores and metadata objects that are registered with Lake Formation. The following diagram illustrates how permissions management works in Lake Formation.



Lake Formation permissions management high-level steps

Before Lake Formation can provide access controls for data in your data lake, a *data lake administrator* or a user with administrative permissions sets up individual Data Catalog table user policies to allow or deny access to Data Catalog tables using Lake Formation permissions.

Then, either the data lake administrator or a user delegated by the administrator grants Lake Formation permissions to users on the Data Catalog databases and tables, and registers the Amazon S3 location of the table with Lake Formation.

1. **Get metadata** – A principal (user) submits a query or an ETL script to an integrated analytical engine such as Amazon Athena, AWS Glue, Amazon EMR, or Amazon Redshift Spectrum. The integrated analytical engine identifies the table that is being requested and sends a request for metadata to the Data Catalog.
2. **Check permissions** – The Data Catalog checks user's permissions with Lake Formation, and if the user is authorized to access the table, returns the metadata that the user is allowed to see to the engine.
3. **Get credentials** – The Data Catalog lets the engine know if the table is managed by Lake Formation or not. If the underlying data is registered with Lake Formation, the analytical engine requests Lake Formation to provide data access by granting temporary access.
4. **Get data** – If the user is authorized to access the table, Lake Formation provides temporary access to the integrated analytical engine. Using the temporary access, the analytical engine

fetches the data from Amazon S3, and performs necessary filtering such as column, row, or cell filtering. When the engine finishes running the job, it returns the results back to the user. This process is called credential vending.

If the table is not managed by Lake Formation, the second call from the analytic engine is made directly to Amazon S3. The concerned Amazon S3 bucket policy and IAM user policy are evaluated for data access.

Whenever you use IAM policies, make sure that you follow IAM best practices. For more information, see Security best practices in IAM in the *IAM User Guide*.

Topics

- Metadata permissions
- Storage access management
- Cross-account data sharing in Lake Formation

Metadata permissions

Lake Formation provides authorization and access control for the Data Catalog. When an IAM role makes a Data Catalog API call from any system, the Data Catalog verifies the user's data permissions and only returns the metadata that the user has permissions to access. For example, if an IAM role has access to only one table within a database, and a service or a user assuming the role performs the `GetTables` operation, the response will contain only the one table, regardless of the number of tables in the database.

Default settings - `IAMAllowedPrincipal` group permissions

AWS Lake Formation, by default, sets permissions to all databases and tables to a virtual group named `IAMAllowedPrincipal`. This group is unique and visible only within Lake Formation. The `IAMAllowedPrincipal` group includes all IAM principals who have access to Data Catalog resources through IAM principal policies and AWS Glue resource policies. If this permissions exists on a database or table, all principals will be granted access to the database or table.

If you want to provide more granular permissions on a database or table, remove `IAMAllowedPrincipal` permission and, Lake Formation enforces all other policies associated with that database or table. For example, if there is a policy that allows User A to access Database

A with DESCRIBE permissions, and the IAMAllowedPrincipal exists with all permissions, User A will continue to perform all other actions, until the IAMAllowedPrincipal permission is revoked.

Additionally, by default, the IAMAllowedPrincipal group has permissions on all new databases and tables when they are created. There are two configurations that control this behaviour. The first is at the account and Region-level that enables this for newly created databases, and the second is at the database level. To modify the default setting, see Change the default permission model or use hybrid access mode.

Granting permissions

Data lake administrators can grant Data Catalog permissions to principals so that the principals can create and manage databases and tables, and can access underlying data.

Database and table-level permissions

When you grant permissions within Lake Formation, the grantor must specify the principal to grant permissions to, the resources to grant permissions on, and the actions that the grantee should have access to perform. For most resources within Lake Formation, the principal list and resources to grant permissions are similar, but the actions that a grantee can perform differs based on the resource type. For example, SELECT permissions are available for tables to read the tables, but SELECT permissions are not allowed on databases. The CREATE_TABLE permission is permissible on databases, but not on tables.

You can grant AWS Lake Formation permissions using two methods:

- Named resource method – Allows you to choose database and table names while granting permissions to users.
- LF-Tag based access control (LF-TBAC) – Users create LF-Tags, associate them with Data Catalog resources, grant Describe permission on LF-Tags, associate permissions to individual users, and write LF permissions policies using LF-Tags to different users. Such LF-Tag-based policies apply to all Data Catalog resources that are associated with those LF-Tag values.

> **Note**
>
> LF-Tags are unique to Lake Formation. They are only visible in Lake Formation and should not be confused with AWS resource tags.

LF-TBAC is a feature that allows users to group resources into user-defined categories of LF-Tags and apply permissions on those resource groups. Hence, it is the best way to scale permissions across huge number of Data Catalog resources.

For more information, see Lake Formation tag-based access control.

When you grant permissions to a principal, Lake Formation evaluates permissions as a union of all the policies for that user. For example, if you have two policies on a table for a principal where one policy grants permissions to columns col1, col2, and col3 through named resource method, and the other policy grants permissions to the same table and principal to col5, and col6 through LF-Tags, the effective permissions will be a union of the permissions which would be col1, col2, col3, col5, and col6. This also includes data filters and rows.

Data location permissions

Data location permissions provides non-administrative users the ability to create databases and tables at specific Amazon S3 locations. If a user attempts to create a database or a table in a location that they don't have permissions to create, the creation task fails. This is to prevent users from creating tables in arbitrary locations within the data lake and provides control over where those users can read and write data. There is an implicit permission when creating tables in the Amazon S3 location within the database it is being created in. For more information, see Granting data location permissions.

Create table and database permissions

Non administrative users by default don't have permissions to create databases or tables within a database. Database creation is controlled at the account-level using the Lake Formation settings so that only authorized principals can create databases. For more information, see Creating a database. To create a table, a principal requires `CREATE_TABLE` permission on the database where the table is being created. For more information, see Creating tables.

Implicit and explicit permissions

Lake Formation provides implicit permissions depending on the persona and the actions that the persona performs. For example, data lake administrators automatically get `DESCRIBE` permissions to all resources within the Data Catalog, data location permissions to all locations, permissions to create databases and tables in all locations, as well as `Grant` and `Revoke` permissions on any resource. Database creators automatically get all database permissions on the databases that they

create, and table creators get all permissions on the tables that they create. For more information, see Implicit Lake Formation permissions.

Grantable permissions

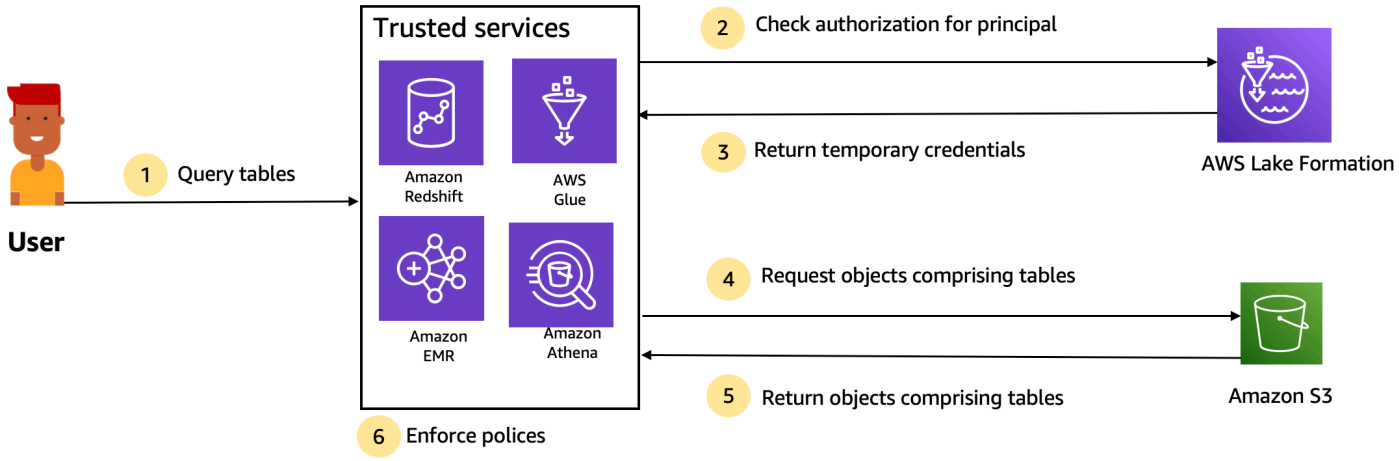
Data lake administrators have the ability to delegate the management of permissions to non administrative users by providing grantable permissions. When a principal is provided grantable permissions on a resource and a set of permissions, that principal gains the ability to grant permissions to other principals on that resource.

Storage access management

Lake Formation uses credential vending functionality to provide temporary access to Amazon S3 data. Credential vending, or token vending is a common pattern that provides temporary credentials to users, services, or some other entity for the purposes of granting short term access to a resource.

Lake Formation leverages this pattern to provide short term access to AWS analytics services such as Athena to access data on behalf of the calling principal. When granting permissions, users don't need to update their Amazon S3 bucket policies or IAM policies, and they don't need direct access to Amazon S3.

The following diagram shows how Lake Formation provides temporary access to registered locations:



Trusted services enforce AWS Lake Formation policies (distributed enforcement with fail close).

1. A principal (user) enters a query or request for data for a table through a trusted integrated service like Athena, Amazon EMR, Redshift Spectrum, or AWS Glue.

2. The integrated service checks for authorization from Lake Formation for the table and requested columns and makes an authorization determination. If the user is not authorized, Lake Formation denies access to data and the query fails.
3. After authorization succeeds and storage authorization is turned on for the table and user, the integrated service retrieves temporary credentials from Lake Formation to access the data.
4. The integrated service uses the temporary credentials from Lake Formation to request objects from Amazon S3.
5. Amazon S3 provides the Amazon S3 objects to the integrated service. The Amazon S3 objects contains all the data from the table.
6. The integrated service performs the necessary enforcement of Lake Formation policies, such as column level, row level and/or cell level filtering. The integrated service processes the queries and returns the results back to the user.

Enable storage-level permissions enforcement for Data Catalog tables

By default, storage-level enforcement is not enabled for tables within the Data Catalog. To enable storage-level enforcement, you must register the Amazon S3 location of your source data with Lake Formation and provide an IAM role. Storage-level permissions will be enabled for all tables with the same table location path or prefix of the Amazon S3 location.

When an integrated service requests access to the data location on behalf of a user, the Lake Formation service assumes this role and returns the credentials to requested service with scoped-down permissions to the resource so that data access can be made. The registered IAM role must have all required access to the Amazon S3 location including AWS KMS keys.

For more information, see Registering an Amazon S3 location.

Supported AWS services

AWS analytic services such as Athena, Redshift Spectrum, Amazon EMR, AWS Glue, Amazon Quick, and Amazon SageMaker AI integrate with AWS Lake Formation using the Lake Formation credential vending API operations. To see a full list of AWS services that integrate with Lake Formation, and the level of granularity and table formats that they support, see Working with other AWS services.

Cross-account data sharing in Lake Formation

With Lake Formation, you can share Data Catalog resources (databases and tables) within an AWS account and across accounts in a simple setup using the named resource method or LF-Tags. You

can share an entire database or select tables from a database to any IAM principals (IAM roles and users) in an account, to other AWS accounts at the account level, or directly to IAM principals in another account.

You can also share Data Catalog tables with data filters to restrict access to the details at the row-level and cell-level details. Lake Formation uses AWS Resource Access Manager (AWS RAM) to facilitate granting permissions between accounts. When a resource is shared between two accounts, AWS RAM sends invites to the recipient account. When a user accepts a AWS RAM share invitation, AWS RAM provides the necessary permissions to Lake Formation to have the Data Catalog resources available as well as enabled storage level enforcement. For more information, see Cross-account data sharing in Lake Formation.

When the data lake administrator of the recipient account accepts the AWS RAM share, the shared resources are available in the recipient account. The data lake administrator grants further Lake Formation permissions on the shared resource to additional IAM principals in the recipient account, if the administrator has `GRANTABLE` permissions on the shared resource.

However, the principals can't query the shared resources using Athena or Redshift Spectrum without a resource link. A resource link is an entity in the Data Catalog and is similar to a Linux-Symlink concept.

The data lake administrator of the recipient account creates a resource link on the shared resource. The administrator grants `Describe` permissions on the resource link with the required permissions on the original shared resource to additional users. A user in recipient account can then use the resource link to query the shared resource using Athena and Redshift Spectrum. For more information about resource links, see Creating resource links.

Lake Formation components

AWS Lake Formation relies on the interaction of several components to create and manage your data lake.

Lake Formation console

You use the Lake Formation console to define and manage your data lake and grant and revoke Lake Formation permissions. You can use blueprints on the console to discover, cleanse, transform, and ingest data. You can also enable or disable access to the console for individual Lake Formation users.

Lake Formation API and Command Line Interface

Lake Formation provides API operations through several language-specific SDKs and the AWS Command Line Interface (AWS CLI). The Lake Formation API works in conjunction with the AWS Glue API. The Lake Formation API focuses primarily on managing Lake Formation permissions, while the AWS Glue API provides a data catalog API and a managed infrastructure for defining, scheduling, and running ETL operations on your data.

For information about the AWS Glue API, see the AWS Glue Developer Guide. For information about using the AWS CLI, see the AWS CLI Command Reference.

Other AWS services

Lake Formation uses the following services:

- AWS Glue to orchestrate jobs and crawlers to transform data using the AWS Glue transforms.
- IAM to grant permissions policies to Lake Formation principals. The Lake Formation permission model augments the IAM permission model to secure your data lake.

Lake Formation terminology

The following are some important terms that you will encounter in this guide.

Data lake

The *data lake* is your persistent data that is stored in Amazon S3 and managed by Lake Formation using a Data Catalog. A data lake typically stores the following:

- Structured and unstructured data
- Raw data and transformed data

For an Amazon S3 path to be within a data lake, it must be *registered* with Lake Formation.

Data access

Lake Formation provides secure and granular access to data through a new grant/revoke permissions model that augments AWS Identity and Access Management (IAM) policies.

Analysts and data scientists can use the full portfolio of AWS analytic and machine learning services, such as Amazon Athena, to access the data. The configured Lake Formation security policies help ensure that users can access only the data that they are authorized to access.

Hybrid access mode

Hybrid access mode lets you secure and access the cataloged data using both Lake Formation permissions and IAM and Amazon S3 permissions. Hybrid access mode allows data administrators to onboard Lake Formation permissions selectively and incrementally, focusing on one data lake use case at a time.

Blueprint

A *blueprint* is a data management template that enables you to easily ingest data into a data lake. Lake Formation provides several blueprints, each for a predefined source type, such as a relational database or AWS CloudTrail logs. From a blueprint, you can create a workflow. Workflows consist of AWS Glue crawlers, jobs, and triggers that are generated to orchestrate the loading and update of data. Blueprints take the data source, data target, and schedule as input to configure the workflow.

Workflow

A *workflow* is a container for a set of related AWS Glue jobs, crawlers, and triggers. You create the workflow in Lake Formation, and it executes in the AWS Glue service. Lake Formation can track the status of a workflow as a single entity.

When you define a workflow, you select the blueprint upon which it is based. You can then run workflows on demand or on a schedule.

Workflows that you create in Lake Formation are visible in the AWS Glue console as a directed acyclic graph (DAG). Using the DAG, you can track the progress of the workflow and perform troubleshooting.

Data Catalog

The *Data Catalog* is your persistent metadata store. It is a managed service that lets you store, annotate, and share metadata in the AWS Cloud in the same way you would in an Apache Hive metastore. It provides a uniform repository where disparate systems can store and find metadata to track data in data silos, and then use that metadata to query and transform the data. Lake Formation uses the AWS Glue Data Catalog to store metadata about data lakes, data sources, transforms, and targets.

Metadata about data sources and targets is in the form of databases and tables. Tables store schema information, location information, and more. Databases are collections of tables. Lake Formation provides a hierarchy of permissions to control access to databases and tables in the Data Catalog.

Each AWS account has one Data Catalog per AWS Region.

Underlying data

Underlying data refers to the source data or data within the data lakes that Data Catalog tables point to.

Principal

A *principal* is an AWS Identity and Access Management (IAM) user or role or an Active Directory user.

Data lake administrator

A *data lake administrator* is a principal who can grant any principal (including self) any permission on any Data Catalog resource or data location. Designate a data lake administrator as the first user of the Data Catalog. This user can then grant more granular permissions of resources to other principals.

> **Note**
>
> IAM administrative users—users with the `AdministratorAccess` AWS managed policy —are not automatically data lake administrators. For example, they can't grant Lake Formation permissions on catalog objects unless they have been granted permissions to do so. However, they can use the Lake Formation console or API to designate themselves as data lake administrators.

For information about the capabilities of a data lake administrator, see Implicit Lake Formation permissions. For information about designating a user as a data lake administrator, see Create a data lake administrator.

AWS service integrations with Lake Formation

You can use Lake Formation to manage database, table, and column-level access permissions on data stored in Amazon S3. After your data is registered with Lake Formation, you can use AWS analytical services like AWS Glue, Amazon Athena, Amazon Redshift Spectrum, Amazon EMR to query the data. The following AWS services integrate with AWS Lake Formation and honor Lake Formation permissions.

AWS Service	Integration details
AWS Glue	Reference topic: Using AWS Lake Formation with AWS Glue AWS Glue and Lake Formation share the same Data Catalog. For console operations (such as viewing a list of tables) and all API operations, AWS Glue users can access only the databases and tables on which they have Lake Formation permissions.
Amazon Athena	Reference topic: Using AWS Lake Formation with Amazon Athena Use Lake Formation to allow or deny permissions to read data in Amazon S3. When Amazon Athena users select the AWS Glue catalog in the query editor, they can query only the databases, tables, and columns that they have Lake Formation permissions on. Queries using manifests are not supported. Currently, Lake Formation doesn't support managing permissio ns on write operations such as `VACUUM`, `MERGE`, `UPDATE` and `OPTIMIZE` on tables in Open Table Formats. In addition to principals who authenticate with Athena through AWS Identity and Access Management (IAM), Lake Formation supports Athena users who connect through the JDBC or ODBC driver and authenticate through SAML. Supported SAML providers include Okta and Microsoft Active Directory Federation Service (AD FS).
Amazon Redshift Spectrum	Reference topic: Using AWS Lake Formation with Amazon Redshift Spectrum

AWS Service	Integration details
	When Amazon Redshift users create an external schema on a database in the AWS Glue Data Catalog, they can query only the tables and columns in that schema on which they have Lake Formation permissions.
Amazon Quick Enterprise Edition	Reference: Using AWS Lake Formation with Quick When an Amazon Quick Enterprise Edition user queries a dataset in an Amazon S3 location, the user must have the Lake Formation SELECT permission on the data.
Amazon EMR	Reference: Using AWS Lake Formation with Amazon EMR You can integrate Lake Formation permissions when you create an Amazon EMR cluster with a runtime role. A runtime role is an IAM role that you associate with Amazon EMR jobs or queries, and then Amazon EMR uses this role to access AWS resources.

Lake Formation also works with AWS Key Management Service (AWS KMS) to enable you to more easily set up these integrated services to encrypt and decrypt data in Amazon Simple Storage Service (Amazon S3) locations.

Additional Lake Formation resources

For more information about AWS Lake Formation, we recommend that you continue to learn more about the concepts introduced in this guide by using the following resources:

Topics

- Blogs
- Tech talks and webinars
- Modern day architecture
- Data mesh resources
- Best practices guides

Blogs

- AWS Lake Formation 2022 year in review
- Highly resilient multi-Region modern data architecture
- Cross-account sharing using LF-Tags to direct IAM principals
- Lake Formation permissions inventory dashboard
- Event driven data mesh

Tech talks and webinars

- re:Invent 2020 – Data lakes: Easily build, secure, and share with AWS Lake Formation
- re:Invent 2022 – Building and operating a datalake on Amazon S3
- AWS Summit SF 2022 – Understanding and achieving a modern data architecture
- AWS Summit ATL 2022 – Modern data lakes with AWS Lake Formation, Amazon Redshift, and AWS Glue
- AWS Summit ANZ 2022 – Data lakes, lake houses and data mesh: what, why, and how?
- AWS Online Tech Talks – Simplifying permissions and governance in your data lake

Modern day architecture

- Modern day architecture patterns

Data mesh resources

- Build a modern data architecture and data mesh pattern at scale using AWS Lake Formation tag-based access control
- How JPMorgan Chase built a data mesh architecture to drive significant value to enhance their enterprise data platform
- Build a data mesh on AWS

Best practices guides

- AWS Lake Formation best practices guides

Getting started with Lake Formation

We recommend that you start with the following sections:

- AWS Lake Formation: How it works — Learn about essential terminology and how the various components interact.
- Getting started with Lake Formation — Get information about prerequisites, and complete important setup tasks.
- AWS Lake Formation tutorials — Follow step-by-step tutorials to learn how to use Lake Formation.
- Security in AWS Lake Formation — Understand how you can help secure access to data in Lake Formation.

Getting started with Lake Formation

If you haven't signed up for AWS or need assistance getting started, be sure to complete the following tasks.

Topics

- Complete initial AWS configuration tasks
- Set up AWS Lake Formation
- Upgrading AWS Glue data permissions to the AWS Lake Formation model
- AWS Lake Formation and interface VPC endpoints (AWS PrivateLink)

Complete initial AWS configuration tasks

To use AWS Lake Formation you must first complete the following tasks:

Topics

- Sign up for an AWS account

Sign up for an AWS account

To get started with AWS, you need an AWS account. There are two ways to sign up for AWS. If you want complete control over account configuration and where your content is stored, or if you have regulated workloads such as HIPAA or FedRAMP, use Sign up for AWS (advanced). If you are a new user and want to use a login you already own to sign in to AWS and create an AWS environment with preconfigured defaults in select AWS Regions, use Sign up for AWS.

Set up AWS Lake Formation

The following sections provide information on setting up Lake Formation for the first time. Not all of the topics in this section are required to start using Lake Formation. You can use the instructions to set up the Lake Formation permissions model to manage your existing AWS Glue Data Catalog objects and data locations in Amazon Simple Storage Service (Amazon S3).

1. Create a data lake administrator

2. Change the default permission model or use hybrid access mode
3. the section called “Configure an Amazon S3 location for your data lake”
4. the section called “Assign permissions to Lake Formation users”
5. the section called “Integrating IAM Identity Center”
6. the section called “(Optional) External data filtering settings”
7. the section called “(Optional) Grant access to the Data Catalog encryption key”
8. (Optional) Create an IAM role for workflows

This section shows you how to set up Lake Formation resources in two different ways:

- Using an AWS CloudFormation template
- Using the Lake Formation console

To set up Lake Formation using AWS console, go to Create a data lake administrator.

Set up Lake Formation resources using CloudFormation template

> **Note**
>
> The CloudFormation stack performs steps 1 to 6 of the above, except step 2 and 5. Perform Change the default permission model or use hybrid access mode and the section called “Integrating IAM Identity Center” manually from the Lake Formation console.

1. Sign into the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation as an IAM administrator in the US East (N. Virginia) Region.
2. Choose Launch Stack.
3. Choose **Next** on the **Create stack** screen.
4. Enter a **Stack name.**
5. For **DatalakeAdminName** and **DatalakeAdminPassword**, enter your user name and password for data lake admin user.
6. For **DatalakeUser1Name** and **DatalakeUser1Password**, enter your user name and password for data lake analyst user.
7. For **DataLakeBucketName**, enter your new bucket name that will be created.

8. Choose **Next**.
9. On the next page, choose `I acknowledge that CloudFormation might create IAM resources with custom names` and choose **Next**.
10. Review the details on the final page and select **I acknowledge that AWS CloudFormation might create IAM resources.**
11. Choose **Create.**

    The stack creation can take up to two minutes.

Clean up resources

If you like to clean up the CloudFormation stack resources:

1. De-register the Amazon S3 bucket that your stack created and registered as a data lake location.
2. Delete the CloudFormation Stack. This will delete all the resources created by the stack.

Create a data lake administrator

Data lake administrators are initially the only AWS Identity and Access Management (IAM) users or roles that can grant Lake Formation permissions on data locations and Data Catalog resources to any principal (including self). For more information about data lake administrator capabilities, see Implicit Lake Formation permissions. By default, Lake Formation allows you to create upto 30 data lake administrators.

You can create a data lake administrator using the Lake Formation console or the `PutDataLakeSettings` operation of the Lake Formation API.

The following permissions are required to create a data lake administrator. The `Administrator` user has these permissions implicitly.

- `lakeformation:PutDataLakeSettings`
- `lakeformation:GetDataLakeSettings`

If you grant a user the `AWSLakeFormationDataAdmin` policy, that user will not be able to create additional Lake Formation administrator users.

To create a data lake administrator (console)

1. If the user who is to be a data lake administrator does not yet exist, use the IAM console to create it. Otherwise, choose an existing user who is to be the data lake administrator.

   > **Note**
   >
   > We recommend that you do not select an IAM administrative user (user with the `AdministratorAccess` AWS managed policy) to be the data lake administrator.

   Attach the following AWS managed policies to the user:

Policies	Mandatory?	Notes
`AWSLakeFormationDataAdmin`	Mandatory	Basic data lake administrator permissions. This AWS managed policy contains an explict deny for the Lake Formation API operation , `PutDataLakeSetting` that restricts users from creating new data lake administrators.
`AWSGlueConsoleFullAccess` , `CloudWatchLogsReadOnlyAccess`	Optional	Attach these policies if the data lake administrator will be troublesh ooting workflows created from Lake Formation blueprints. These policies enable the data lake administrator to view troubleshooting informati on in the AWS Glue console and the Amazon CloudWatch Logs console. For information about workflows, see the section called “Importing data using workflows”.

Policies	Mandatory?	Notes
`AWSLakeFormationCrossAccountManager`	Optional	Attach this policy to enable the data lake administrator to grant and revoke cross-account permissions on Data Catalog resources. For more information, see Cross-account data sharing in Lake Formation.
`AmazonAthenaFullAccess`	Optional	Attach this policy if the data lake administrator will be running queries in Amazon Athena.

2. Attach the following inline policy, which grants the data lake administrator permission to create the Lake Formation service-linked role. A suggested name for the policy is `LakeFormationSLR`.

   The service-linked role enables the data lake administrator to more easily register Amazon S3 locations with Lake Formation. For more information about the Lake Formation service-linked role, see the section called "Using service-linked roles".

   > **Important**
   >
   > In all the following policy, replace *<account-id>* with a valid AWS account number.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lakeformation.amazonaws.com"
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::<account-id>:role/aws-service-role/
lakeformation.amazonaws.com/AWSServiceRoleForLakeFormationDataAccess"
        }
    ]
}
```

3. (Optional) Attach the following `PassRole` inline policy to the user. This policy enables the data lake administrator to create and run workflows. The `iam:PassRole` permission enables the workflow to assume the role `LakeFormationWorkflowRole` to create crawlers and jobs, and to attach the role to the created crawlers and jobs. A suggested name for the policy is `UserPassRole`.

   Important

   Replace *<account-id>* with a valid AWS account number.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PassRolePermissions",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/LakeFormationWorkflowRole"
            ]
        }
    ]
}
```

4. (Optional) Attach this additional inline policy if your account will be granting or receiving cross-account Lake Formation permissions. This policy enables the data lake administrator to

view and accept AWS Resource Access Manager (AWS RAM) resource share invitations. Also, for data lake administrators in the AWS Organizations management account, the policy includes a permission to enable cross-account grants to organizations. For more information, see Cross-account data sharing in Lake Formation.

A suggested name for the policy is `RAMAccess`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ram:AcceptResourceShareInvitation",
                "ram:RejectResourceShareInvitation",
                "ec2:DescribeAvailabilityZones",
                "ram:EnableSharingWithAwsOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

5. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/ and sign in as a user with the `AdministratorAccess` AWS managed policy.
6. If a **Welcome to Lake Formation** window appears, choose the IAM user that you created or selected in Step 1, and then choose **Get started**.
7. If you do not see a **Welcome to Lake Formation** window, then perform the following steps to configure a Lake Formation Administrator.

   a. In the navigation pane, under **Administration**, choose **Administrative roles and tasks**. In the **Data lake administrators** section of the console page, choose **Add**.

   b. In the **Add administrators** dialog box, under Access type, choose **Data lake administrator**.

   c. For **IAM users and roles**, choose the IAM user that you created or selected in Step 1, and then choose **Save**.

Change the default permission model or use hybrid access mode

Lake Formation starts with the "Use only IAM access control" settings enabled for compatibility with existing AWS Glue Data Catalog behavior. This settings allows you to manage access to your data in the data lake and its metadata through IAM policies and Amazon S3 bucket policies.

To ease the transition of data lake permissions from an IAM and Amazon S3 model to Lake Formation permissions, we recommend you to use hybrid access mode for Data Catalog. With the hybrid access mode, you have an incremental path where you can enable Lake Formation permissions for a specific set of users without interrupting other existing users or workloads.

For more information, see Hybrid access mode.

Disable the default settings to move all existing users of a table to Lake Formation in a single step.

> **Important**
>
> If you have existing AWS Glue Data Catalog databases and tables, do not follow the instructions in this section. Instead, follow the instructions in the section called “Upgrading AWS Glue data permissions to the Lake Formation model”.

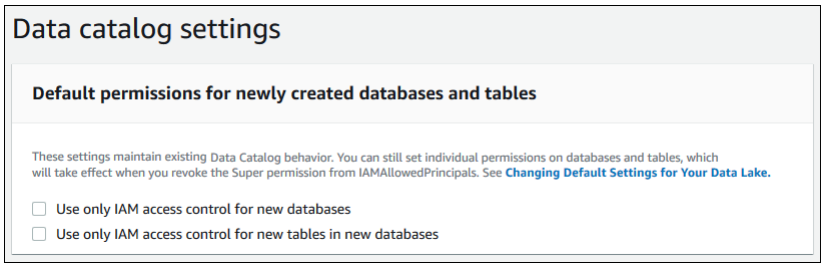
> **Warning**
>
> If you have automation in place that creates databases and tables in the Data Catalog, the following steps might cause the automation and downstream extract, transform, and load (ETL) jobs to fail. Proceed only after you have either modified your existing processes or granted explicit Lake Formation permissions to the required principals. For information about Lake Formation permissions, see the section called “Lake Formation permissions reference”.

To change the default Data Catalog settings

1. Continue in the Lake Formation console at https://console.aws.amazon.com/lakeformation/. Ensure that you are signed in as a user with the `AdministratorAccess` AWS managed policy.
2. Modify the Data Catalog settings:

   a. In the navigation pane, under **Administration**, choose **Data Catalog settings**.

b. Clear both check boxes and choose **Save**.



3. Revoke `IAMAllowedPrincipals` permission for database creators.

   a. In the navigation pane, under **Administration**, choose **Administrative roles and tasks**.

   b. In the **Administrative roles and tasks** console page, in the **Database creators** section, select the `IAMAllowedPrincipals` group, and choose **Revoke**.

      The **Revoke** permissions dialog box appears, showing that `IAMAllowedPrincipals` has the **Create database** permission.

   c. Choose **Revoke**.

Assign permissions to Lake Formation users

Create a user to have access to the data lake in AWS Lake Formation. This user has the least-privilege permissions to query the data lake.

For more information on creating users or groups, see IAM identities in the IAM User Guide.

To attach permissions to a non-administrator user to access Lake Formation data

1. Open the IAM console at https://console.aws.amazon.com/iam and sign in as a user with the `AdministratorAccess` AWS managed policy.
2. Choose **Users** or **User groups**.
3. In the list, choose the name of the user or group to embed a policy in.

   Choose **Permissions**.
4. Choose **Add permissions**, and choose **Attach policies directly**. Enter `Athena` in the **Filter policies** text field. In the result list, check the box for `AmazonAthenaFullAccess`.
5. Choose the **Create policy** button. On the **Create policy** page, choose the **JSON** tab. Copy and paste the following code into the policy editor.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess",
                "glue:GetTable",
                "glue:GetTables",
                "glue:SearchTables",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetPartitions",
                "lakeformation:GetResourceLFTags",
                "lakeformation:ListLFTags",
                "lakeformation:GetLFTag",
                "lakeformation:SearchTablesByLFTags",
                "lakeformation:SearchDatabasesByLFTags"
            ],
            "Resource": "*"
        }
    ]
}
```

6. Choose **Next** until you see the **Review policy** page. Enter a name for the policy, for example, `DatalakeUserBasic`. Choose **Create policy**, then close the **Policies** tab or browser window.

Configure an Amazon S3 location for your data lake

To use Lake Formation to manage and secure the data in your data lake, you must first register an Amazon S3 location. When you register a location, that Amazon S3 path and all folders under that path are registered, which enables Lake Formation to enforce storage level permissions. When the user requests data from an integrated engine like Amazon Athena, Lake Formation provides data access rather than using the users permissions.

When you register a location, you specify an IAM role that grants read/write permissions on that location. Lake Formation assumes that role when supplying temporary credentials to integrated

AWS services that request access to data in the registered Amazon S3 location. You can specify either the Lake Formation service-linked role (SLR) or create your own role.

Use a custom role in the following situations:

- You plan to publish metrics in Amazon CloudWatch Logs. The user-defined role must include a policy for adding logs in CloudWatch Logs and publishing metrics in addition to the SLR permissions. For an example inline policy that grants the necessary CloudWatch permissions, see Requirements for roles used to register locations.
- The Amazon S3 location exists in a different account. For details, see the section called "Registering an Amazon S3 location in another AWS account".
- The Amazon S3 location contains data encrypted with an AWS managed key. For details, see Registering an encrypted Amazon S3 location and Registering an encrypted Amazon S3 location across AWS accounts.
- You plan to access the Amazon S3 location using Amazon EMR. For more information about the role requirements, see IAM roles for Lake Formation in the *Amazon EMR Management Guide*.

The role that you choose must have the necessary permissions, as described in Requirements for roles used to register locations. For instructions on how to register an Amazon S3 location, see Adding an Amazon S3 location to your data lake.

(Optional) External data filtering settings

If you intend to analyze and process data in your data lake using third-party query engines, you must opt in to allow external engines to access data managed by Lake Formation. If you don't opt in, external engines will not be able to access data in Amazon S3 locations that are registered with Lake Formation.

Lake Formation supports column-level permissions to restrict access to specific columns in a table. Integrated analytic services like Amazon Athena, Amazon Redshift Spectrum, and Amazon EMR retrieve non-filtered table metadata from the AWS Glue Data Catalog. The actual filtering of columns in query responses is the responsibility of the integrated service. It's the responsibility of third-party administrators to properly handle permissions to avoid unauthorized access to data.

To opt in to allow third-party engines to access and filter data (console)

1. Continue in the Lake Formation console at https://console.aws.amazon.com/lakeformation/. Ensure that you are signed in as a principal that has the IAM permission on the Lake Formation

`PutDataLakeSettings` API operation. The IAM administrator user that you created in Sign up for an AWS account has this permission.

2. In the navigation pane, under **Administration**, choose **Application integration settings**.
3. On the **Application integration settings** page, do the following:

   a. Check the box **Allow external engines to filter data in Amazon S3 locations registered with Lake Formation**.

   b. Enter **Session tag values** defined for third-party engines.

   c. For **AWS account IDs**, enter the account IDs from where third-party engines are allowed to access locations registered with Lake Formation. Press **Enter** after each account ID.

   d. Choose **Save**.

To allow external engines to access data without session tag validation, see Application integration for full table access

(Optional) Grant access to the Data Catalog encryption key

If the AWS Glue Data Catalog is encrypted, grant AWS Identity and Access Management (IAM) permissions on the AWS KMS key to any principals who need to grant Lake Formation permissions on Data Catalog databases and tables.

For more information, see the *AWS Key Management Service Developer Guide*.

(Optional) Create an IAM role for workflows

With AWS Lake Formation, you can import your data using *workflows* that are executed by AWS Glue crawlers. A workflow defines the data source and schedule to import data into your data lake. You can easily define workflows using the *blueprints*, or templates that Lake Formation provides.

When you create a workflow, you must assign it an AWS Identity and Access Management (IAM) role that grants Lake Formation the necessary permissions to ingest the data.

The following procedure assumes familiarity with IAM.

To create an IAM role for workflows

1. Open the IAM console at https://console.aws.amazon.com/iam and sign in as a user with the `AdministratorAccess` AWS managed policy.
2. In the navigation pane, choose **Roles**, then **Create role**.

3. On the **Create role** page, choose **AWS service**, and then choose **Glue**. Choose **Next**.
4. On the **Add permissions** page, search for the **AWSGlueServiceRole** managed policy, and select the checkbox next to the policy name in the list. Then complete the **Create role** wizard, naming the role `LFWorkflowRole`. To finish, choose **Create role**.
5. Back on the **Roles** page, search for `LFWorkflowRole`, and choose the role name.
6. On the role **Summary** page, under the **Permissions** tab, choose **Create inline policy**. On the **Create policy** screen, navigate to the JSON tab, and add the following inline policy. A suggested name for the policy is `LakeFormationWorkflow`.

> **Important**
>
> In the following policy, replace *<account-id>* with a valid AWS account number.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "lakeformation:GetDataAccess",
                 "lakeformation:GrantPermissions"
             ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": ["iam:PassRole"],
            "Resource": [
                "arn:aws:iam::111122223333:role/LakeFormationWorkflowRole"
            ]
        }
    ]
}
```

The following are brief descriptions of the permissions in this policy:

- `lakeformation:GetDataAccess` enables jobs created by the workflow to write to the target location.
- `lakeformation:GrantPermissions` enables the workflow to grant the `SELECT` permission on target tables.
- `iam:PassRole` enables the service to assume the role `LakeFormationWorkflowRole` to create crawlers and jobs (instances of workflows), and to attach the role to the created crawlers and jobs.

7. Verify that the role `LakeFormationWorkflowRole` has two policies attached.
8. If you are ingesting data that is outside the data lake location, add an inline policy granting permissions to read the source data.

Upgrading AWS Glue data permissions to the AWS Lake Formation model

AWS Lake Formation permissions enable fine-grained access control for data in your data lake. You can use the Lake Formation permissions model to manage your existing AWS Glue Data Catalog objects and data locations in Amazon Simple Storage Service (Amazon S3).

The Lake Formation permissions model uses coarse-grained AWS Identity and Access Management (IAM) permissions for API service access. Lake Formation uses Data filtering and cell-level security in Lake Formation functionality to restrict table access at the column, row, and cell-level for users and their applications. By comparison, the AWS Glue model grants data access via Identity based and resource based IAM policies.

To make the switch, follow the steps in this guide.

For more information, see Overview of Lake Formation permissions.

About default permissions

To maintain backward compatibility with AWS Glue, by default, AWS Lake Formation grants the `Super` permission to the `IAMAllowedPrincipals` group on all existing AWS Glue Data Catalog resources, and grants the `Super` permission on new Data Catalog resources if the **Use only IAM access control** settings are enabled. This effectively causes access to Data Catalog resources and Amazon S3 locations to be controlled solely by AWS Identity and Access Management (IAM)

policies. The `IAMAllowedPrincipals` group includes any IAM users and roles that are allowed access to your Data Catalog objects by your IAM policies. The `Super` permission enables a principal to perform every supported Lake Formation operation on the database or table on which it is granted.

You can start using Lake Formation to manage access to your data by registering the locations of existing Data Catalog resources in Lake Formation or by using hybrid access mode. When you register Amazon S3 location in hybrid access mode, you can enable Lake Formation permissions by opting in principals for databases and tables under that location.

To ease the transition of data lake permissions from an IAM and Amazon S3 model to Lake Formation permissions, we recommend you to use hybrid access mode for Data Catalog. With the hybrid access mode, you have an incremental path where you can enable Lake Formation permissions for a specific set of users without interrupting other existing users or workloads.

For more information, see Hybrid access mode.

Disable the default Data Catalog settings to move all existing users of a table to Lake Formation in a single step.

To start using Lake Formation permissions with your existing AWS Glue Data Catalog databases and tables, you must do the following:

1. Determine your users' existing IAM permissions for each database and table.
2. Replicate these permissions in Lake Formation.
3. For each Amazon S3 location that contains data:
   a. Revoke the `Super` permission from the `IAMAllowedPrincipals` group on each Data Catalog resource that references that location.
   b. Register the location with Lake Formation.
4. Clean up existing fine-grained access control IAM policies.

> **Important**
>
> To add new users while in the process of transitioning your Data Catalog, you must set up granular AWS Glue permissions in IAM as before. You also must replicate those permissions in Lake Formation as described in this section. If new users have the coarse-grained IAM policies that are described in this guide, they can list any databases or tables that have the

`Super` permission granted to `IAMAllowedPrincipals`. They can also view the metadata for those resources.

Follow the steps in this section to upgrade to the Lake Formation permissions model.

Topics

- Step 1: List users' and roles' existing permissions
- Step 2: Set up equivalent Lake Formation permissions
- Step 3: Give users IAM permissions to use Lake Formation
- Step 4: Switch your data stores to the Lake Formation permissions model
- Step 5: Secure new Data Catalog resources
- Step 6: Give users a new IAM policy for future data lake access
- Step 7: Clean up existing IAM policies

Step 1: List users' and roles' existing permissions

To start using AWS Lake Formation permissions with your existing AWS Glue databases and tables, you must first determine your users' existing permissions.

Important

Before you begin, ensure that you have completed the tasks in *Getting started*.

Topics

- Using the API operation
- Using the AWS Management Console
- Using AWS CloudTrail

Using the API operation

Use the AWS Identity and Access Management (IAM) ListPoliciesGrantingServiceAccess API operation to determine the IAM policies attached to each principal (user or role). From the policies

returned in the results, you can determine the IAM permissions that are granted to the principal. You must invoke the API for each principal separately.

Example

The following AWS CLI example returns the policies attached to user `glue_user1`.

```
aws iam list-policies-granting-service-access --arn arn:aws:iam::111122223333:user/
glue_user1 --service-namespaces glue
```

The command returns results similar to the following.

```
{
    "PoliciesGrantingServiceAccess": [
        {
            "ServiceNamespace": "glue",
            "Policies": [
                {
                    "PolicyType": "INLINE",
                    "PolicyName": "GlueUserBasic",
                    "EntityName": "glue_user1",
                    "EntityType": "USER"
                },
                {
                    "PolicyType": "MANAGED",
                    "PolicyArn": "arn:aws:iam::aws:policy/AmazonAthenaFullAccess",
                    "PolicyName": "AmazonAthenaFullAccess"
                }
            ]
        }
    ],
    "IsTruncated": false
}
```

Using the AWS Management Console

You can also see this information on the AWS Identity and Access Management (IAM) console, in the **Access Advisor** tab on the user or role **Summary** page:

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users** or **Roles**.

3. Choose a name in the list to open its **Summary** page, and choose the **Access Advisor** tab.
4. Inspect each of the policies to determine the combination of databases, tables, and actions that each user has permissions for.

   Remember to inspect roles in addition to users during this process because your data processing jobs might be assuming roles to access data.

Using AWS CloudTrail

Another way to determine your existing permissions is to look in AWS CloudTrail for AWS Glue API calls where the `additionaleventdata` field of the logs contains an `insufficientLakeFormationPermissions` entry. This entry lists the database and table that the user needs Lake Formation permissions on to take the same action.

These are data access logs, so they are not guaranteed to produce a comprehensive list of users and their permissions. We recommend choosing a wide time range to capture most of your users' data access patterns, for example, several weeks or months.

For more information, see Viewing Events with CloudTrail Event History in the *AWS CloudTrail User Guide*.

Next, you can set up Lake Formation permissions to match the AWS Glue permissions. See Step 2: Set up equivalent Lake Formation permissions.

Step 2: Set up equivalent Lake Formation permissions

Using the information collected in Step 1: List users' and roles' existing permissions, grant AWS Lake Formation permissions to match the AWS Glue permissions. Use any of the following methods to performs the grants:

- Use the Lake Formation console or the AWS CLI.

  See the section called "Granting data permissions".
- Use the `GrantPermissions` or `BatchGrantPermissions` API operations.

  See Permissions APIs.

For more information, see Overview of Lake Formation permissions.

After setting up Lake Formation permissions, proceed to Step 3: Give users IAM permissions to use Lake Formation.

Step 3: Give users IAM permissions to use Lake Formation

To use the AWS Lake Formation permissions model, principals must have AWS Identity and Access Management (IAM) permissions on the Lake Formation APIs.

Create the following policy in IAM and attach it to every user who needs access to your data lake. Name the policy `LakeFormationDataAccess`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "LakeFormationDataAccess",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": "*"
        }
    ]
}
```

Next, upgrade to Lake Formation permissions one data location at a time. See Step 4: Switch your data stores to the Lake Formation permissions model.

Step 4: Switch your data stores to the Lake Formation permissions model

Upgrade to Lake Formation permissions one data location at a time. To do that, repeat this entire section until you have registered all Amazon Simple Storage Service (Amazon S3) paths that are referenced by your Data Catalog.

Topics

- Verify Lake Formation permissions
- Secure existing Data Catalog resources
- Turn on Lake Formation permissions for your Amazon S3 location

Verify Lake Formation permissions

Before registering a location, perform a verification step to ensure that the correct principals have the required Lake Formation permissions, and that no Lake Formation permissions are granted to principals that should not have them. Using the Lake Formation `GetEffectivePermissionsForPath` API operation, identify the Data Catalog resources that reference the Amazon S3 location, along with the principals that have permissions on those resources.

The following AWS CLI example returns the Data Catalog databases and tables that reference the Amazon S3 bucket `products`.

```
aws lakeformation get-effective-permissions-for-path --resource-arn
 arn:aws:s3:::products --profile datalake_admin
```

Note the `profile` option. We recommend that you run the command as a data lake administrator.

The following is an excerpt from the returned results.

```
{
        "PermissionsWithGrantOption": [
            "SELECT"
        ],
        "Resource": {
            "TableWithColumns": {
                "Name": "inventory_product",
                "ColumnWildcard": {},
                "DatabaseName": "inventory"
            }
        },
        "Permissions": [
            "SELECT"
        ],
        "Principal": {
            "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/
datalake_user1",
```

```
            "DataLakePrincipalType": "IAM_USER"
        }
},...
```

> **Important**
>
> If your AWS Glue Data Catalog is encrypted, `GetEffectivePermissionsForPath` returns only databases and tables that were created or modified after Lake Formation general availability.

Secure existing Data Catalog resources

Next, revoke the `Super` permission from `IAMAllowedPrincipals` on each table and database that you identified for the location.

> **Warning**
>
> If you have automation in place that creates databases and tables in the Data Catalog, the following steps might cause the automation and downstream extract, transform, and load (ETL) jobs to fail. Proceed only after you have either modified your existing processes or granted explicit Lake Formation permissions to the required principals. For information about Lake Formation permissions, see the section called “Lake Formation permissions reference”.

To revoke `Super` from `IAMAllowedPrincipals` on a table

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as a data lake administrator.
2. In the navigation pane, choose **Tables**.
3. On the **Tables** page, select the radio button next to the desired table.
4. On the **Actions** menu, choose **Revoke**.
5. In the **Revoke permissions** dialog box, in the **IAM users and roles** list, scroll down to the **Group** heading, and choose **IAMAllowedPrincipals**.
6. Under **Table permissions**, ensure that **Super** is selected, and then choose **Revoke**.

To revoke Super from IAMAllowedPrincipals on a database

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as a data lake administrator.
2. In the navigation pane, choose **Databases**.
3. On the **Databases** page, select the radio button next to the desired database.
4. On the **Actions** menu, choose **Edit**.
5. On the **Edit database** page, clear **Use only IAM access control for new tables in this database**, and then choose **Save**.
6. Back on the **Databases** page, ensure that the database is still selected, and then on the **Actions** menu, choose **Revoke**.
7. In the **Revoke permissions** dialog box, in the **IAM users and roles** list, scroll down to the **Group** heading, and choose **IAMAllowedPrincipals**.
8. Under **Database permissions**, ensure that **Super** is selected, and then choose **Revoke**.

Turn on Lake Formation permissions for your Amazon S3 location

Next, register the Amazon S3 location with Lake Formation. To do this, you can use the process described in Adding an Amazon S3 location to your data lake. Or, use the `RegisterResource` API operation as described in Credential vending APIs.

> **Note**
>
> If a parent location is registered, you don't need to register child locations.

After you finish these steps and test that your users can access their data, you have successfully upgraded to Lake Formation permissions. Continue with the next step, Step 5: Secure new Data Catalog resources.

Step 5: Secure new Data Catalog resources

Next, secure all new Data Catalog resources by changing the default Data Catalog settings. Turn off the options to use only AWS Identity and Access Management (IAM) access control for new databases and tables.

> **Warning**
>
> If you have automation in place that creates databases and tables in the Data Catalog, the following steps might cause the automation and downstream extract, transform, and load (ETL) jobs to fail. Proceed only after you have either modified your existing processes or granted explicit Lake Formation permissions to the required principals. For information about Lake Formation permissions, see the section called "Lake Formation permissions reference".

To change the default Data Catalog settings

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as an IAM administrative user (the user `Administrator` or another user with the `AdministratorAccess` AWS managed policy).
2. In the navigation pane, choose **Settings**.
3. On the **Data catalog settings** page, clear both check boxes, and then choose **Save**.

The next step is to grant users access to additional databases or tables in the future. See Step 6: Give users a new IAM policy for future data lake access.

Step 6: Give users a new IAM policy for future data lake access

To grant your users access to additional Data Catalog databases or tables in the future, you must give them the coarse-grained AWS Identity and Access Management (IAM) inline policy that follows. Name the policy `GlueFullReadAccess`.

> **Important**
>
> If you attach this policy to a user before revoking `Super` from `IAMAllowedPrincipals` on every database and table in your Data Catalog, that user can view all metadata for any resource on which `Super` is granted to `IAMAllowedPrincipals`.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GlueFullReadAccess",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess",
                "glue:GetTable",
                "glue:GetTables",
                "glue:SearchTables",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetPartitions"
            ],
            "Resource": "*"
        }
    ]
}
```

Note

The inline policies designated in this step and previous steps contain minimal IAM permissions. For suggested policies for data lake administrators, data analysts, and other personas, see the section called “Lake Formation personas and IAM permissions reference”.

Next, proceed to Step 7: Clean up existing IAM policies.

Step 7: Clean up existing IAM policies

After you set up the AWS Lake Formation permissions and you create and attach the coarse-grained access control AWS Identity and Access Management (IAM) policies, complete the following final step:

- Remove from users, groups, and roles the old fine-grained access control IAM policies that you replicated in Lake Formation.

By doing this, you ensure that those principals no longer have direct access to the data in Amazon Simple Storage Service (Amazon S3). You can then manage data lake access for those principals entirely through Lake Formation.

AWS Lake Formation and interface VPC endpoints (AWS PrivateLink)

Amazon VPC is an AWS service that you can use to launch AWS resources in a virtual network that you define. With a VPC, you have control over your network settings, such the IP address range, subnets, route tables, and network gateways.

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC and Lake Formation. You use this connection so that Lake Formation can communicate with the resources in your VPC without going through the public internet.

You can establish a private connection between your VPC and AWS Lake Formation by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access Lake Formation APIs without an internet gateway, NAT device, VPN connection, or Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with Lake Formation APIs. Traffic between your VPC and Lake Formation does not leave the Amazon network.

Each interface endpoint is represented by one or more Elastic Network Interfaces in your subnets.

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Considerations for Lake Formation VPC endpoints

Before you set up an interface VPC endpoint for Lake Formation, ensure that you review Interface endpoint properties and limitations in the *Amazon VPC User Guide*.

Lake Formation supports making calls to all of its API actions from your VPC. You can use Lake Formation with VPC endpoints in all AWS Regions that support both Lake Formation and Amazon VPC endpoints.

Creating an interface VPC endpoint for Lake Formation

You can create a VPC endpoint for the Lake Formation service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

Create a VPC endpoint for Lake Formation using the following service name:

- com.amazonaws.*region*.lakeformation

If you enable private DNS for the endpoint, you can make API requests to Lake Formation using its default DNS name for the Region, for example, `lakeformation.us-east-1.amazonaws.com`.

For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for Lake Formation

Lake Formation supports VPC endpoint policies. An endpoint policy is a resource-based policy that you attach to a VPC endpoint to control which AWS principals can use the endpoint to access an AWS service.

You can attach an endpoint policy to your VPC endpoint that controls access to Lake Formation. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Example: VPC endpoint policy for Lake Formation actions

The following example VPC endpoint policy for Lake Formation allows for credential vending using Lake Formation permissions. You might use this policy to run queries using Lake Formation permissions from an Amazon Redshift cluster or an Amazon EMR cluster located in a private subnet.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lakeformation:GetDataAccess",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

> **Note**
>
> If you don't attach a policy when you create an endpoint, a default policy that allows full access to the service is attached.

For more information, see these topics in the Amazon VPC documentation:

- What Is Amazon VPC?
- Create an Interface Endpoint
- Use VPC endpoint policies

AWS Lake Formation tutorials

The following tutorials are organized into three tracks and provide step-by-step instructions on how to build a data lake, ingest data, share, and secure data lakes using AWS Lake Formation:

1. **Build a data lake and ingest data:** Learn to build a data lake and use blueprints to move, store, catalog, clean, and organize your data. You will also learn to set up governed tables. A governed table is a new Amazon S3 table type that supports atomic, consistent, isolated, and durable (ACID) transactions.

   Before you begin, make sure that you have completed the steps in Getting started with Lake Formation.

   - Creating a data lake from an AWS CloudTrail source

     Create and load your first data lake by using your own CloudTrail logs as the data source.

   - Creating a data lake from a JDBC source in Lake Formation

     Create a data lake by using one of your JDBC-accessible data stores, such as a relational database, as the data source.

2. **Securing data lakes:** Learn to use tag-based and row-level access controls to effectively secure and manage access to your data lakes.

   - Setting up permissions for open table storage formats in Lake Formation

     This tutorial demonstrates how to set up permissions for open source transactional table formats (Apache Iceberg, Apache Hudi, and Linux Foundation Delta Lake tables) in Lake Formation.

   - Managing a data lake using Lake Formation tag-based access control

     Learn to manage access to the data within a data lake using tag-based access control in Lake Formation.

   - Securing data lakes with row-level access control

     Learn to set up row-level permissions that allow you to restrict access to specific rows based on data compliance and governance policies in Lake Formation.

3. **Sharing data:** Learn to securely share your data across AWS accounts using tag-based access control (TBAC) and manage granular permissions on datasets shared between AWS accounts.

   - Sharing a data lake using Lake Formation tag-based access control and named resources

In this tutorial, you learn how to securely share your data across AWS accounts using Lake Formation.

- Sharing a data lake using Lake Formation fine-grained access control

  In this tutorial, you learn how to quickly and easily share datasets using Lake Formation when managing multiple AWS accounts with AWS Organizations.

Topics

- Creating a data lake from an AWS CloudTrail source
- Creating a data lake from a JDBC source in Lake Formation
- Setting up permissions for open table storage formats in Lake Formation
- Managing a data lake using Lake Formation tag-based access control
- Securing data lakes with row-level access control
- Sharing a data lake using Lake Formation tag-based access control and named resources
- Sharing a data lake using Lake Formation fine-grained access control

Creating a data lake from an AWS CloudTrail source

This tutorial guides you through the actions to take on the Lake Formation console to create and load your first data lake from an AWS CloudTrail source.

High-level steps for creating a data lake

1. Register an Amazon Simple Storage Service (Amazon S3) path as a data lake.
2. Grant Lake Formation permissions to write to the Data Catalog and to Amazon S3 locations in the data lake.
3. Create a database to organize the metadata tables in the Data Catalog.
4. Use a blueprint to create a workflow. Run the workflow to ingest data from a data source.
5. Set up your Lake Formation permissions to allow others to manage data in the Data Catalog and the data lake.
6. Set up Amazon Athena to query the data that you imported into your Amazon S3 data lake.
7. For some data store types, set up Amazon Redshift Spectrum to query the data that you imported into your Amazon S3 data lake.

Topics

- Intended audience
- Prerequisites
- Step 1: Create a data analyst user
- Step 2: Add permissions to read AWS CloudTrail logs to the workflow role
- Step 3: Create an Amazon S3 bucket for the data lake
- Step 4: Register an Amazon S3 path
- Step 5: Grant data location permissions
- Step 6: Create a database in the Data Catalog
- Step 7: Grant data permissions
- Step 8: Use a blueprint to create a workflow
- Step 9: Run the workflow
- Step 10: Grant SELECT on the tables
- Step 11: Query the data lake Using Amazon Athena

Intended audience

The following table lists the roles used in this tutorial to create a data lake.

Intended audience

Role	Description
IAM Administrator	Has the AWS managed policy: `AdministratorAccess`. Can create IAM roles and Amazon S3 buckets.
Data lake administrator	User who can access the data catalog, create databases, and grant Lake Formation permissions to other users. Has fewer IAM permissions than the IAM administrator, but enough to administer the data lake.

Role	Description
Data analyst	User who can run queries against the data lake. Has only enough permissions to run queries.
Workflow role	Role with the required IAM policies to run a workflow. For more information, see (Optional) Create an IAM role for workflows.

Prerequisites

Before you begin:

- Ensure that you have completed the tasks in Set up AWS Lake Formation.
- Know the location of your CloudTrail logs.
- Athena requires the data analyst persona to create an Amazon S3 bucket to store query results before using Athena.

Familiarity with AWS Identity and Access Management (IAM) is assumed. For information about IAM, see the IAM User Guide.

Step 1: Create a data analyst user

This user has the minimum set of permissions to query the data lake.

1. Open the IAM console at https://console.aws.amazon.com/iam. Sign in as a user with the `AdministratorAccess` AWS managed policy.
2. Create a user named `datalake_user` with the following settings:

   - Enable AWS Management Console access.
   - Set a password and do not require password reset.
   - Attach the `AmazonAthenaFullAccess` AWS managed policy.
   - Attach the following inline policy. Name the policy `DatalakeUserBasic`.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess",
                "glue:GetTable",
                "glue:GetTables",
                "glue:SearchTables",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetPartitions",
                "lakeformation:GetResourceLFTags",
                "lakeformation:ListLFTags",
                "lakeformation:GetLFTag",
                "lakeformation:SearchTablesByLFTags",
                "lakeformation:SearchDatabasesByLFTags"
           ],
            "Resource": "*"
        }
    ]
}
```

Step 2: Add permissions to read AWS CloudTrail logs to the workflow role

1. Attach the following inline policy to the role `LakeFormationWorkflowRole`. The policy grants permission to read your AWS CloudTrail logs. Name the policy `DatalakeGetCloudTrail`.

   To create the `LakeFormationWorkflowRole` role, see (Optional) Create an IAM role for workflows.

   > **Important**
   >
   > Replace *`<your-s3-cloudtrail-bucket>`* with the Amazon S3 location of your CloudTrail data.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": ["arn:aws:s3:::<your-s3-cloudtrail-bucket>/*"]
        }
    ]
}
```

2. Verify that there are three policies attached to the role.

Step 3: Create an Amazon S3 bucket for the data lake

Create the Amazon S3 bucket that is to be the root location of your data lake.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/ and sign in as a user with administrative access.
2. Choose **Create bucket**, and go through the wizard to create a bucket named *<yourName>*-`datalake-cloudtrail`, where *<yourName>* is your first initial and last name. For example: `jdoe-datalake-cloudtrail`.

   For detailed instructions on creating an Amazon S3 bucket, see Creating a bucket.

Step 4: Register an Amazon S3 path

Register an Amazon S3 path as the root location of your data lake.

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator.
2. In the navigation pane, under **Register and ingest**, choose **Data lake locations**.
3. Choose **Register location** and then **Browse**.

4. Select the *<yourName>*-datalake-cloudtrail bucket that you created previously, accept the default IAM role AWSServiceRoleForLakeFormationDataAccess, and then choose **Register location**.

   For more information about registering locations, see Adding an Amazon S3 location to your data lake.

Step 5: Grant data location permissions

Principals must have *data location permissions* on a data lake location to create Data Catalog tables or databases that point to that location. You must grant data location permissions to the IAM role for workflows so that the workflow can write to the data ingestion destination.

1. In the navigation pane, under **Permissions**, choose **Data locations**.
2. Choose **Grant**, and in the **Grant permissions** dialog box, make these selections:
   a. For **IAM user and roles**, choose LakeFormationWorkflowRole.
   b. For **Storage locations**, choose your *<yourName>*-datalake-cloudtrail bucket.
3. Choose **Grant**.

For more information about data location permissions, see Underlying data access control.

Step 6: Create a database in the Data Catalog

Metadata tables in the Lake Formation Data Catalog are stored within a database.

1. In the navigation pane, under **Data catalog**, choose **Databases**.
2. Choose **Create database**, and under **Database details**, enter the name lakeformation_cloudtrail.
3. Leave the other fields blank, and choose **Create database**.

Step 7: Grant data permissions

You must grant permissions to create metadata tables in the Data Catalog. Because the workflow will run with the role LakeFormationWorkflowRole, you must grant these permissions to the role.

1. In the Lake Formation console, in the navigation pane, under **Data catalog**, choose **Databases**.
2. Choose the `lakeformation_cloudtrail` database, then, from the **Actions** drop-down list, choose **Grant** under the heading Permissions.
3. In the **Grant data permissions** dialog box, make these selections:
   a. Under **Principals**, for **IAM user and roles**, choose `LakeFormationWorkflowRole`.
   b. Under **LF-Tags or catalog resources**, choose **Named Data Catalog resources**.
   c. For **Databases**, you should see that the `lakeformation_cloudtrail` database is already added.
   d. Under **Database permissions**, select **Create table**, **Alter**, and **Drop**, and clear **Super** if it is selected.
4. Choose **Grant**.

For more information about granting Lake Formation permissions, see Managing Lake Formation permissions.

Step 8: Use a blueprint to create a workflow

In order to read the CloudTrail logs, understand their structure, create the appropriate tables in the Data Catalog, we need to set up a workflow that consists of a AWS Glue crawlers, jobs, triggers, and workflows. Lake Formation's blueprints simplifies this process.

The workflow generates the jobs, crawlers, and triggers that discover and ingest data into your data lake. You create a workflow based on one of the predefined Lake Formation blueprints.

1. In the Lake Formation console, in the navigation pane, choose **Blueprints** under **Ingestion**, and then choose **Use blueprint**.
2. On the **Use a blueprint** page, under **Blueprint type**, choose **AWS CloudTrail**.
3. Under **Import source**, choose a CloudTrail source and start date.
4. Under **Import target**, specify these parameters:

Target database	`lakeformation_cloudtrail`
Target storage location	`s3://<yourName> -datalake-cloudtrail`

Data format	Parquet

5. For import frequency, choose **Run on demand**.
6. Under **Import options**, specify these parameters:

Workflow name	`lakeformationcloudtrailtest`
IAM role	`LakeFormationWorkflowRole`
Table prefix	`cloudtrailtest` **Note** Must be lower case.

7. Choose **Create**, and wait for the console to report that the workflow was successfully created.

> **Tip**
>
> Did you get the following error message?
> `User: arn:aws:iam::<account-id>:user/<datalake_administrator_user> is not authorized to perform: iam:PassRole on resource:arn:aws:iam::<account-id>:role/LakeFormationWorkflowRole...`
> If so, check that you replaced *<account-id>* in the inline policy for the data lake administrator user with a valid AWS account number.

Step 9: Run the workflow

Because you specified that the workflow is run-on-demand, you must manually start the workflow.

- On the **Blueprints** page, select the workflow `lakeformationcloudtrailtest`, and on the **Actions** menu, choose **Start**.

  As the workflow runs, you can view its progress in the **Last run status** column. Choose the refresh button occasionally.

The status goes from **RUNNING**, to **Discovering**, to **Importing**, to **COMPLETED**.

When the workflow completes:

- The Data Catalog will have new metadata tables.
- Your CloudTrail logs will be ingested into the data lake.

If the workflow fails, do the following:

a. Select the workflow, and on the **Actions** menu, choose **View graph**.

   The workflow opens in the AWS Glue console.

b. Ensure that the workflow is selected, and choose the **History** tab.

c. Under **History**, select the most recent run and choose **View run details**.

d. Select a failed job or crawler in the dynamic (runtime) graph, and review the error message. Failed nodes are either red or yellow.

Step 10: Grant SELECT on the tables

You must grant the SELECT permission on the new Data Catalog tables so that the data analyst can query the data that the tables point to.

> **Note**
>
> A workflow automatically grants the SELECT permission on the tables that it creates to the user who ran it. Because the data lake administrator ran this workflow, you must grant SELECT to the data analyst.

1. In the Lake Formation console, in the navigation pane, under **Data catalog**, choose **Databases**.
2. Choose the `lakeformation_cloudtrail` database, then, from the **Actions** drop-down list, choose **Grant** under the heading Permissions.
3. In the **Grant data permissions** dialog box, make these selections:

   a. Under **Principals**, for **IAM user and roles**, choose `datalake_user`.

   b. Under **LF-Tags or catalog resources**, choose **Named data catalog resources**.

c. For **Databases**, the `lakeformation_cloudtrail` database should already be selected.

d. For **Tables**, choose `cloudtrailtest-cloudtrail`.

e. Under **Table and column permissions**, choose **Select**.

4. Choose **Grant**.

The next step is performed as the data analyst.

Step 11: Query the data lake Using Amazon Athena

Use the Amazon Athena console to query the CloudTrail data in your data lake.

1. Open the Athena console at https://console.aws.amazon.com/athena/ and sign in as the data analyst, user `datalake_user`.
2. If necessary, choose **Get Started** to continue to the Athena query editor.
3. For **Data source**, choose **AwsDataCatalog**.
4. For **Database**, choose `lakeformation_cloudtrail`.

   The **Tables** list populates.
5. On the overflow menu (3 dots arranged horizontally) beside the table `cloudtrailtest-cloudtrail`, choose **Preview table**, then choose **Run**.

   The query runs and displays 10 rows of data.

   If you have not used Athena before, you must first configure an Amazon S3 location in the Athena console for storing the query results. The `datalake_user` must have the necessary permissions to access the Amazon S3 bucket that you choose.

> **Note**
>
> Now that you have completed the tutorial, grant data permissions and data location permissions to the principals in your organization.

Creating a data lake from a JDBC source in Lake Formation

This tutorial guides you through the steps to take on the AWS Lake Formation console to create and load your first data lake from a JDBC source using Lake Formation.

Topics

- Intended audience
- JDBC tutorial prerequisites
- Step 1: Create a data analyst user
- Step 2: Create a connection in AWS Glue
- Step 3: Create an Amazon S3 bucket for the data lake
- Step 4: Register an Amazon S3 path
- Step 5: Grant data location permissions
- Step 6: Create a database in the Data Catalog
- Step 7: Grant data permissions
- Step 8: Use a blueprint to create a workflow
- Step 9: Run the workflow
- Step 10: Grant SELECT on the tables
- Step 11: Query the data lake using Amazon Athena
- Step 12: Query the data in the data lake using Amazon Redshift Spectrum
- Step 13: Grant or revoke Lake Formation permissions using Amazon Redshift Spectrum

Intended audience

The following table lists the roles that are used in this AWS Lake Formation JDBC tutorial.

Role	Description
IAM administrator	A user who can create AWS Identity and Access Management (IAM) users and roles and Amazon Simple Storage Service (Amazon S3) buckets. Has the `AdministratorAccess` AWS managed policy.

Role	Description
Data lake administrator	A user who can access the Data Catalog, create databases, and grant Lake Formation permissions to other users. Has fewer IAM permissions than the IAM administrator, but enough to administer the data lake.
Data analyst	A user who can run queries against the data lake. Has only enough permissions to run queries.
Workflow role	A role with the required IAM policies to run a workflow.

For information about prerequisites for completing the tutorial, see JDBC tutorial prerequisites.

JDBC tutorial prerequisites

Before you begin the AWS Lake Formation JDBC tutorial, ensure that you've done the following:

- Complete the tasks in Getting started with Lake Formation.
- Decide on a JDBC-accessible data store that you want to use for the tutorial.
- Gather the information that is required to create an AWS Glue connection of type JDBC. This Data Catalog object includes the URL to the data store, login credentials, and if the data store was created in an Amazon Virtual Private Cloud (Amazon VPC), additional VPC-specific configuration information. For more information, see Defining Connections in the AWS Glue Data Catalog in the *AWS Glue Developer Guide*.

The tutorial assumes that you are familiar with AWS Identity and Access Management (IAM). For information about IAM, see the IAM User Guide.

To get started, proceed to the section called “Step 1: Create a data analyst user”.

Step 1: Create a data analyst user

In this step, you create an AWS Identity and Access Management (IAM) user to be the data analyst for your data lake in AWS Lake Formation.

This user has the minimum set of permissions to query the data lake.

1. Open the IAM console at https://console.aws.amazon.com/iam. Sign in as a user with the `AdministratorAccess` AWS managed policy.
2. Create a user named `datalake_user` with the following settings:
   - Enable AWS Management Console access.
   - Set a password and do not require password reset.
   - Attach the `AmazonAthenaFullAccess` AWS managed policy.
   - Attach the following inline policy. Name the policy `DatalakeUserBasic`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess",
                "glue:GetTable",
                "glue:GetTables",
                "glue:SearchTables",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetPartitions",
                "lakeformation:GetResourceLFTags",
                "lakeformation:ListLFTags",
                "lakeformation:GetLFTag",
                "lakeformation:SearchTablesByLFTags",
                "lakeformation:SearchDatabasesByLFTags"
           ],
            "Resource": "*"
        }
    ]
}
```

Step 2: Create a connection in AWS Glue

> **Note**
>
> Skip this step if you already have an AWS Glue connection to your JDBC data source.

AWS Lake Formation accesses JDBC data sources through an AWS Glue *connection*. A connection is a Data Catalog object that contains all the information required to connect to the data source. You can create a connection using the AWS Glue console.

To create a connection

1. Open the AWS Glue console at https://console.aws.amazon.com/glue/, and sign in as a user with administrative access.
2. In the navigation pane, under **Data catalog**, choose **Connections**.
3. On the **Connectors** page, choose **Create connection**.
4. On the **Choose data source** page, choose **JDBC** as the connection type. Then choose **Next**.
5. Continue through the connection wizard and save the connection.

   For information on creating a connection, see AWS Glue JDBC connection properties in the *AWS Glue Developer Guide*.

Step 3: Create an Amazon S3 bucket for the data lake

In this step, you create the Amazon Simple Storage Service (Amazon S3) bucket that is to be the root location of your data lake.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/ and sign in as a user with administrative access.
2. Choose **Create bucket**, and go through the wizard to create a bucket named `<yourName>-datalake-tutorial`, where `<yourName>` is your first initial and last name. For example: `jdoe-datalake-tutorial`.

   For detailed instructions on creating an Amazon S3 bucket, see How Do I Create an S3 Bucket? in the *Amazon Simple Storage Service User Guide*.

Step 4: Register an Amazon S3 path

In this step, you register an Amazon Simple Storage Service (Amazon S3) path as the root location of your data lake.

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator.
2. In the navigation pane, under **Administration**, choose **Data lake locations**.
3. Choose **Register location**, and then choose **Browse**.
4. Select the *<yourName>*`-datalake-tutorial` bucket that you created previously, accept the default IAM role `AWSServiceRoleForLakeFormationDataAccess`, and then choose **Register location**.

   For more information about registering locations, see Adding an Amazon S3 location to your data lake.

Step 5: Grant data location permissions

Principals must have *data location permissions* on a data lake location to create Data Catalog tables or databases that point to that location. You must grant data location permissions to the IAM role for workflows so that the workflow can write to the data ingestion destination.

1. On the Lake Formation console, in the navigation pane, under **Permissions**, choose **Data locations**.
2. Choose **Grant**, and in the **Grant permissions** dialog box, do the following:
   a. For **IAM user and roles**, choose `LakeFormationWorkflowRole`.
   b. For **Storage locations**, choose your *<yourName>*`-datalake-tutorial` bucket.
3. Choose **Grant**.

For more information about data location permissions, see Underlying data access control.

Step 6: Create a database in the Data Catalog

Metadata tables in the Lake Formation Data Catalog are stored within a database.

1. On the Lake Formation console, in the navigation pane, under **Data catalog**, choose **Databases**.
2. Choose **Create database**, and under **Database details**, enter the name `lakeformation_tutorial`.
3. Leave the other fields blank, and choose **Create database**.

Step 7: Grant data permissions

You must grant permissions to create metadata tables in the Data Catalog. Because the workflow runs with the role `LakeFormationWorkflowRole`, you must grant these permissions to the role.

1. On the Lake Formation console, in the navigation pane, under **Permissions**, choose **Data lake permissions**.
2. Choose **Grant**, and in the **Grant data permissions** dialog box, do the following:
   a. Under **Principals**, for **IAM user and roles**, choose `LakeFormationWorkflowRole`.
   b. Under **LF-Tags or catalog resources**, choose **Named data catalog resources**.
   c. For **Databases**, choose the database that you created previously, `lakeformation_tutorial`.
   d. Under **Database permissions**, select **Create table**, **Alter**, and **Drop**, and clear **Super** if it is selected.
3. Choose **Grant**.

For more information about granting Lake Formation permissions, see Overview of Lake Formation permissions.

Step 8: Use a blueprint to create a workflow

The AWS Lake Formation workflow generates the AWS Glue jobs, crawlers, and triggers that discover and ingest data into your data lake. You create a workflow based on one of the predefined Lake Formation blueprints.

1. On the Lake Formation console, in the navigation pane, choose **Blueprints**, and then choose **Use blueprint**.
2. On the **Use a blueprint** page, under **Blueprint type**, choose **Database snapshot**.

3. Under **Import source**, for **Database connection**, choose the connection that you just created, `datalake-tutorial`, or choose an existing connection for your data source.
4. For **Source data path**, enter the path from which to ingest data, in the form *<database>*/*<schema>*/*<table>*.

   You can substitute the percent (%) wildcard for schema or table. For databases that support schemas, enter *<database>*/*<schema>*/% to match all tables in *<schema>* within *<database>*. Oracle Database and MySQL don't support schema in the path; instead, enter *<database>*/%. For Oracle Database, *<database>* is the system identifier (SID).

   For example, if an Oracle database has `orcl` as its SID, enter `orcl/%` to match all tables that the user specified in the JDCB connection has access to.

   > **Important**
   >
   > This field is case-sensitive.

5. Under **Import target**, specify these parameters:

Target database	`lakeformation_tutorial`
Target storage location	`s3://`*<yourName>* `-datalake-tutorial`
Data format	(Choose Parquet or CSV)

6. For import frequency, choose **Run on demand**.
7. Under **Import options**, specify these parameters:

Workflow name	`lakeformationjdbctest`
IAM role	`LakeFormationWorkflowRole`
Table prefix	`jdbctest`

> **Note**
>
> Must be lower case.

8. Choose **Create**, and wait for the console to report that the workflow was successfully created.

> **Tip**
>
> Did you get the following error message?
> `User: arn:aws:iam::<account-id>:user/<datalake_administrator_user> is not authorized to perform: iam:PassRole on resource:arn:aws:iam::<account-id>:role/LakeFormationWorkflowRole...`
> If so, check that you replaced *<account-id>* in the inline policy for the data lake administrator user with a valid AWS account number.

Step 9: Run the workflow

Because you specified that the workflow is run-on-demand, you must manually start the workflow in AWS Lake Formation.

1. On the Lake Formation console, on the **Blueprints** page, select the workflow `lakeformationjdbctest`.
2. Choose **Actions**, and then choose **Start**.
3. As the workflow runs, view its progress in the **Last run status** column. Choose the refresh button occasionally.

   The status goes from **RUNNING**, to **Discovering**, to **Importing**, to **COMPLETED**.

   When the workflow is complete:

   - The Data Catalog has new metadata tables.
   - Your data is ingested into the data lake.

   If the workflow fails, do the following:

a. Select the workflow. Choose **Actions**, and then choose **View graph**.

   The workflow opens in the AWS Glue console.

b. Select the workflow and choose the **History** tab.

c. Select the most recent run and choose **View run details**.

d. Select a failed job or crawler in the dynamic (runtime) graph, and review the error message. Failed nodes are either red or yellow.

Step 10: Grant SELECT on the tables

You must grant the SELECT permission on the new Data Catalog tables in AWS Lake Formation so that the data analyst can query the data that the tables point to.

> **Note**
>
> A workflow automatically grants the SELECT permission on the tables that it creates to the user who ran it. Because the data lake administrator ran this workflow, you must grant SELECT to the data analyst.

1. On the Lake Formation console, in the navigation pane, under **Permissions**, choose **Data lake permissions**.
2. Choose **Grant**, and in the **Grant data permissions** dialog box, do the following:

   a. Under **Principals**, for **IAM user and roles**, choose `datalake_user`.

   b. Under **LF-Tags or catalog resources**, choose **Named data catalog resources**.

   c. For **Databases**, choose `lakeformation_tutorial`.

      The **Tables** list populates.

   d. For **Tables**, choose one or more tables from your data source.

   e. Under **Table and column permissions**, choose **Select**.
3. Choose **Grant**.

The next step is performed as the data analyst.

Step 11: Query the data lake using Amazon Athena

Use the Amazon Athena console to query the data in your data lake.

1. Open the Athena console at https://console.aws.amazon.com/athena/, and sign in as the data analyst, user `datalake_user`.
2. If necessary, choose **Get Started** to continue to the Athena query editor.
3. For **Data source**, choose **AwsDataCatalog**.
4. For **Database**, choose `lakeformation_tutorial`.

   The **Tables** list populates.
5. In the pop-up menu beside one of the tables, choose **Preview table**.

   The query runs and displays 10 rows of data.

Step 12: Query the data in the data lake using Amazon Redshift Spectrum

You can set up Amazon Redshift Spectrum to query the data that you imported into your Amazon Simple Storage Service (Amazon S3) data lake. First, create an AWS Identity and Access Management (IAM) role that is used to launch the Amazon Redshift cluster and to query the Amazon S3 data. Then, grant this role the `Select` permissions on the tables that you want to query. Then, grant the user permissions to use the Amazon Redshift query editor. Finally, create an Amazon Redshift cluster and run queries.

You create the cluster as an administrator, and query the cluster as a data analyst.

For more information about Amazon Redshift Spectrum, see Using Amazon Redshift Spectrum to Query External Data in the *Amazon Redshift Database Developer Guide*.

To set up permissions to run Amazon Redshift queries

1. Open the IAM console at https://console.aws.amazon.com/iam/. Sign in as a user with the `AdministratorAccess` AWS managed policy.
2. In the navigation pane, choose **Policies**.

   If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.

3. Choose **Create policy**.
4. Choose the **JSON** tab.
5. Paste in the following JSON policy document.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess",
                "glue:GetTable",
                "glue:GetTables",
                "glue:SearchTables",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetPartitions",
                "lakeformation:GetResourceLFTags",
                "lakeformation:ListLFTags",
                "lakeformation:GetLFTag",
                "lakeformation:SearchTablesByLFTags",
                "lakeformation:SearchDatabasesByLFTags"
           ],
            "Resource": "*"
        }
    ]
}
```

6. When you are finished, choose **Review** to review the policy. The policy validator reports any syntax errors.
7. On the **Review policy** page, enter the **Name** as **`RedshiftLakeFormationPolicy`** for the policy that you are creating. Enter a **Description** (optional). Review the policy **Summary** to see the permissions that are granted by your policy. Then choose **Create policy** to save your work.
8. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
9. For **Select trusted entity**, choose **AWS service**.
10. Choose the Amazon Redshift service to assume this role.
11. Choose the **Redshift Customizable** use case for your service. Then choose **Next: Permissions**.
12. Search for the permissions policy that you created, `RedshiftLakeFormationPolicy`, and select the check box next to the policy name in the list.

13. Choose **Next: Tags**.
14. Choose **Next: Review**.
15. For **Role name**, enter the name **`RedshiftLakeFormationRole`**.
16. (Optional) For **Role description**, enter a description for the new role.
17. Review the role, and then choose **Create role**.

To grant `Select` permissions on the table to be queried in the Lake Formation database

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator.
2. In the navigation pane, under **Permissions**, choose **Data lake permissions**, and then choose **Grant**.
3. Provide the following information:
   - For **IAM users and roles**, choose the IAM role you created, `RedshiftLakeFormationRole`. When you run the Amazon Redshift Query Editor, it uses this IAM role for permission to the data.
   - For **Database**, choose `lakeformation_tutorial`.

     The tables list populates.
   - For **Table**, choose a table within the data source to query.
   - Choose the **Select** table permission.
4. Choose **Grant**.

To set up Amazon Redshift Spectrum and run queries

1. Open the Amazon Redshift console at https://console.aws.amazon.com/redshift. Sign in as the user `Administrator`.
2. Choose **Create cluster**.
3. On the **Create cluster** page, enter `redshift-lakeformation-demo` for the **Cluster identifier**.
4. For the **Node type**, select **dc2.large**.
5. Scroll down, and under **Database configurations**, enter or accept these parameters:
   - **Admin user name**: `awsuser`

- **Admin user password**: (*Choose a password*)

6. Expand **Cluster permissions**, and for **Available IAM roles**, choose **RedshiftLakeFormationRole**. Then choose **Add IAM role**.
7. If you must use a different port than the default value of 5439, next to **Additional configurations**, turn off the **Use defaults** option. Expand the section for **Database configurations**, and enter a new **Database port** number.
8. Choose **Create cluster**.

   The **Clusters** page loads.
9. Wait until the cluster status becomes **Available**. Choose the refresh icon periodically.
10. Grant the data analyst permission to run queries against the cluster. To do so, complete the following steps.

    a. Open the IAM console at https://console.aws.amazon.com/iam/, and sign in as the `Administrator` user.

    b. In the navigation pane, choose **Users**, and attach the following managed policies to the user `datalake_user`.

    - `AmazonRedshiftQueryEditor`
    - `AmazonRedshiftReadOnlyAccess`

11. Sign out of the Amazon Redshift console and sign back in as user `datalake_user`.
12. In the left vertical toolbar, choose the **EDITOR** icon to open the query editor and connect to the cluster. If the **Connect to database** dialog box appears, choose the cluster name `redshift-lakeformation-demo`, and enter the database name **dev**, the user name **awsuser**, and the password that you created. Then choose **Connect to database**.

    > **Note**
    >
    > If you are not prompted for connection parameters and another cluster is already selected in the query editor, choose **Change Connection** to open the **Connect to database** dialog box.

13. In the **New Query 1** text box, enter and run the following statement to map the database `lakeformation_tutorial` in Lake Formation to the Amazon Redshift schema name `redshift_jdbc`:

> **Important**
>
> Replace *<account-id>* with a valid AWS account number, and *<region>* with a valid AWS Region name (for example, us-east-1).

```
create external schema if not exists redshift_jdbc from DATA CATALOG
 database 'lakeformation_tutorial' iam_role 'arn:aws:iam::<account-id>:role/
RedshiftLakeFormationRole' region '<region>';
```

14. In the schema list under **Select schema**, choose **redshift_jdbc**.

    The tables list populates. The query editor shows only the tables on which you were granted Lake Formation data lake permissions.

15. On the pop-up menu next to a table name, choose **Preview data**.

    Amazon Redshift returns the first 10 rows.

    You can now run queries against the tables and columns for which you have permissions.

Step 13: Grant or revoke Lake Formation permissions using Amazon Redshift Spectrum

Amazon Redshift supports the ability to grant and revoke Lake Formation permissions on databases and tables using modified SQL statements. These statements are similar to the existing Amazon Redshift statements. For more information, see GRANT and REVOKE in the *Amazon Redshift Database Developer Guide*.

Setting up permissions for open table storage formats in Lake Formation

AWS Lake Formation supports managing access permissions for *Open Table Formats* (OTFs) such as Apache Iceberg, Apache Hudi, and Linux foundation Delta Lake. In this tutorial, you'll learn how to create Iceberg, Hudi, and Delta Lake with symlink manifest tables in the AWS Glue Data Catalog using AWS Glue, set up fine-grained permissions using Lake Formation, and query data using Amazon Athena.

> **Note**
>
> AWS analytics services don't support all transactional table formats. For more information, see Working with other AWS services. This tutorial manually covers creating a new database and a table in the Data Catalog using AWS Glue jobs only.

This tutorial includes an AWS CloudFormation template for quick setup. You can review and customize it to suit your needs.

Topics

- Intended audience
- Prerequisites
- Step 1: Provision your resources
- Step 2: Set up permissions for an Iceberg table
- Step 3: Set up permissions for a Hudi table
- Step 4: Set up permissions for a Delta Lake table
- Step 5: Clean up AWS resources

Intended audience

This tutorial is intended for IAM administrators, data lake administrators, and business analysts. The following table lists the roles used in this tutorial for creating a governed table using Lake Formation.

Role	Description
IAM Administrator	A user who can create IAM users and roles and Amazon S3 buckets. Has the `AdministratorAccess` AWS managed policy.
Data lake administrator	A user who can access the Data Catalog, create databases, and grant Lake Formation permissions to other users. Has fewer IAM

Role	Description
	permissions than the IAM administrator, but enough to administer the data lake.
Business analyst	A user who can run queries against the data lake. Has permissions to run queries.

Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in as a user with the correct permissions. For more information, see Sign up for an AWS account.

The tutorial assumes that you are familiar with IAM roles and policies. For information about IAM, see the IAM User Guide.

You need to set up the following AWS resources to complete this tutorial:

- Data lake administrator user
- Lake Formation data lake settings
- Amazon Athena engine version 3

To create a data lake administrator

1. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as an administrator user. You will create resources in the US East (N. Virginia) Region for this tutorial.
2. On the Lake Formation console, in the navigation pane, under **Permissions**, choose **Administrative roles and tasks**.
3. Select **Choose Administrators** under **Data lake administrators**.
4. In the pop-up window, **Manage data lake administrators**, under **IAM users and roles**, choose **IAM admin user**.
5. Choose **Save**.

To enable data lake settings

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/. In the navigation pane, under **Data catalog**, choose **Settings**. Uncheck the following:
   - Use only IAM access control for new databases.
   - Use only IAM access control for new tables in new databases.
2. Under **Cross account version settings**, choose **Version 3** as the cross account version.
3. Choose **Save**.

To upgrade Amazon Athena engine to version 3

1. Open Athena console at https://console.aws.amazon.com/athena/.
2. Select the **Workgroup** and select primary workgroup.
3. Ensure that the workgroup is at a minimum version of 3. If it is not, edit the workgroup, choose **Manual** for **Upgrade query engine**, and select version 3.
4. Choose **Save changes**.

Step 1: Provision your resources

This section shows you how to set up the AWS resources using an CloudFormation template.

To create your resources using CloudFormation template

1. Sign into the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation as an IAM administrator in the US East (N. Virginia) Region.
2. Choose Launch Stack.
3. Choose **Next** on the **Create stack** screen.
4. Enter a **Stack name**.
5. Choose **Next**.
6. On the next page, choose **Next**.
7. Review the details on the final page and select **I acknowledge that AWS CloudFormation might create IAM resources.**
8. Choose **Create**.

The stack creation can take up to two minutes.

Launching the cloud formation stack creates the following resources:

- lf-otf-datalake-123456789012 – Amazon S3 bucket to store data

> **Note**
>
> The account id appended to the Amazon S3 bucket name is replaced with your account id.

- lf-otf-tutorial-123456789012 – Amazon S3 bucket to store query results and AWS Glue job scripts
- lficebergdb – AWS Glue Iceberg database
- lfhudidb – AWS Glue Hudi database
- lfdeltadb – AWS Glue Delta database
- native-iceberg-create – AWS Glue job that creates an Iceberg table in the Data Catalog
- native-hudi-create – AWS Glue job that creates a Hudi table in the Data Catalog
- native-delta-create – AWS Glue job that creates a Delta table in the Data Catalog
- LF-OTF-GlueServiceRole – IAM role that you pass to AWS Glue to run the jobs. This role has the required policies attached to access the resources like Data Catalog, Amazon S3 bucket etc.
- LF-OTF-RegisterRole – IAM role to register the Amazon S3 location with Lake Formation. This role has `LF-Data-Lake-Storage-Policy` attached to the role.
- lf-consumer-analystuser – IAM user to query the data using Athena
- lf-consumer-analystuser-credentials – Password for the data analyst user stored in AWS Secrets Manager

After the stack creations is complete, navigate to the output tab and note down the values for:

- AthenaQueryResultLocation – Amazon S3 location for Athena query output
- BusinessAnalystUserCredentials – Password for the data analyst user

  To retrieve the password value:

1. Choose the `lf-consumer-analystuser-credentials` value by navigating to the Secrets Manager console.
2. In the **Secret value** section, choose **Retrieve secret value**.
3. Note down the secret value for the password.

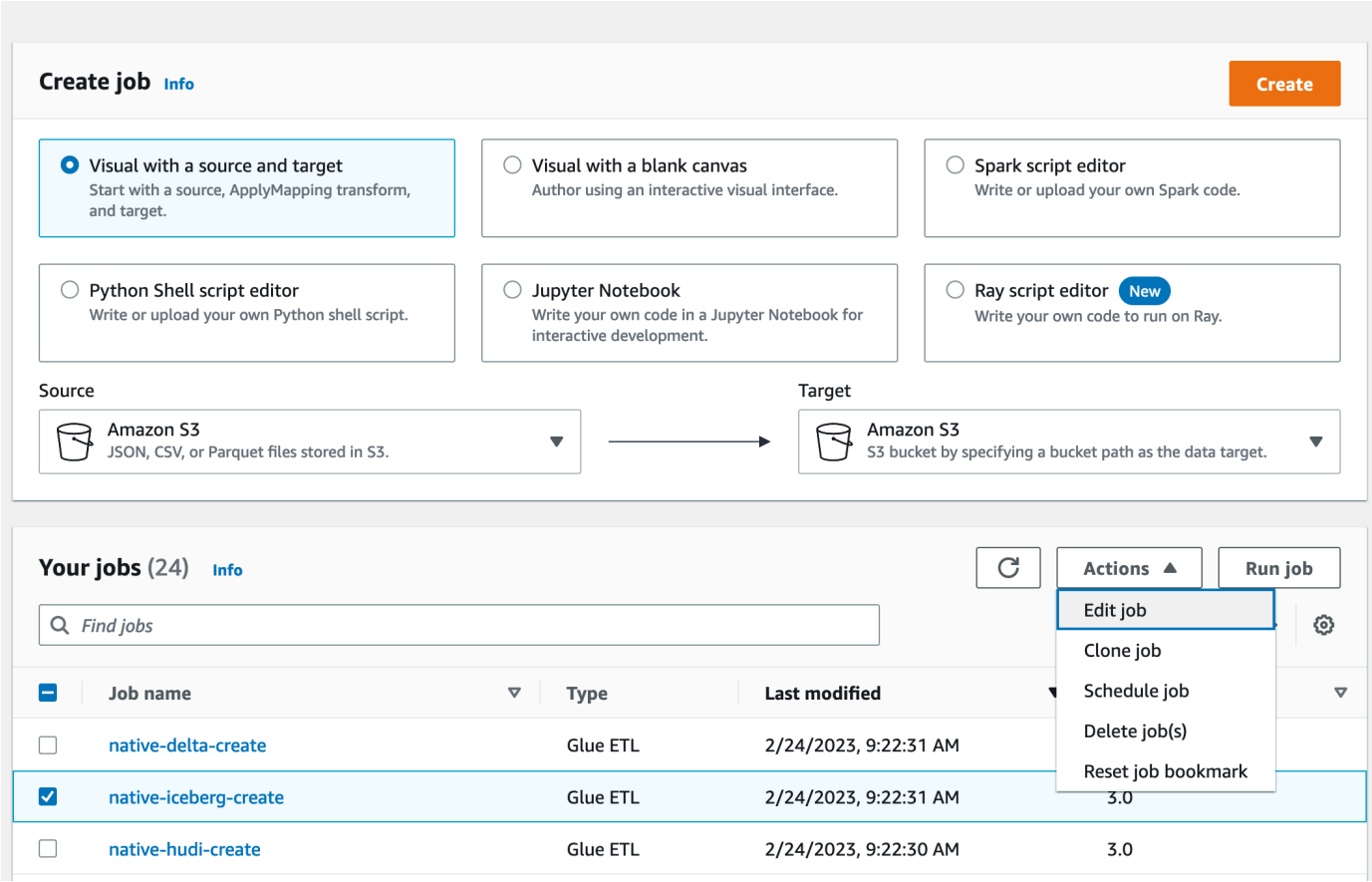
Step 2: Set up permissions for an Iceberg table

In this section, you'll learn how to create an Iceberg table in the AWS Glue Data Catalog, set up data permissions in AWS Lake Formation, and query data using Amazon Athena.

To create an Iceberg table

In this step, you'll run an AWS Glue job that creates an Iceberg transactional table in the Data Catalog.

1. Open the AWS Glue console at https://console.aws.amazon.com/glue/ in the US East (N. Virginia) Region as the data lake administrator user.
2. Choose **jobs** from the left navigation pane.
3. Select `native-iceberg-create`.



4. Under **Actions**, choose **Edit job**.
5. Under **Job details**, expand **Advanced properties**, and check the box next to **Use AWS Glue Data Catalog as the Hive metastore** to add the table metadata in the AWS Glue Data Catalog. This specifies AWS Glue Data Catalog as the metastore for the Data Catalog resources used in the job and enables Lake Formation permissions to be applied later on the catalog resources.
6. Choose **Save**.
7. Choose **Run**. You can view the status of the job while it is running.

   For more information on AWS Glue jobs, see Working with jobs on the AWS Glue console in the *AWS Glue Developer Guide*.

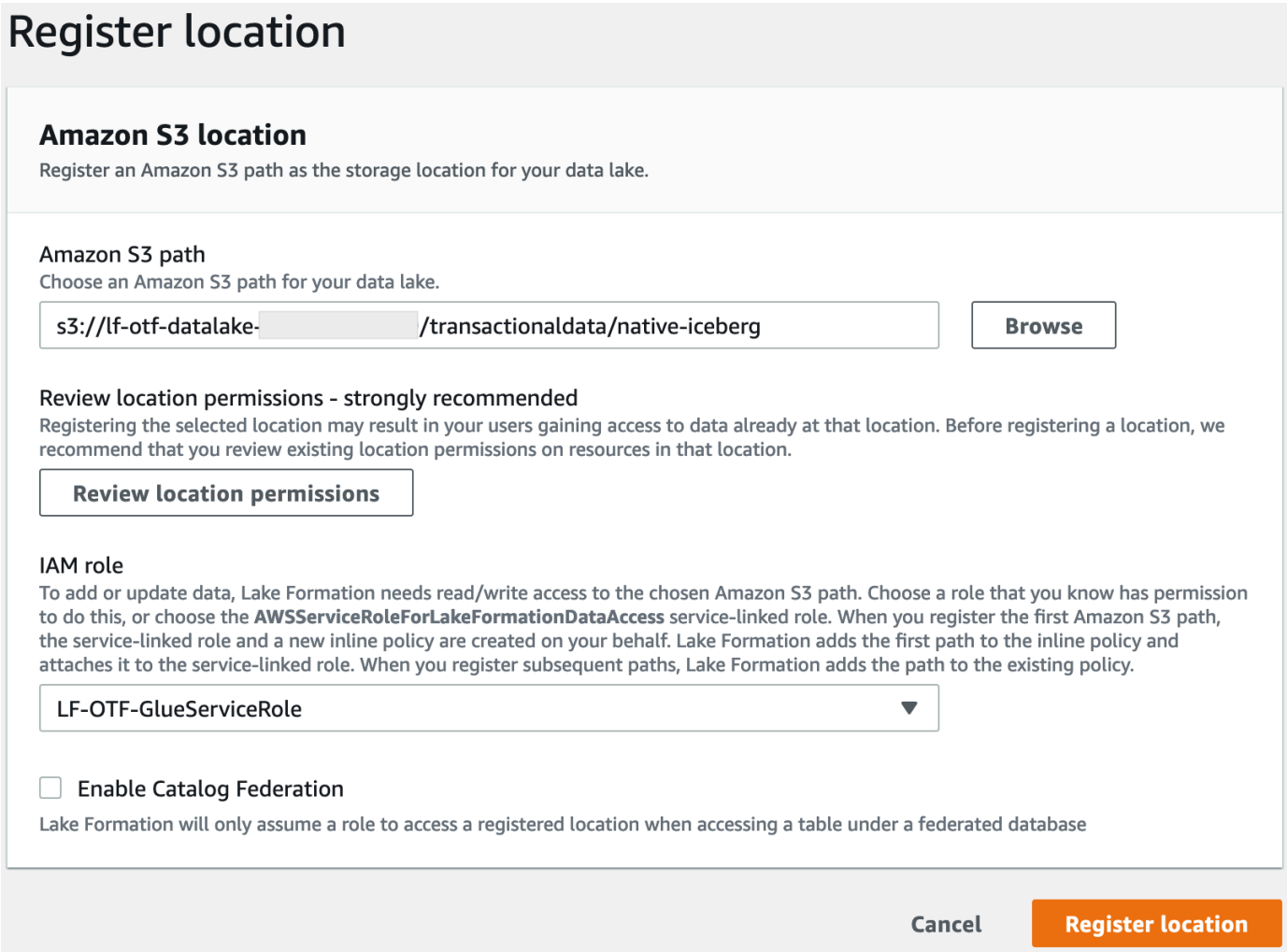
   This job creates an Iceberg table named `product` in the `lficebergdb` database. Verify the product table in the Lake Formation console.

To register the data location with Lake Formation

Next, register the Amazon S3 path as the location of your data lake.

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as the data lake administrator user.
2. In the navigation pane, under **Register and ingest**, choose **Data location**.
3. Choose **Register location**.
4. On the **Register location** page, enter the following:

   - **Amazon S3 path** – Choose **Browse** and select `lf-otf-datalake-123456789012`. Choose the expand arrow next to the Amazon S3 root location to navigate to the `s3/buckets/lf-otf-datalake-123456789012/transactionaldata/native-iceberg` location.
   - **IAM role** – Choose `LF-OTF-RegisterRole` as the IAM role.
   - Choose **Register location**.

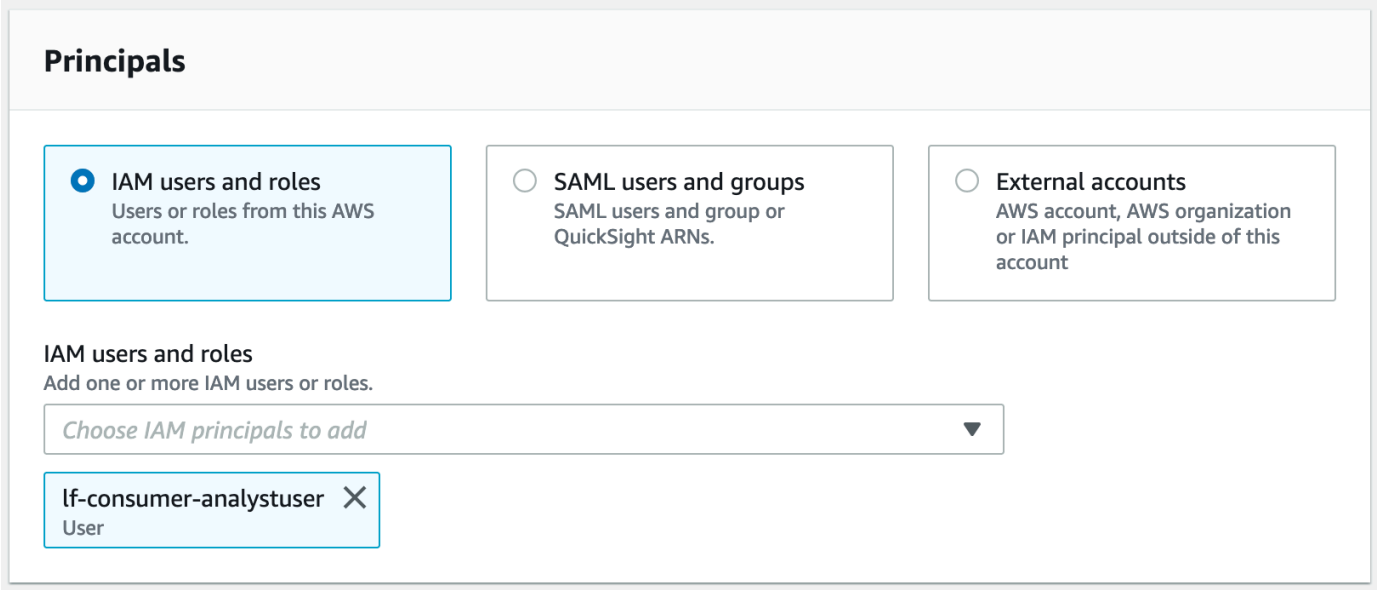


For more information on registering a data location with Lake Formation, see Adding an Amazon S3 location to your data lake.

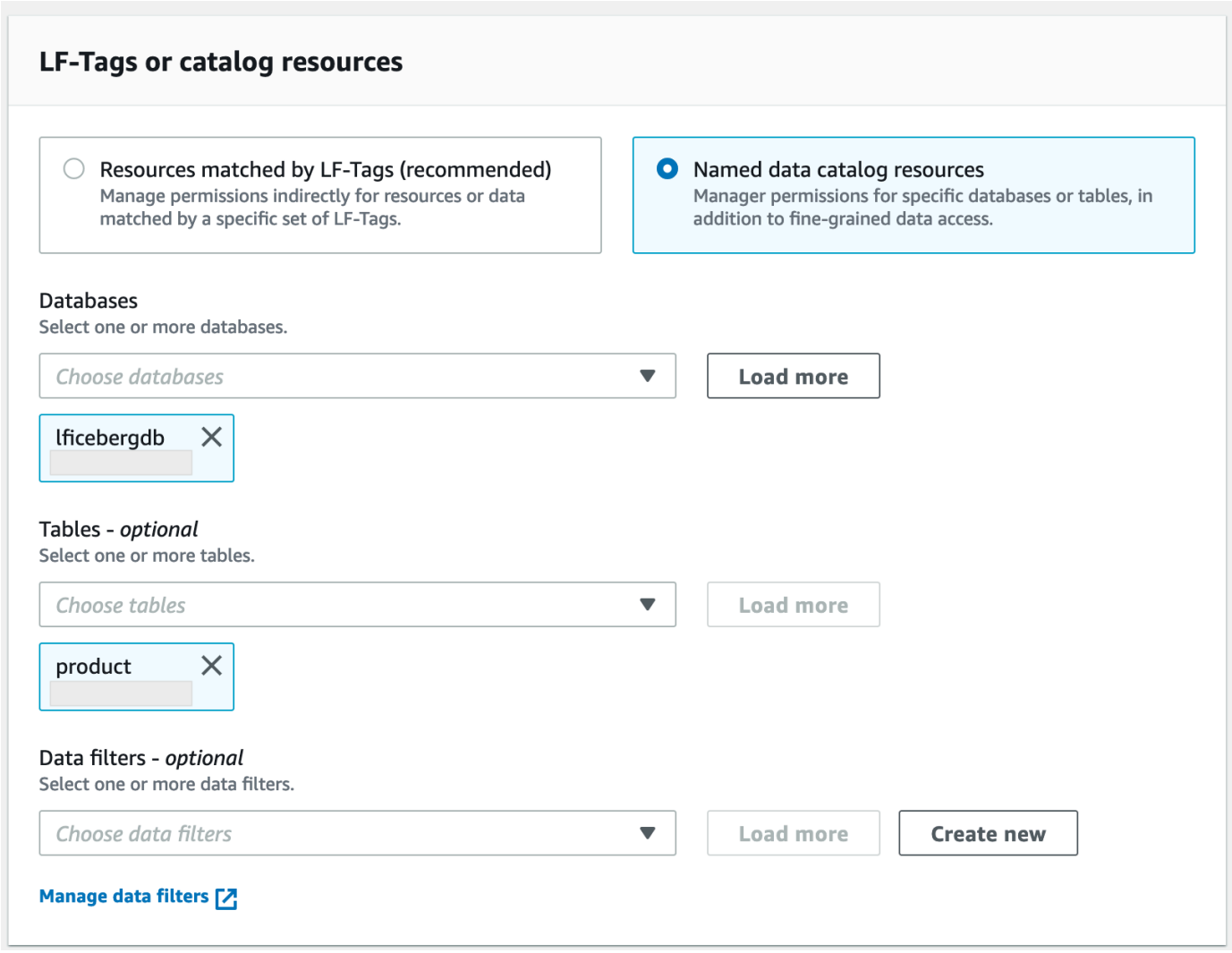
To grant Lake Formation permissions on the Iceberg table

In this step, we'll grant data lake permissions to the business analyst user.

1. Under **Data lake permissions**, choose **Grant**.
2. On the **Grant data permissions** screen, choose, **IAM users and roles**.
3. Choose `lf-consumer-analystuser` from the drop down.



4. Choose **Named data catalog resource**.
5. For **Databases** choose `lficebergdb`.
6. For **Tables**, choose `product`.



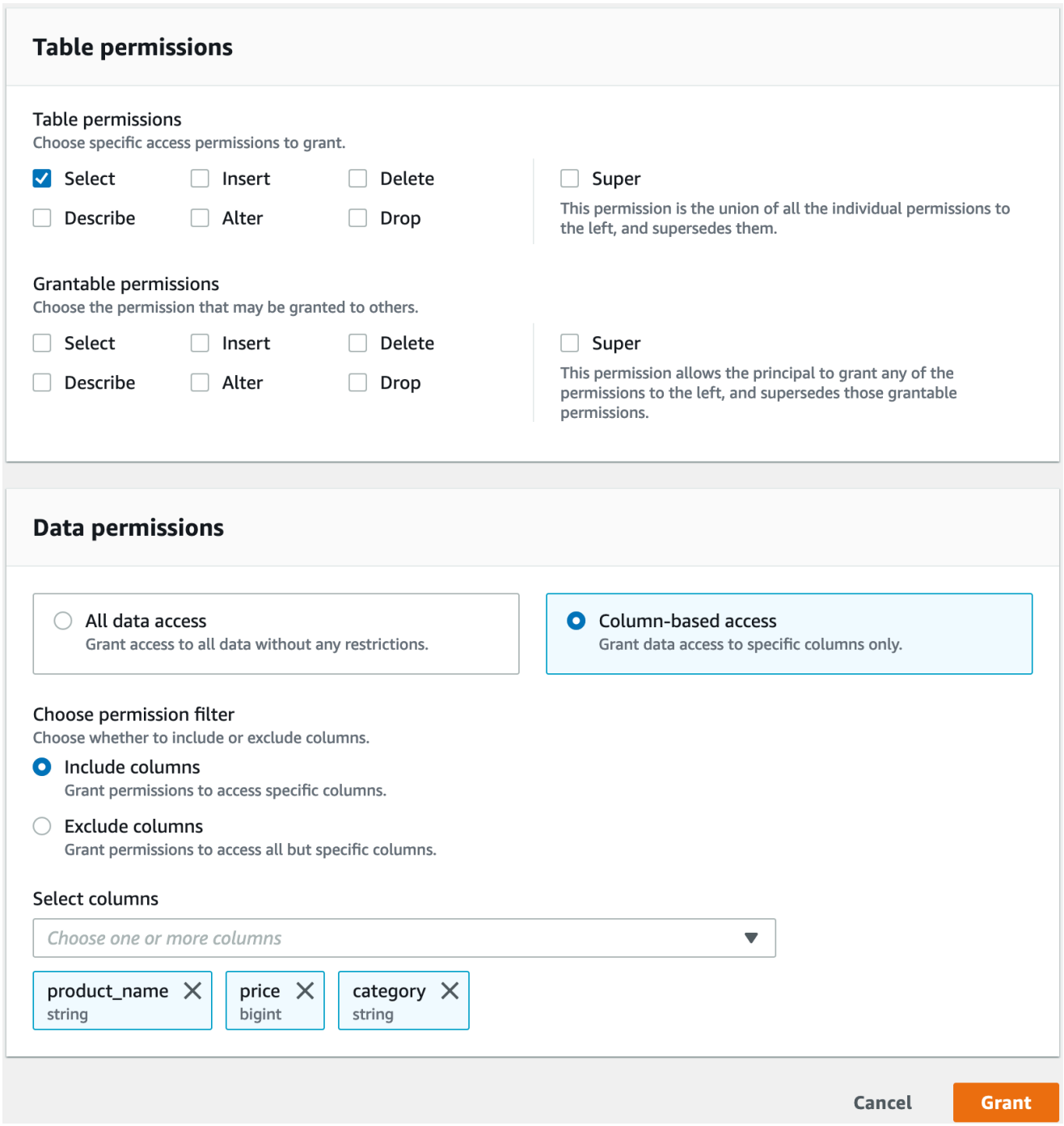
7. Next, you can grant column-based access by specifying columns.

   a. Under **Table permissions**, choose **Select**.

   b. Under **Data permissions**, choose **Column-based access**, choose **Include columns**.

   c. Choose `product_name`, `price`, and `category` columns.

   d. Choose **Grant**.



To query the Iceberg table using Athena

Now you can start querying the Iceberg table you created using Athena. If it is your first time running queries in Athena, you need to configure a query result location. For more information, see Specifying a query result location.

1. Sign out as the data lake administrator user and sign in as `lf-consumer-analystuser` in US East (N. Virginia) Region using the password noted earlier from the CloudFormation output.
2. Open the Athena console at https://console.aws.amazon.com/athena/.
3. Choose **Settings** and select **Manage**.
4. In the **Location of query result** box, enter the path to the bucket that you created in CloudFormation outputs. Copy the value of `AthenaQueryResultLocation` (s3://lf-otf-tutorial-123456789012/athena-results/) and choose **Save**.
5. Run the following query to preview 10 records stored in the Iceberg table:

   ```
   select * from lficebergdb.product limit 10;
   ```

   For more information on querying Iceberg tables using Athena, see Querying Iceberg tables in the *Amazon Athena User Guide*.

Step 3: Set up permissions for a Hudi table

In this section, you'll learn how to create a Hudi table in the AWS Glue Data Catalog, set up data permissions in AWS Lake Formation, and query data using Amazon Athena.

To create a Hudi table

In this step, you'll run an AWS Glue job that creates an Hudi transactional table in the Data Catalog.

1. Sign in to the AWS Glue console at https://console.aws.amazon.com/glue/ in the US East (N. Virginia) Region

   as the data lake administrator user.
2. Choose **jobs** from the left navigation pane.
3. Select `native-hudi-create`.
4. Under **Actions**, choose **Edit job**.
5. Under **Job details**, expand **Advanced properties**, and check the box next to **Use AWS Glue Data Catalog as the Hive metastore** to add the table metadata in the AWS Glue Data Catalog. This specifies AWS Glue Data Catalog as the metastore for the Data Catalog resources used in the job and enables Lake Formation permissions to be applied later on the catalog resources.
6. Choose **Save**.

7. Choose **Run**. You can view the status of the job while it is running.

   For more information on AWS Glue jobs, see Working with jobs on the AWS Glue console in the *AWS Glue Developer Guide*.

   This job creates a Hudi(cow) table in the database:lfhudidb. Verify the `product` table in the Lake Formation console.

To register the data location with Lake Formation

Next, register an Amazon S3 path as the root location of your data lake.

1. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as the data lake administrator user.
2. In the navigation pane, under **Register and ingest**, choose **Data location**.
3. Choose **Register location**.
4. On the **Register location** page, enter the following:
   - **Amazon S3 path** – Choose **Browse** and select `lf-otf-datalake-123456789012`. Choose the expand arrow next to the Amazon S3 root location to navigate to the `s3/buckets/lf-otf-datalake-123456789012/transactionaldata/native-hudi` location.
   - **IAM role** – Choose `LF-OTF-RegisterRole` as the IAM role.
   - Choose **Register location**.

To grant data lake permissions on the Hudi table

In this step, we'll grant data lake permissions to the business analyst user.

1. Under **Data lake permissions**, choose **Grant**.
2. On the **Grant data permissions** screen, choose, **IAM users and roles**.
3. `lf-consumer-analystuser` from the drop down.
4. Choose **Named data catalog resource**.
5. For **Databases** choose `lfhudidb`.
6. For **Tables**, choose `product`.
7. Next, you can grant column-based access by specifying columns.

a. Under **Table permissions**, choose **Select**.

b. Under **Data permissions**, choose **Column-based access**, choose **Include columns**.

c. Choose `product_name`, `price`, and `category` columns.

d. Choose **Grant**.

To query the Hudi table using Athena

Now start querying the Hudi table you created using Athena. If it is your first time running queries in Athena, you need to configure a query result location. For more information, see Specifying a query result location.

1. Sign out as the data lake administrator user and sign in as `lf-consumer-analystuser` in US East (N. Virginia) Region using the password noted earlier from the CloudFormation output.
2. Open the Athena console at https://console.aws.amazon.com/athena/.
3. Choose **Settings** and select **Manage**.
4. In the **Location of query result** box, enter the path to the bucket that you created in CloudFormation outputs. Copy the value of `AthenaQueryResultLocation` (s3://lf-otf-tutorial-123456789012/athena-results/) and **Save**.
5. Run the following query to preview 10 records stored in the Hudi table:

```
select * from lfhudidb.product limit 10;
```

For more information on querying Hudi tables, see the Querying Hudi tables section in the *Amazon Athena User Guide*.

Step 4: Set up permissions for a Delta Lake table

In this section, you'll learn how to create a Delta Lake table with symlink manifest file in the AWS Glue Data Catalog, set up data permissions in AWS Lake Formation and query data using Amazon Athena.

To create a Delta Lake table

In this step, you'll run an AWS Glue job that creates a Delta Lake transactional table in the Data Catalog.

1. Sign in to the AWS Glue console at https://console.aws.amazon.com/glue/ in the US East (N. Virginia) Region

   as the data lake administrator user.
2. Choose **jobs** from the left navigation pane.
3. Select `native-delta-create`.
4. Under **Actions**, choose **Edit job**.
5. Under **Job details**, expand **Advanced properties**, and check the box next to **Use AWS Glue Data Catalog as the Hive metastore** to add the table metadata in the AWS Glue Data Catalog. This specifies AWS Glue Data Catalog as the metastore for the Data Catalog resources used in the job and enables Lake Formation permissions to be applied later on the catalog resources.
6. Choose **Save**.
7. Choose **Run** under **Actions**.

   This job creates a Delta Lake table named `product` in the `lfdeltadb` database. Verify the `product` table in the Lake Formation console.

To register the data location with Lake Formation

Next, register the Amazon S3 path as the root location of your data lake.

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/ the data lake administrator user.
2. In the navigation pane, under **Register and ingest**, choose **Data location**.
3. Choose **Register location**.
4. On the **Register location** page, enter the following:

   - **Amazon S3 path** – Choose **Browse** and select `lf-otf-datalake-123456789012`. Choose the expand arrow next to the Amazon S3 root location to navigate to the `s3/buckets/lf-otf-datalake-123456789012/transactionaldata/native-delta` location.
   - **IAM role** – Choose `LF-OTF-RegisterRole` as the IAM role.
   - Choose **Register location**.

To grant data lake permissions on the Delta Lake table

In this step, we'll grant data lake permissions to the business analyst user.

1. Under **Data lake permissions**, choose **Grant**.
2. On the **Grant data permissions** screen, choose, **IAM users and roles**.
3. `lf-consumer-analystuser` from the drop down.
4. Choose **Named data catalog resource**.
5. For **Databases** choose `lfdeltadb`.
6. For **Tables**, choose `product`.
7. Next, you can grant column-based access by specifying columns.
   a. Under **Table permissions**, choose **Select**.
   b. Under **Data permissions**, choose **Column-based access**, choose **Include columns**.
   c. Choose `product_name`, `price`, and `category` columns.
   d. Choose **Grant**.

To query the Delta Lake table using Athena

Now start querying the Delta Lake table you created using Athena. If it is your first time running queries in Athena, you need to configure a query result location. For more information, see Specifying a query result location.

1. Log out as the data lake administrator user and login as `BusinessAnalystUser` in US East (N. Virginia) Region using the password noted earlier from the CloudFormation output.
2. Open the Athena console at https://console.aws.amazon.com/athena/.
3. Choose **Settings** and select **Manage**.
4. In the **Location of query result** box, enter the path to the bucket that you created in CloudFormation outputs. Copy the value of `AthenaQueryResultLocation` (s3://lf-otf-tutorial-123456789012/athena-results/) and **Save**.
5. Run the following query to preview 10 records stored in the Delta Lake table:

```
select * from lfdeltadb.product limit 10;
```

   For more information on querying Delta Lake tables, see the Querying Delta Lake tables section in the *Amazon Athena User Guide*.

Step 5: Clean up AWS resources

To clean up resources

To prevent unwanted charges to your AWS account, delete the AWS resources that you used for this tutorial.

1. Sign in to the CloudFormation console at https://console.aws.amazon.com/cloudformation as the IAM administrator.
2. Delete the cloud formation stack. The tables you created are automatically deleted with the stack.

Managing a data lake using Lake Formation tag-based access control

Thousands of customers are building petabyte-scale data lakes on AWS. Many of these customers use AWS Lake Formation to easily build and share their data lakes across the organization. As the number of tables and users increase, data stewards and administrators are looking for ways to manage permissions on data lakes easily at scale. Lake Formation Tag-based access control (LF-TBAC) solves this problem by allowing data stewards to create *LF-tags* (based on their data classification and ontology) that can then be attached to resources.

LF-TBAC is an authorization strategy that defines permissions based on attributes. In Lake Formation, these attributes are called LF-tags. You can attach LF-tags to Data Catalog resources and Lake Formation principals. Data lake administrators can assign and revoke permissions on Lake Formation resources using LF-tags. For more information about see, Lake Formation tag-based access control.

This tutorial demonstrates how to create a Lake Formation tag-based access control policy using an AWS public dataset. In addition, it shows how to query tables, databases, and columns that have Lake Formation tag-based access policies associated with them.

You can use LF-TBAC for the following use cases:

- You have a large number of tables and principals that the data lake administrator has to grant access
- You want to classify your data based on an ontology and grant permissions based on classification

- The data lake administrator wants to assign permissions dynamically, in a loosely coupled way

Following are the high-level steps for configuring permissions using LF-TBAC:

1. The data steward defines the tag ontology with two LF-tags: `Confidential` and `Sensitive`. Data with `Confidential=True` has tighter access controls. Data with `Sensitive=True` requires specific analysis from the analyst.
2. The data steward assigns different permission levels to the data engineer to build tables with different LF-tags.
3. The data engineer builds two databases: `tag_database` and `col_tag_database`. All tables in `tag_database` are configured with `Confidential=True`. All tables in the `col_tag_database` are configured with `Confidential=False`. Some columns of the table in `col_tag_database` are tagged with `Sensitive=True` for specific analysis needs.
4. The data engineer grants read permission to the analyst for tables with specific expression condition `Confidential=True` and `Confidential=False,Sensitive=True`.
5. With this configuration, the data analyst can focus on performing analysis with the right data.

Topics

- Intended audience
- Prerequisites
- Step 1: Provision your resources
- Step 2: Register your data location, create an LF-Tag ontology, and grant permissions
- Step 3: Create Lake Formation databases
- Step 4: Grant table permissions
- Step 5: Run a query in Amazon Athena to verify the permissions
- Step 6: Clean up AWS resources

Intended audience

This tutorial is intended for data stewards, data engineers, and data analysts. When it comes to managing AWS Glue Data Catalog and administering permission in Lake Formation, data stewards within the producing accounts have functional ownership based on the functions they support, and can grant access to various consumers, external organizations, and accounts.

The following table lists the roles that are used in this tutorial:

Role	Description
Data steward (administrator)	The `lf-data-steward` user has the following access: • Read access to all resources in the Data Catalog • Can create LF-tags and associate to the data engineer role for grantable permission to other principals
Data engineer	`lf-data-engineer` user has the following access: • Full read, write, and update access to all resources in the Data Catalog • Data location permissions in the data lake • Can associate LF-tags and associate to the Data Catalog • Can attach LF-tags to resources, which provides access to principals based on any policies created by data stewards
Data analyst	The `lf-data-analyst` user has the following access: • Fine-grained access to resources shared by Lake Formation tag-based access policies

Prerequisites

Before you start this tutorial, you must have an AWS account that you can use to sign in as an administrative user with correct permissions. For more information, see Complete initial AWS configuration tasks.

The tutorial assumes that you are familiar with IAM. For information about IAM, see the IAM User Guide.

Step 1: Provision your resources

This tutorial includes an AWS CloudFormation template for a quick setup. You can review and customize it to suit your needs. The template creates three different roles (listed in Intended audience) to perform this exercise and copies the nyc-taxi-data dataset to your local Amazon S3 bucket.

- An Amazon S3 bucket
- The appropriate Lake Formation settings
- The appropriate Amazon EC2 resources
- Three IAM roles with credentials

Create your resources

1. Sign into the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation in the US East (N. Virginia) region.
2. Choose Launch Stack.
3. Choose **Next**.
4. In the **User Configuration** section, enter password for three roles: `DataStewardUserPassword`, `DataEngineerUserPassword` and `DataAnalystUserPassword`.
5. Review the details on the final page and select **I acknowledge that AWS CloudFormation might create IAM resources**.
6. Choose **Create**.

   The stack creation can take up to five minutes.

> **Note**
>
> After you complete the tutorial, you might want to delete the stack in CloudFormation to avoid continuing to incur charges. Verify that the resources are successfully deleted in the event status for the stack.

Step 2: Register your data location, create an LF-Tag ontology, and grant permissions

In this step, the data steward user defines the tag ontology with two LF-Tags: `Confidential` and `Sensitive`, and gives specific IAM principals the ability to attach newly created LF-Tags to resources.

Register a data location and define LF-Tag ontology

1. Perform the first step as the data steward user (`lf-data-steward`) to verify the data in Amazon S3 and the Data Catalog in Lake Formation.

   a. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as `lf-data-steward` with the password used while deploying the CloudFormation stack.

   b. In the navigation pane, under **Permissions**, choose **Administrative roles and tasks**.

   c. Choose **Add** in the **Data lake administrators** section.

   d. On the **Add administrator** page, for **IAM users and roles**, choose the user `lf-data-steward`.

   e. Choose **Save** to add `lf-data-steward` as a Lake Formation administrator.

2. Next, update the Data Catalog settings to use Lake Formation permission to control catalog resources instead of IAM based access control.

   a. In the navigation pane, under **Administration**, choose **Data Catalog settings**.

   b. Uncheck **Use only IAM access control for new databases**.

   c. Uncheck **Use only IAM access control for new tables in new databases**.

   d. Click **Save**.

3. Next, register the data location for the data lake.

   a. In the navigation pane, under **Administration**, choose **Data lake locations**.

   b. Choose **Register location**.

   c. On the **Register location** page, for **Amazon S3 path**, enter `s3://lf-tagbased-demo-`*`Account-ID`*.

   d. For **IAM** role, leave the default value `AWSServiceRoleForLakeFormationDataAccess` as it is.

   e. Choose **Lake Formation** as the permission mode.

f. Choose **Register location**.

4. Next, create the ontology by defining an LF-tag.

   a. Under **Permissions** in the navigation pane, choose **LF-Tags and permissions.**.

   b. Choose **Add LF-Tag**.

   c. For **Key**, enter `Confidential`.

   d. For **Values**, add `True` and `False`.

   e. Choose **Add LF-tag**.

   f. Repeat the steps to create the **LF-Tag** `Sensitive` with the value `True`.

   You have created all the necessary LF-Tags for this exercise.

Grant permissions to IAM users

1. Next, give specific IAM principals the ability to attach newly created LF-tags to resources.

   a. Under **Permissions** in the navigation pane, choose **LF-Tags and permissions**.

   b. In the **LF-Tag permissions** section, choose **Grant permissions**.

   c. For **Permission type**, choose **LF-Tag key-value pair permissions**.

   d. Select **IAM users and roles**.

   e. For **IAM users and roles**, search for and choose the `lf-data-engineer` role.

   f. In the **LF-Tags** section, add the key `Confidential` with values `True` and `False`, and the key `Sensitive` with value `True`.

   g. Under **Permissions**, select **Describe** and **Associate** for **Permissions** and **Grantable permissions**.

   h. Choose **Grant**.

2. Next, grant permissions to `lf-data-engineer` to create databases in our Data Catalog and on the underlying Amazon S3 bucket created by AWS CloudFormation.

   a. Under **Administration** in the navigation pane, choose **Administrative roles and tasks**.

   b. In the **Database creators** section, choose **Grant**.

   c. For **IAM users and roles**, choose the `lf-data-engineer` role.

   d. For **Catalog permissions**, select **Create database**.

e. Choose **Grant**.

3. Next, grant permissions on the Amazon S3 bucket (`s3://lf-tagbased-demo-Account-ID`) to the `lf-data-engineer` user.

   a. In the navigation pane, under **Permissions**, choose **Data locations**.

   b. Choose **Grant**.

   c. Select **My account**.

   d. For **IAM users and roles**, choose the `lf-data-engineer` role.

   e. For **Storage locations**, enter the Amazon S3 bucket created by the CloudFormation template (`s3://lf-tagbased-demo-Account-ID`).

   f. Choose **Grant**.

4. Next, grant `lf-data-engineer` grantable permissions on resources associated with the **LF-Tag** expression `Confidential=True`.

   a. In the navigation pane, under **Permissions**, choose **Data lake permissions**.

   b. Choose **Grant**.

   c. Select **IAM users and roles**.

   d. Choose the role `lf-data-engineer`.

   e. In the **LF-Tags or catalog resources** section, select **Resources matched by LF-Tags**.

   f. Choose **Add LF-Tag key-value pair**.

   g. Add the key `Confidential` with the values `True`.

   h. In the **Database permissions** section, select **Describe** for **Database permissions** and **Grantable permissions**.

   i. In the **Table permissions** section, select **Describe**, **Select**, and **Alter** for both **Table permissions** and **Grantable permissions**.

   j. Choose **Grant**.

5. Next, grant `lf-data-engineer` grantable permissions on resources associated with the LF-Tag expression `Confidential=False`.

   a. In the navigation pane, under **Permissions**, choose **Data lake permissions**.

   b. Choose **Grant**.

   c. Select **IAM users and roles**.

   d. Choose the role `lf-data-engineer`.

e. Select **Resources matched by LF-tags**.

f. Choose **Add LF-tag**.

g. Add the key `Confidential` with the value `False`.

h. In the **Database permissions** section, select **Describe** for **Database permissions** and **Grantable permissions**.

i. In the **Table and column permissions** section, do not select anything.

j. Choose **Grant**.

6. Next, we grant `lf-data-engineer` grantable permissions on resources associated with the **LF-Tag** key-value pairs `Confidential=False` and `Sensitive=True`.

a. In the navigation pane, under **Permissions**, choose **Data permissions**.

b. Choose **Grant**.

c. Select **IAM users and roles**.

d. Choose the role `lf-data-engineer`.

e. Under **LF-Tags or catalog resources** section, select **Resources matched by LF-Tags**.

f. Choose **Add LF-Tag**.

g. Add the key `Confidential` with the value `False`.

h. Choose **Add LF-Tag key-value pair**.

i. Add the key `Sensitive` with the value `True`.

j. In the **Database permissions** section, select **Describe** for **Database permissions** and **Grantable permissions**.

k. In the **Table permissions** section, select **Describe**, **Select**, and **Alter** for both **Table permissions** and **Grantable permissions**.

l. Choose **Grant**.

Step 3: Create Lake Formation databases

In this step, you create two databases and attach LF-Tags to the databases and specific columns for testing purposes.

Create your databases and table for database-level access

1. First, create the database `tag_database`, the table `source_data`, and attach appropriate LF-Tags.

a. On the Lake Formation console (https://console.aws.amazon.com/lakeformation/), under **Data Catalog**, choose **Databases**.

b. Choose **Create database**.

c. For **Name**, enter `tag_database`.

d. For **Location**, enter the Amazon S3 location created by the CloudFormation template (`s3://lf-tagbased-demo-`*`Account-ID`*`/tag_database/`).

e. Deselect **Use only IAM access control for new tables in this database**.

f. Choose **Create database**.

2. Next, create a new table within `tag_database`.

a. On the **Databases** page, select the database `tag_database`.

b. Choose**View Tables** and click **Create table**.

c. For **Name**, enter `source_data`.

d. For **Database**, choose the database `tag_database`.

e. For **Table format**, choose **Standard AWS Glue table**.

f. For **Data is located in**, select **Specified path in my account**.

g. For Include path, enter the path to `tag_database` created by the CloudFormation template (`s3://lf-tagbased-demo`*`Account-ID`*`/tag_database/`).

h. For **Data format**, select **CSV**.

i. Under **Upload schema**, enter the following JSON array of column structure to create a schema:

```
[
                {
                     "Name": "vendorid",
                     "Type": "string"
                },
                {
                     "Name": "lpep_pickup_datetime",
                     "Type": "string"
                },
                {
                     "Name": "lpep_dropoff_datetime",
                     "Type": "string"

                },
```

```
                {
                  "Name": "store_and_fwd_flag",
                  "Type": "string"
                },
                {
                  "Name": "ratecodeid",
                  "Type": "string"

                },
                {
                  "Name": "pulocationid",
                  "Type": "string"

                },
                {
                  "Name": "dolocationid",
                  "Type": "string"

                },
                {
                  "Name": "passenger_count",
                  "Type": "string"

                },
                {
                  "Name": "trip_distance",
                  "Type": "string"

                },
                {
                  "Name": "fare_amount",
                  "Type": "string"

                },
                {
                  "Name": "extra",
                  "Type": "string"

                },
                {
                  "Name": "mta_tax",
                  "Type": "string"

                },
```

```
                {
                     "Name": "tip_amount",
                     "Type": "string"

                },
                   {
                     "Name": "tolls_amount",
                     "Type": "string"

                },
                {
                     "Name": "ehail_fee",
                     "Type": "string"

                },
                {
                     "Name": "improvement_surcharge",
                     "Type": "string"

                },
                {
                     "Name": "total_amount",
                     "Type": "string"

                },
                {
                     "Name": "payment_type",
                     "Type": "string"

                }
]
```

j. Choose **Upload**. After uploading the schema, the table schema should look like the following screenshot:

#	Column Name	Data type
1	vendorid	string
2	lpep_pickup_datetime	string
3	lpep_dropoff_datetime	string
4	store_and_fwd_flag	string
5	ratecodeid	string
6	pulocationid	string
7	dolocationid	string
8	passenger_count	string
9	trip_distance	string
10	fare_amount	string
11	extra	string
12	mta_tax	string
13	tip_amount	string
14	tolls_amount	string
15	ehail_fee	string
16	improvement_surcharge	string
17	total_amount	string
18	payment_type	string

k. Choose **Submit**.

3. Next, attach LF-Tags at the database level.

   a. On the **Databases** page, find and select `tag_database`.

   b. On the **Actions** menu, choose **Edit LF-Tags**.

   c. Choose **Assign new LF-tag**.

   d. For **Assigned keys**, choose the `Confidential` LF-Tag you created earlier.

   e. For **Values**, choose `True`.

   f. Choose **Save**.

   This completes the LF-Tag assignment to the tag_database database.

Create your database and table for column-level access

Repeat the following steps to create the database `col_tag_database` and table `source_data_col_lvl`, and attach LF-Tags at the column level.

1. On the **Databases** page, choose **Create database**.
2. For **Name**, enter `col_tag_database`.
3. For **Location**, enter the Amazon S3 location created by the CloudFormation template (`s3://lf-tagbased-demo-`*`Account-ID`*`/col_tag_database/`).
4. Deselect **Use only IAM access control for new tables in this database**.
5. Choose **Create database**.
6. On the **Databases** page, select your new database (`col_tag_database`).
7. Choose **View tables** and click **Create table**.
8. For **Name**, enter `source_data_col_lvl`.
9. For **Database**, choose your new database (`col_tag_database`).
10. For **Table format**, choose **Standard AWS Glue table**.
11. For **Data is located in**, select **Specified path in my account**.
12. Enter the Amazon S3 path for `col_tag_database` (`s3://lf-tagbased-demo-`*`Account-ID`*`/col_tag_database/`).
13. For **Data format**, select CSV.
14. Under `Upload schema`, enter the following schema JSON:

```
[
                {
                     "Name": "vendorid",
                     "Type": "string"


                },
                {
                     "Name": "lpep_pickup_datetime",
                     "Type": "string"


                },
                {
                     "Name": "lpep_dropoff_datetime",
                     "Type": "string"


                },
                   {
                     "Name": "store_and_fwd_flag",
                     "Type": "string"


                },
                   {
                     "Name": "ratecodeid",
                     "Type": "string"


                },
                   {
                     "Name": "pulocationid",
                     "Type": "string"


                },
                {
                     "Name": "dolocationid",
                     "Type": "string"


                },
```

```
                {
                  "Name": "passenger_count",
                  "Type": "string"


              },
              {
                  "Name": "trip_distance",
                  "Type": "string"


              },
                {
                  "Name": "fare_amount",
                  "Type": "string"


              },
              {
                  "Name": "extra",
                  "Type": "string"


              },
                {
                  "Name": "mta_tax",
                  "Type": "string"


              },
              {
                  "Name": "tip_amount",
                  "Type": "string"


              },
                {
                  "Name": "tolls_amount",
                  "Type": "string"


              },
              {
                  "Name": "ehail_fee",
```

```
                    "Type": "string"


               },
               {
                    "Name": "improvement_surcharge",
                    "Type": "string"


               },
               {
                    "Name": "total_amount",
                    "Type": "string"


               },
               {
                    "Name": "payment_type",
                    "Type": "string"


               }
]
```

15. Choose `Upload`. After uploading the schema, the table schema should look like the following screenshot.

#	Column Name	Data type
1	vendorid	string
2	lpep_pickup_datetime	string
3	lpep_dropoff_datetime	string
4	store_and_fwd_flag	string
5	ratecodeid	string
6	pulocationid	string
7	dolocationid	string
8	passenger_count	string
9	trip_distance	string
10	fare_amount	string
11	extra	string
12	mta_tax	string
13	tip_amount	string
14	tolls_amount	string
15	ehail_fee	string
16	improvement_surcharge	string
17	total_amount	string
18	payment_type	string

16. Choose **Submit** to complete the creation of the table.
17. Now, associate the `Sensitive=True` LF-Tag to the columns `vendorid` and `fare_amount`.
    a. On the **Tables** page, select the table you created (`source_data_col_lvl`).
    b. On the **Actions** menu, choose **Schema**.
    c. Select the column `vendorid` and choose **Edit LF-Tags**.
    d. For **Assigned keys**, choose **Sensitive**.
    e. For **Values**, choose **True**.
    f. Choose **Save**.
18. Next, associate the `Confidential=False` LF-Tag to `col_tag_database`. This is required for `lf-data-analyst` to be able to describe the database `col_tag_database` when logged in from Amazon Athena.
    a. On the **Databases** page, find and select `col_tag_database`.
    b. On the **Actions** menu, choose **Edit LF-Tags**.
    c. Choose **Assign new LF-Tag**.
    d. For **Assigned keys**, choose the `Confidential` LF-Tag you created earlier.
    e. For **Values**, choose `False`.
    f. Choose **Save**.

Step 4: Grant table permissions

Grant permissions to data analysts for consumption of the databases `tag_database` and the table `col_tag_database` using LF-tags `Confidential` and `Sensitive`.

1. Follow these steps to grant permissions to the `lf-data-analyst` user on the objects associated with the LF-Tag `Confidential=True` (Database:tag_database) to have `Describe` the database and `Select` permission on tables.
    a. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as `lf-data-engineer`.
    b. Under **Permissions**, select **Data lake permissions**.
    c. Choose **Grant**.
    d. Under **Principals**, select **IAM users and roles**.

e. For **IAM users and roles**, choose `lf-data-analyst`.

f. Under **LF-Tags or catalog resources**, select **Resources matched by LF-Tags**.

g. Choose **Add LF-tag**.

h. For **Key**, choose `Confidential`.

i. For **Values**, choose `True`.

j. For **Database permissions**, select `Describe`.

k. For **Table permissions**, choose **Select** and **Describe**.

l. Choose **Grant**.

2. Next, repeat the steps to grant permissions to data analysts for LF-Tag expression for `Confidential=False`. This **LF-tag** is used for describing the `col_tag_database` and the table `source_data_col_lvl` when logged in as `lf-data-analyst` from Amazon Athena.

   a. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as `lf-data-engineer`.

   b. On the **Databases** page, select the database `col_tag_database`.

   c. Choose **Action** and **Grant**.

   d. Under **Principals**, select **IAM users and roles**.

   e. For **IAM users and roles**, choose `lf-data-analyst`.

   f. Select **Resources matched by LF-Tags**.

   g. Choose **Add LF-Tag**.

   h. For **Key**, choose `Confidential`.

   i. For **Values**¸ choose `False`.

   j. For **Database permissions**, select `Describe`.

   k. For **Table permissions**, do not select anything.

   l. Choose **Grant**.

3. Next, repeat the steps to grant permissions to data analysts for LF-Tag expression for `Confidential=False` and `Sensitive=True`. This LF-tag is used for describing the `col_tag_database` and the table `source_data_col_lvl` (column-level) when logged in as `lf-data-analyst` from Amazon Athena.

   a. Sign into the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as `lf-data-engineer`.

   b. On the Databases page, select the database `col_tag_database`.

c. Choose **Action** and **Grant**.

d. Under **Principals**, select **IAM users and roles**.

e. For **IAM users and roles**, choose `lf-data-analyst`.

f. Select **Resources matched by LF-Tags**.

g. Choose **Add LF-tag**.

h. For **Key**, choose `Confidential`.

i. For **Values**, choose `False`.

j. Choose **Add LF-tag**.

k. For **Key**, choose `Sensitive`.

l. For **Values**, choose `True`.

m. For **Database permissions**, select `Describe`.

n. For **Table permissions**, select `Select` and `Describe`.

o. Choose **Grant**.

Step 5: Run a query in Amazon Athena to verify the permissions

For this step, use Amazon Athena to run SELECT queries against the two tables (`source_data` and `source_data_col_lvl`). Use the Amazon S3 path as the query result location (`s3://lf-tagbased-demo-`*`Account-ID`*`/athena-results/`).

1. Sign into the Athena console at https://console.aws.amazon.com/athena/ as `lf-data-analyst`.
2. In the Athena query editor, choose `tag_database` in the left panel.
3. Choose the additional menu options icon (three vertical dots) next to `source_data` and choose **Preview table**.
4. Choose **Run query**.

   The query should take a few minutes to run. The query displays all the columns in the output because the LF-tag is associated at the database level and the `source_data` table automatically inherited the `LF-tag` from the database `tag_database`.
5. Run another query using `col_tag_database` and `source_data_col_lvl`.

   The second query returns the two columns that were tagged as `Non-Confidential` and `Sensitive`.

6. You can also check to see the Lake Formation tag-based access policy behavior on columns to which you do not have policy grants. When an untagged column is selected from the table `source_data_col_lvl`, Athena returns an error. For example, you can run the following query to choose untagged columns `geolocationid`:

```
SELECT geolocationid FROM "col_tag_database"."source_data_col_lvl" limit 10;
```

Step 6: Clean up AWS resources

To prevent unwanted charges to your AWS account, you can delete the AWS resources that you used for this tutorial.

1. Sign into Lake Formation console as `lf-data-engineer` and delete the databases `tag_database` and `col_tag_database`.
2. Next, sign in as `lf-data-steward` and clean up all the **LF-tag Permissions**, **Data Permissions** and **Data Location Permissions** that were granted above that were granted `lf-data-engineer` and `lf-data-analyst`..
3. Sign into the Amazon S3 console as the account owner using the IAM credentials you used to deploy the CloudFormation stack.
4. Delete the following buckets:
   - lf-tagbased-demo-accesslogs-*acct-id*
   - lf-tagbased-demo-*acct-id*
5. Sign into CloudFormation console at https://console.aws.amazon.com/cloudformation, and delete the stack you created. Wait for the stack status to change to `DELETE_COMPLETE`.

Securing data lakes with row-level access control

AWS Lake Formation row-level permissions allow you to provide access to specific rows in a table based on data compliance and governance policies. If you have large tables storing billions of records, you need a way to enable different users and teams to access only the data they are allowed to see. Row-level access control is a simple and performant way to protect data, while giving users access to the data they need to perform their job. Lake Formation provides centralized auditing and compliance reporting by identifying which principals accessed what data, when, and through which services.

In this tutorial, you learn how row-level access controls work in Lake Formation, and how to set them up.

This tutorial includes an AWS CloudFormation template for quickly set up the required resources. You can review and customize it to suit your needs.

Topics

- Intended audience
- Prerequisites
- Step 1: Provision your resources
- Step 2: Query without data filters
- Step 3: Set up data filters and grant permissions
- Step 4: Query with data filters
- Step 5: Clean up AWS resources

Intended audience

This tutorial is intended for data stewards, data engineers, and data analysts. The following table lists the roles and responsibilities of a data owner and a data consumer.

Role	Description
IAM Administrator	A user who can create users and roles and Amazon Simple Storage Service (Amazon S3) buckets. Has the `AdministratorAccess` AWS managed policy.
Data lake administrator	A user responsible for setting up the data lake, creating data filters, and granting permissions to data analysts.
Data analyst	A user who can run queries against the data lake. Data analysts residing in different countries (for our use case, the US and Japan) can only analyze product reviews for customers located in their own country and

Role	Description
	for compliance reasons, should not be able to see customer data located in other countries.

Prerequisites

Before you start this tutorial, you must have an AWS account that you can use to sign in as an administrative user with correct permissions. For more information, see Complete initial AWS configuration tasks.

The tutorial assumes that you are familiar with IAM. For information about IAM, see the IAM User Guide.

Change Lake Formation settings

> **Important**
>
> Before launching the CloudFormation template, disable the option **Use only IAM access control for new databases/tables** in Lake Formation by following the steps below:

1. Sign into the Lake Formation console at https://console.aws.amazon.com/lakeformation/ in the US East (N. Virginia) region or US West (Oregon) region.
2. Under Data Catalog, choose **Settings**.
3. Deselect **Use only IAM access control for new databases** and **Use only IAM access control for new tables in new databases**.
4. Choose **Save**.

Step 1: Provision your resources

This tutorial includes an CloudFormation template for a quick setup. You can review and customize it to suit your needs. The CloudFormation template generates the following resources:

- Users and policies for:
  - DataLakeAdmin
  - DataAnalystUS

  - DataAnalystJP
- Lake Formation data lake settings and permissions
- A Lambda function (for Lambda-backed CloudFormation custom resources) used to copy sample data files from the public Amazon S3 bucket to your Amazon S3 bucket
- An Amazon S3 bucket to serve as our data lake
- An AWS Glue Data Catalog database, table, and partition

Create your resources

Follow these steps to create your resources using the CloudFormation template.

1. Sign into the CloudFormation console at https://console.aws.amazon.com/cloudformation in the US East (N. Virginia) region.
2. Choose Launch Stack.
3. Choose **Next** on the **Create stack** screen.
4. Enter a **Stack name.**
5. For **DatalakeAdminUserName** and **DatalakeAdminUserPassword**, enter your IAM user name and password for data lake admin user.
6. For **DataAnalystUsUserName** and **DataAnalystUsUserPassword**, enter the user name and password for user name and password you want for the data analyst user who is responsible for the US marketplace.
7. For **DataAnalystJpUserName** and **DataAnalystJpUserPassword**, enter the user name and password for user name and password you want for the data analyst user who is responsible for the Japanese marketplace.
8. For **DataLakeBucketName**, enter the name of your data bucket.
9. For **DatabaseName**, and **TableName** leave as the default.
10. Choose **Next**
11. On the next page, choose **Next**.
12. Review the details on the final page and select **I acknowledge that CloudFormation might create IAM resources.**
13. Choose **Create**.

    The stack creation can take one minute to complete.

Step 2: Query without data filters

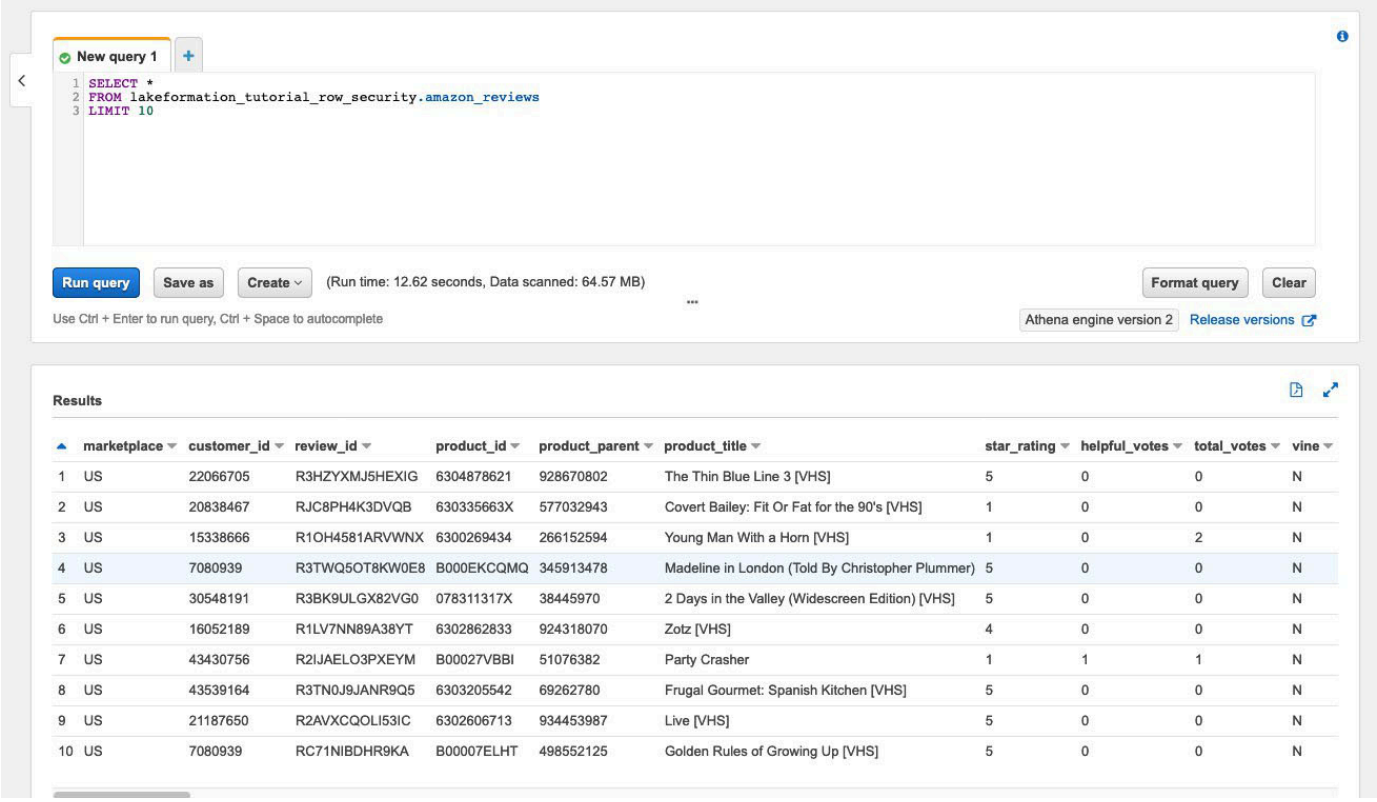
After you set up the environment, you can query the product reviews table. First query the table without row-level access controls to make sure you can see the data. If you are running queries in Amazon Athena for the first time, you need to configure the query result location.

Query the table without row-level access control

1. Sign into Athena console at https://console.aws.amazon.com/athena/ as the `DatalakeAdmin` user, and run the following query:

```
SELECT *
FROM lakeformation_tutorial_row_security.amazon_reviews
LIMIT 10
```

   The following screenshot shows the query result. This table has only one partition, `product_category=Video`, so each record is a review comment for a video product.



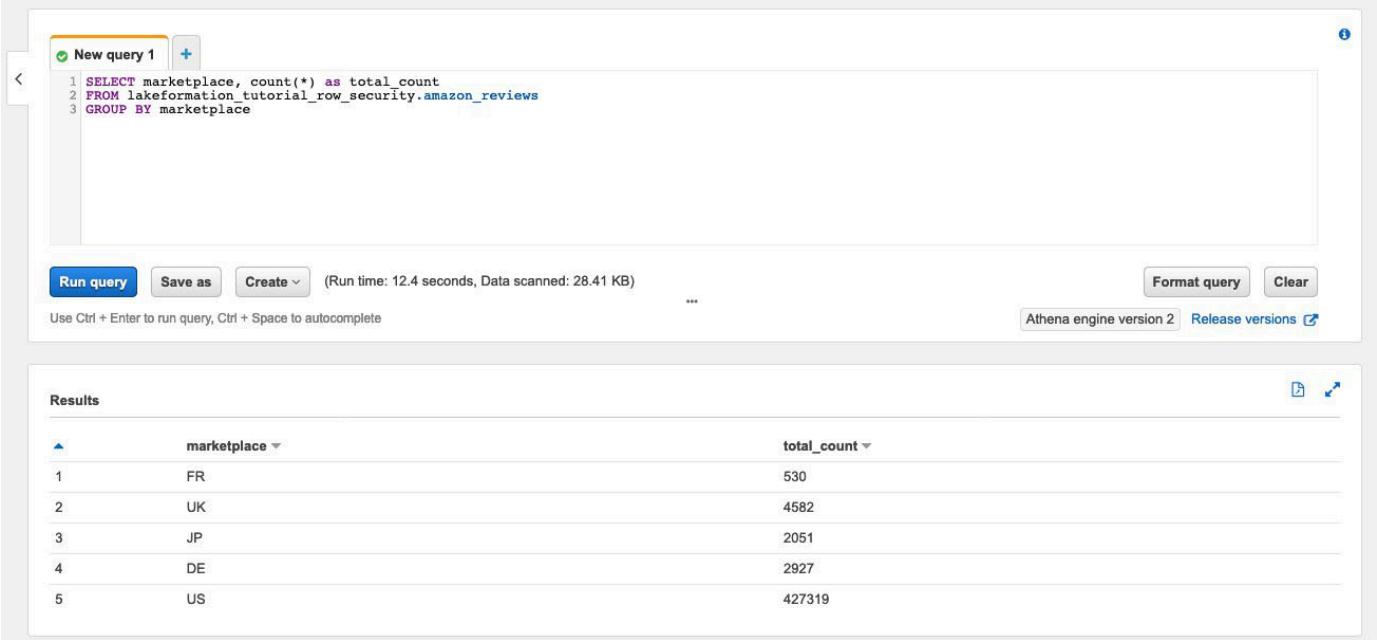
	marketplace	customer_id	review_id	product_id	product_parent	product_title	star_rating	helpful_votes	total_votes	vine
1	US	22066705	R3HZYXMJ5HEXIG	6304878621	928670802	The Thin Blue Line 3 [VHS]	5	0	0	N
2	US	20838467	RJC8PH4K3DVQB	630335663X	577032943	Covert Bailey: Fit Or Fat for the 90's [VHS]	1	0	0	N
3	US	15338666	R1OH4581ARVWNX	6300269434	266152594	Young Man With a Horn [VHS]	1	0	2	N
4	US	7080939	R3TWQ5OT8KW0E8	B000EKCQMQ	345913478	Madeline in London (Told By Christopher Plummer)	5	0	0	N
5	US	30548191	R3BK9ULGX82VG0	078311317X	38445970	2 Days in the Valley (Widescreen Edition) [VHS]	5	0	0	N
6	US	16052189	R1LV7NN89A38YT	6302862833	924318070	Zotz [VHS]	4	0	0	N
7	US	43430756	R2IJAELO3PXEYM	B00027VBBI	51076382	Party Crasher	1	1	1	N
8	US	43539164	R3TN0J9JANR9Q5	6303205542	69262780	Frugal Gourmet: Spanish Kitchen [VHS]	5	0	0	N
9	US	21187650	R2AVXCQOLI53IC	6302606713	934453987	Live [VHS]	5	0	0	N
10	US	7080939	RC71NIBDHR9KA	B00007ELHT	498552125	Golden Rules of Growing Up [VHS]	5	0	0	N

2. Next, run an aggregation query to retrieve the total number of records per `marketplace`.

```
SELECT marketplace, count(*) as total_count
FROM lakeformation_tutorial_row_security.amazon_reviews
```

```
GROUP BY marketplace
```

The following screenshot shows the query result. The `marketplace` column has five different values. In the subsequent steps, you will set up row-based filters using the `marketplace` column.



Step 3: Set up data filters and grant permissions

This tutorial uses two data analysts: one responsible for the US marketplace and another for the Japanese marketplace. Each analyst uses Athena to analyze customer reviews for their specific marketplace only. Create two different data filters, one for the analyst responsible for the US marketplace, and another for the one responsible for the Japanese marketplace. Then, grant the analysts their respective permissions.

Create data filters and grant permissions

1. Create a filter to restrict access to the US `marketplace` data.

   a. Sign into the Lake Formation console at https://console.aws.amazon.com/lakeformation/ in US East (N. Virginia) region as the `DatalakeAdmin` user.

   b. Choose **Data filters**.

   c. Choose **Create new filter**.

   d. For **Data filter name**, enter `amazon_reviews_US`.

e. For **Target database**, choose the database `lakeformation_tutorial_row_security`.

f. For **Target table**, choose the table `amazon_reviews`.

g. For **Column-level access**, leave as the default.

h. For **Row filter expression**, enter `marketplace='US'`.

i. Choose **Create filter**.

2. Create a filter to restrict access to the Japanese `marketplace` data.

a. On the **Data filters** page, choose **Create new filter**.

b. For **Data filter name**, enter `amazon_reviews_JP`.

c. For **Target database**, choose the database `lakeformation_tutorial_row_security`.

d. For **Target table**, choose the `table amazon_reviews`.

e. For **Column-level access**, leave as the default.

f. For Row filter expression, enter `marketplace='JP'`.

g. Choose **Create filter**.

3. Next, grant permissions to the data analysts using these data filters. Follow these steps to grant permissions to the US data analyst (`DataAnalystUS`):

a. Under **Permissions**, choose **Data lake permissions**.

b. Under **Data permission**, choose **Grant**.

c. For **Principals**, choose **IAM users and roles**, and select the role `DataAnalystUS`.

d. For **LF tags or catalog resources**, choose **Named data catalog resources**.

e. For **Database**, choose `lakeformation_tutorial_row_security`.

f. For **Tables-optional**, choose `amazon_reviews`.

g. For **Data filters – optional**¸ select `amazon_reviews_US`.

h. For **Data filter permissions**, select **Select**.

i. Choose **Grant**.

4. Follow these steps to grant permissions to the Japanese data analyst (`DataAnalystJP`):

a. Under **Permissions**, choose **Data lake permissions**.

b. Under **Data permission**, choose **Grant**.

c. For **Principals**, choose **IAM users and roles**, and select the role `DataAnalystJP`.

d. For **LF tags or catalog resources**, choose **Named data catalog resources**.

e. For **Database**, choose `lakeformation_tutorial_row_security`.

f. For **Tables-optional**, choose `amazon_reviews`.

g. For **Data filters – optional**, select `amazon_reviews_JP`.

h. For **Data filter permissions**, select **Select**.

i. Choose **Grant**.

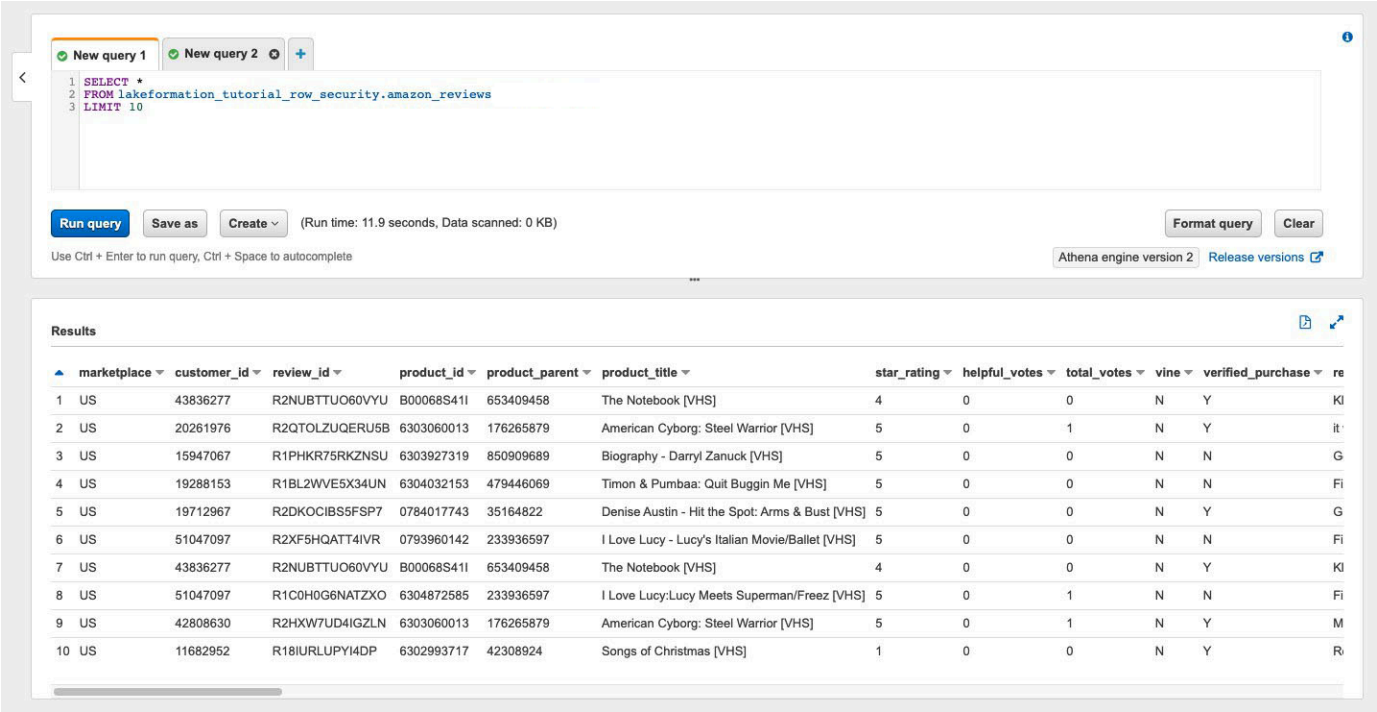
Step 4: Query with data filters

With the data filters attached to the product reviews table, run some queries and see how permissions are enforced by Lake Formation.

1. Sign into the Athena console at https://console.aws.amazon.com/athena/ as the `DataAnalystUS` user.
2. Run the following query to retrieve a few records, which are filtered based on the row-level permissions we defined:

```
SELECT *
FROM lakeformation_tutorial_row_security.amazon_reviews
LIMIT 10
```

The following screenshot shows the query result.



	marketplace	customer_id	review_id	product_id	product_parent	product_title	star_rating	helpful_votes	total_votes	vine	verified_purchase	re
1	US	43836277	R2NUBTTUO60VYU	B00068S41I	653409458	The Notebook [VHS]	4	0	0	N	Y	Kl
2	US	20261976	R2QTOLZUQERU5B	6303060013	176265879	American Cyborg: Steel Warrior [VHS]	5	0	1	N	Y	it
3	US	15947067	R1PHKR75RKZNSU	6303927319	850909689	Biography - Darryl Zanuck [VHS]	5	0	0	N	N	G
4	US	19288153	R1BL2WVE5X34UN	6304032153	479446069	Timon & Pumbaa: Quit Buggin Me [VHS]	5	0	0	N	N	Fi
5	US	19712967	R2DKOCIBS5FSP7	0784017743	35164822	Denise Austin - Hit the Spot: Arms & Bust [VHS]	5	0	0	N	Y	G
6	US	51047097	R2XF5HQATT4IVR	0793960142	233936597	I Love Lucy - Lucy's Italian Movie/Ballet [VHS]	5	0	0	N	N	Fi
7	US	43836277	R2NUBTTUO60VYU	B00068S41I	653409458	The Notebook [VHS]	4	0	0	N	Y	Kl
8	US	51047097	R1C0H0G6NATZXO	6304872585	233936597	I Love Lucy:Lucy Meets Superman/Freez [VHS]	5	0	1	N	N	Fi
9	US	42808630	R2HXW7UD4IGZLN	6303060013	176265879	American Cyborg: Steel Warrior [VHS]	5	0	1	N	Y	M
10	US	11682952	R18IURLUPYI4DP	6302993717	42308924	Songs of Christmas [VHS]	1	0	0	N	Y	R

3. Similarly, run a query to count the total number of records per marketplace.

```
SELECT marketplace , count ( * ) as total_count
FROM lakeformation_tutorial_row_security .amazon_reviews
GROUP BY marketplace
```

The query result only shows the `marketplace` US in the results. This is because the user is only allowed to see rows where the `marketplace` column value is equal to US.

4. Switch to the `DataAnalystJP` user and run the same query.

```
SELECT *
FROM lakeformation_tutorial_row_security.amazon_reviews
LIMIT 10
```

The query result shows only the records belong to the `JP` `marketplace`.

5. Run the query to count the total number of records per `marketplace`.

```
SELECT marketplace, count(*) as total_count
FROM lakeformation_tutorial_row_security.amazon_reviews
GROUP BY marketplace
```

The query result shows only the row belonging to the `JP` `marketplace`.

Step 5: Clean up AWS resources

Clean up resources

To prevent unwanted charges to your AWS account, you can delete the AWS resources that you used for this tutorial.

- Delete the cloud formation stack.

Sharing a data lake using Lake Formation tag-based access control and named resources

This tutorial demonstrates how you can configure AWS Lake Formation to securely share data stored within a data lake with multiple companies, organizations, or business units, without having to copy the entire database. There are two options to share your databases and tables with another AWS account by using Lake Formation cross-account access control:

- **Lake Formation tag-based access control (recommended)**

  Lake Formation tag-based access control is an authorization strategy that defines permissions based on attributes. In Lake Formation, these attributes are called *LF-Tags*. For more details, refer to Managing a data lake using Lake Formation tag-based access control.
- **Lake Formation named resources**

  The Lake Formation named resource method is an authorization strategy that defines permissions for resources. Resources include databases, tables, and columns. Data lake administrators can assign and revoke permissions on Lake Formation resources. For more details, refer to Cross-account data sharing in Lake Formation.

  We recommend using named resources if the data lake administrator prefers granting permissions explicitly to individual resources. When you use the named resource method to grant Lake Formation permissions on a Data Catalog resource to an external account, Lake Formation uses AWS Resource Access Manager (AWS RAM) to share the resource.

Topics

- Intended audience
- Configure Lake Formation Data Catalog settings in the producer account

- Step 1: Provision your resources using AWS CloudFormation templates
- Step 2: Lake Formation cross-account sharing prerequisites
- Step 3: Implement cross-account sharing using the tag-based access control method
- Step 4: Implement the named resource method
- Step 5: Clean up AWS resources

Intended audience

This tutorial is intended for data stewards, data engineers, and data analysts. When it comes to sharing Data Catalog tables from AWS Glue and administering permission in Lake Formation, data stewards within the producing accounts have functional ownership based on the functions they support, and can grant access to various consumers, external organizations, and accounts. The following table lists the roles that are used in this tutorial:

Role	Description
DataLakeAdminProducer	The data lake admin IAM user has the following access: • Full read, write, and update access to all resources in the Data Catalog • Ability to grant permissions to resources • Can create resource links for the shared table • Can attach LF-Tags to resources, which provides access to principals based on any policies created by data stewards
DataLakeAdminConsumer	The data lake admin IAM user has the following access: • Full read, write, and update access to all resources in the Data Catalog • Ability to grant permissions to resources

Role	Description
	• Can create resource links for the shared table • Can attach LF-Tags to resources, which provides access to principals based on any policies created by data stewards
DataAnalyst	The DataAnalyst user has the following access: • Fine-grained access to resources shared by Lake Formation tag-based access policies or using named resources method

Configure Lake Formation Data Catalog settings in the producer account

Before you start this tutorial, you must have an AWS account that you can use to sign in as an administrative user with correct permissions. For more information, see Complete initial AWS configuration tasks.

The tutorial assumes that you are familiar with IAM. For information about IAM, see the IAM User Guide.

Configure Lake Formation Data Catalog settings in the producer account

> **Note**
>
> In this tutorial, the account that has the source table is called the producer account, and the account that needs access to the source table is called a consumer account.

Lake Formation provides its own permission management model. To maintain backward compatibility with the IAM permission model, the `Super` permission is granted to the group `IAMAllowedPrincipals` on all existing AWS Glue Data Catalog resources by default. Also, **Use only IAM access control settings** are enabled for new Data Catalog resources. This tutorial uses fine grained access control using Lake Formation permissions and use IAM policies for coarse grained access control. See Methods for fine-grained access control for details. Therefore, before

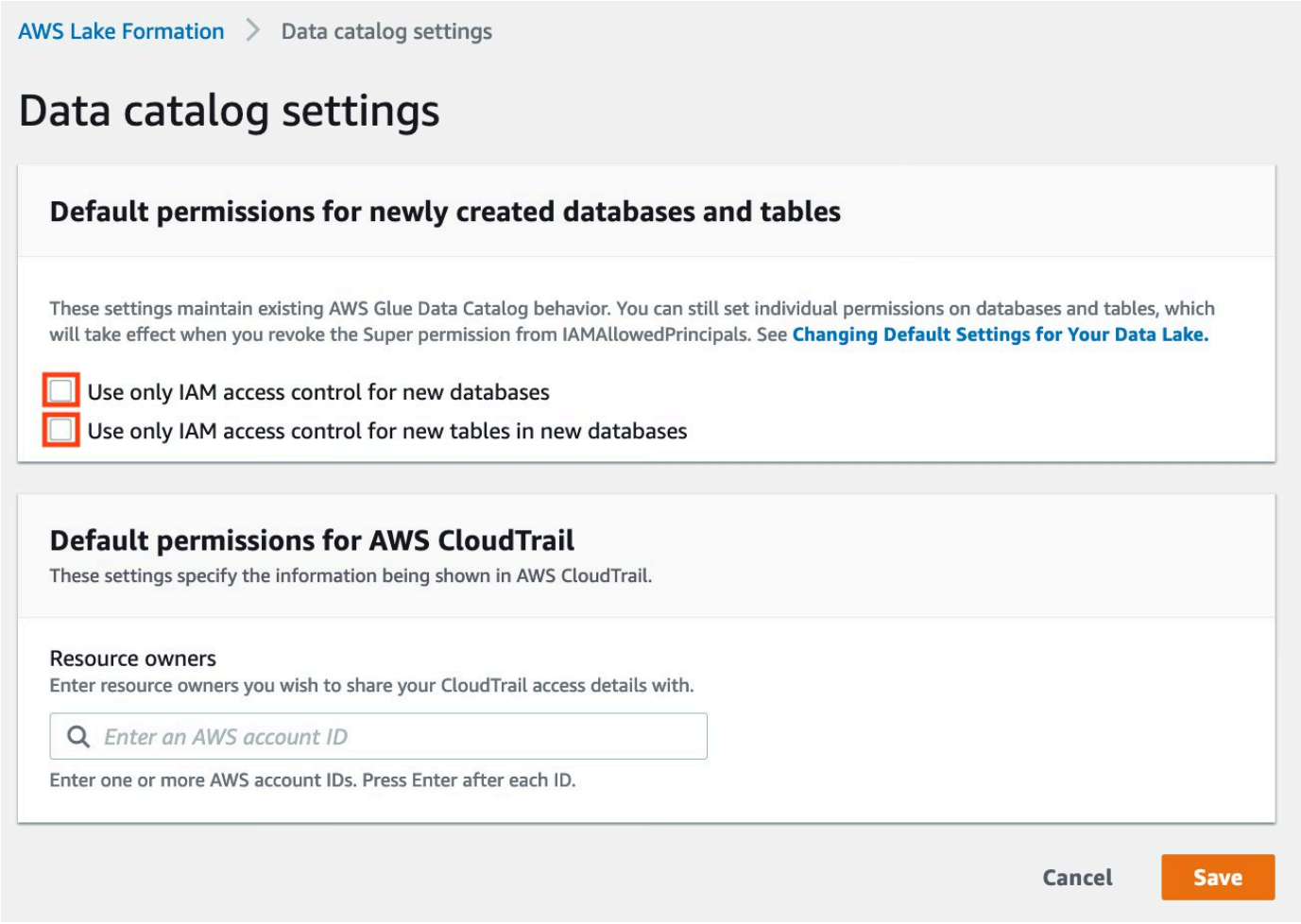
you use an AWS CloudFormation template for a quick setup, you need to change Lake Formation Data Catalog settings in the producer account.

> **Important**
>
> This setting affects all newly created databases and tables, so we strongly recommend completing this tutorial in a non-production account or in a new account. Also, if you are using a shared account (such as your company's development account), make sure it does not affect others resources. If you prefer to keep the default security settings, you must complete an extra step when sharing resources to other accounts, in which you revoke the default **Super** permission from `IAMAllowedPrincipals` on the database or table. We discuss the details later in this tutorial.

To configure Lake Formation Data Catalog settings in the producer account, complete the following steps:

1. Sign into the AWS Management Console using the producer account as an admin user, or as a user with Lake Formation `PutDataLakeSettings` API permission.
2. On the Lake Formation console, in the navigation pane, under **Data Catalog**, choose **Settings**.
3. Deselect **Use only IAM access control for new databases** and **Use only IAM access control for new tables in new databases**

   Choose **Save**.



Additionally, you can remove `CREATE_DATABASE` permissions for `IAMAllowedPrincipals` under **Administrative roles and tasks**, **Database creators**. Only then, you can govern who can create a new database through Lake Formation permissions.

Step 1: Provision your resources using AWS CloudFormation templates

The CloudFormation template for the producer account generates the following resources:

- An Amazon S3 bucket to serve as the data lake.
- A Lambda function (for Lambda-backed CloudFormation custom resources). We use the function to copy sample data files from the public Amazon S3 bucket to your Amazon S3 bucket.
- IAM users and policies: DataLakeAdminProducer.
- The appropriate Lake Formation settings and permissions including:
  - Defining the Lake Formation data lake administrator in the producer account

  - Registering an Amazon S3 bucket as the Lake Formation data lake location (producer account)
- An AWS Glue Data Catalog database, table, and partition. Since there are two options for sharing resources across AWS accounts, this template creates two separate sets of database and table.

The CloudFormation template for the consumer account generates the following resources:

- IAM users and policies:
  - DataLakeAdminConsumer
  - DataAnalyst
- An AWS Glue Data Catalog database. This database is for creating resource links to shared resources.

Create your resources in the producer account

1. Sign into the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation in the US East (N. Virginia) region.
2. Choose Launch Stack.
3. Choose **Next**.
4. For **Stack name**, enter a stack name, such as `stack-producer`.
5. In the **User Configuration** section, enter user name and password for `ProducerDatalakeAdminUserName` and `ProducerDatalakeAdminUserPassword`.
6. For **DataLakeBucketName**, enter the name of your data lake bucket. This name needs to be globally unique.
7. For **DatabaseName** and **TableName**, leave the default values.
8. Choose **Next**.
9. On the next page, choose **Next**.
10. Review the details on the final page and select **I acknowledge that AWS CloudFormation might create IAM resources**.
11. Choose **Create**.

    The stack creation can take up to one minute.

Create your resources in the consumer account

1. Sign into the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation in the US East (N. Virginia) region.
2. Choose Launch Stack.
3. Choose **Next**.
4. For **Stack name**, enter a stack name, such as `stack-consumer`.
5. In the **User Configuration** section, enter user name and password for `ConsumerDatalakeAdminUserName` and `ConsumerDatalakeAdminUserPassword`.
6. For `DataAnalystUserName` and `DataAnalystUserPassword`, enter the user name and password you want for the data analyst IAM user.
7. For **DataLakeBucketName**, enter the name of your data lake bucket. This name needs to be globally unique.
8. For **DatabaseName**, leave the default values.
9. For `AthenaQueryResultS3BucketName`, enter the name of the Amazon S3 bucket that stores Amazon Athena query results. If you don't have one, create an Amazon S3 bucket.
10. Choose **Next**.
11. On the next page, choose **Next**.
12. Review the details on the final page and select **I acknowledge that AWS CloudFormation might create IAM resources**.
13. Choose **Create**.

    The stack creation can take up to one minutes.

> **Note**
>
> After completing the tutorial, delete the stack in CloudFormation to avoid incurring charges. Verify that the resources are successfully deleted in the event status for the stack.

Step 2: Lake Formation cross-account sharing prerequisites

Before sharing resources with Lake Formation, there are prerequisites for both the tag-based access control method and named resource method.

Complete tag-based access control cross-account data sharing prerequisites

- For more information on cross-account data sharing requirements, see the Prerequisites section in the Cross-account data sharing chapter.

  To share Data Catalog resources with version 3 or above of the **Cross account version settings**, the grantor requires to have the IAM permissions defined in the AWS managed policy `AWSLakeFormationCrossAccountManager` in your account.

  If you are using version 1 or version 2 of the **Cross account version settings**, before you can use the tag-based access control method to grant cross-account access to resources, you must add the following `JSON` permissions object to the Data Catalog resource policy in the producer account. This gives the consumer account permission to access the Data Catalog when `glue:EvaluatedByLakeFormationTags` is true. Also, this condition becomes true for resources on which you granted permission using Lake Formation permission tags to the consumer's account. This policy is required for every AWS account to which you are granting permissions.

  The following policy must be within a `Statement` element. We discuss the full IAM policy in the next section.

```
{
    "Effect": "Allow",
    "Action": [
        "glue:*"
    ],
    "Principal": {
        "AWS": [
            "consumer-account-id"
        ]
    },
    "Resource": [
        "arn:aws:glue:region:account-id:table/*",
        "arn:aws:glue:region:account-id:database/*",
        "arn:aws:glue:region:account-id:catalog"
    ],
    "Condition": {
        "Bool": {
            "glue:EvaluatedByLakeFormationTags": true
        }
    }
```

```
}
```

Complete named resource method cross-account sharing prerequisites

1. If there is no Data Catalog resource policy in your account, the Lake Formation cross-account grants that you make proceed as usual. However, if a Data Catalog resource policy exists, you must add the following statement to it to permit your cross-account grants to succeed if they're made with the named resource method. If you plan to use only the named resource method, or only the tag-based access control method, you can skip this step. In this tutorial, we evaluate both methods, and we need to add the following policy.

   The following policy must be within a `Statement` element. We discuss the full IAM policy in the next section.

   ```
   {
             "Effect": "Allow",
             "Action": [
             "glue:ShareResource"
             ],
             "Principal": {
               "Service":"ram.amazonaws.com"
             },
             "Resource": [
                 "arn:aws:glue:region:account-id:table/*/*",
                 "arn:aws:glue:region:account-id:database/*",
                 "arn:aws:glue:region:account-id:catalog"
             ]
   }
   ```

2. Next, add the AWS Glue Data Catalog resource policy using the AWS Command Line Interface (AWS CLI).

   If you grant cross-account permissions by using both the tag-based access control method and named resource method, you must set the `EnableHybrid` argument to 'true' when adding the preceding policies. Because this option is not currently supported on the console, and you must use the `glue:PutResourcePolicy` API and AWS CLI.

First, create a policy document (such as policy.json) and add the preceding two policies. Replace *consumer-account-id* with the *account ID* of the AWS account receiving the grant, *region* with the Region of the Data Catalog containing the databases and tables that you are granting permissions on, and *account-id* with the producer AWS account ID.

Enter the following AWS CLI command. Replace *glue-resource-policy* with the correct values (such as file://policy.json).

```
aws glue put-resource-policy --policy-in-json glue-resource-policy --enable-hybrid
 TRUE
```

For more information, see put-resource-policy.

Step 3: Implement cross-account sharing using the tag-based access control method

In this section, we walk you through the following high-level steps:

1. Define an LF-Tag.
2. Assign the LF-Tag to the target resource.
3. Grant LF-Tag permissions to the consumer account.
4. Grant data permissions to the consumer account.
5. Optionally, revoke permissions for `IAMAllowedPrincipals` on the database, tables, and columns.
6. Create a resource link to the shared table.
7. Create an LF-Tag and assign it to the target database.
8. Grant LF-Tag data permissions to the consumer account.

Define an LF-Tag

> **Note**
>
> If you are signed in to your producer account, sign out before completing the following steps.

1. Sign into the producer account as the data lake administrator at https://console.aws.amazon.com/lakeformation/. Use the producer account number, IAM user name (the default is `DatalakeAdminProducer`), and password that you specified during CloudFormation stack creation.
2. On the Lake Formation console (https://console.aws.amazon.com/lakeformation/), in the navigation pane, under **Permissions**, choose **LF-Tags and Permissions**.
3. Choose **Add LF-Tag**.

Assign the LF-Tag to the target resource

Assign the LF-Tag to the target resource and grant data permissions to another account

As a data lake administrator, you can attach tags to resources. If you plan to use a separate role, you may have to grant describe and attach permissions to the separate role.

1. In the navigation pane, under **Data Catalog**, select **Databases**.
2. Select the target database (`lakeformation_tutorial_cross_account_database_tbac`) and on the **Actions** menu, choose **Edit LF-Tags**.

   For this tutorial, you assign an LF-Tag to a database, but you can also assign LF-Tags to tables and columns.
3. Choose **Assign new LF-Tag**.
4. Add the key `Confidentiality` and value `public`.
5. Choose **Save**.

Grant LF-Tag permission to the consumer account

Still in the producer account, grant permissions to the consumer account to access the LF-Tag.

1. In the navigation pane, under **Permissions**, choose **LF-Tags and permissions**.
2. Choose the **LF-Tags** tab, and choose the **key** and **values** of the LF-Tag that is being shared with the consumer account (**key** `Confidentiality` and **value** `public`).
3. Choose **Grant permissions**.
4. For **Permission type**, choose **LF-Tag key-value pair permissions.**

5. For **Principals**, choose **External accounts**.
6. Enter the target **AWS account ID**.

   AWS accounts within the same organization appear automatically. Otherwise, you have to manually enter the AWS account ID.
7. Under **Permissions**, select **Describe**.

   This is the permissions given to the consumer account. Grantable permissions are permissions that the consumer account can grant to other principals.
8. Choose **Grant**.

   At this point, the consumer data lake administrator should be able to find the policy tag being shared via the consumer account Lake Formation console, under **Permissions**, **LF-Tags and permissions**.

Grant data permission to the consumer account

We will now provide data access to the consumer account by specifying an LF-Tag expression and granting the consumer account access to any table or database that matches the expression..

1. In the navigation pane, under **Permissions**,**Data lake permissions**, choose **Grant**.
2. For **Principals**, choose **External accounts**, and enter the target AWS account ID.
3. For **LF-Tags or catalog resources**, choose the **key** and **values** of the **LF-Tag** that is being shared with the consumer account (**key** `Confidentiality` and **value** `public`).
4. For **Permissions**, under **Resources matched by LF-Tags (recommended)** choose **Add LF-Tag**.
5. Select the **key** and **value** of the tag that is being shared with the consumer account (key `Confidentiality` and value `public`).
6. For **Database permissions**, select **Describe** under **Database permissions** to grant access permissions at the database level.
7. The consumer data lake administrator should be able to find the policy tag being shared via the consumer account on the Lake Formation console at https://console.aws.amazon.com/lakeformation/, under **Permissions**, **Administrative roles and tasks**, **LF-Tags**.
8. Select **Describe** under **Grantable permissions** so the consumer account can grant database-level permissions to its users.
9. For **Table and column permissions**, select **Select** and **Describe** under **Table permissions**.

10. Select **Select** and **Describe** under **Grantable permissions**.
11. Choose **Grant**.

Revoke permission for `IAMAllowedPrincipals` on the database, tables, and columns (Optional).

At the very beginning of this tutorial, you changed the Lake Formation Data Catalog settings. If you skipped that part, this step is required. If you changed your Lake Formation Data Catalog settings, you can skip this step.

In this step, we need to revoke the default **Super** permission from `IAMAllowedPrincipals` on the database or table. See Step 4: Switch your data stores to the Lake Formation permissions model for details.

Before revoking permission for `IAMAllowedPrincipals`, make sure that you granted existing IAM principals with necessary permission through Lake Formation. This includes three steps:

1. Add IAM permission to the target IAM user or role with the Lake Formation `GetDataAccess` action (with IAM policy).
2. Grant the target IAM user or role with Lake Formation data permissions (alter, select, and so on).
3. Then, revoke permissions for `IAMAllowedPrincipals`. Otherwise, after revoking permissions for `IAMAllowedPrincipals`, existing IAM principals may no longer be able to access the target database or Data Catalog.

   Revoking **Super** permission for `IAMAllowedPrincipals` is required when you want to apply the Lake Formation permission model (instead of the IAM policy model) to manage user access within a single account or among multiple accounts using the Lake Formation permission model. You do not have to revoke permission of `IAMAllowedPrincipals` for other tables where you want to keep the traditional IAM policy model.

   At this point, the consumer account data lake administrator should be able to find the database and table being shared via the consumer account on the Lake Formation console at https://console.aws.amazon.com/lakeformation/, under **Data Catalog, databases**. If not, confirm if the following are properly configured:

   1. The correct policy tag and values are assigned to the target databases and tables.
   2. The correct tag permission and data permission are assigned to the consumer account.

3. Revoke the default super permission from `IAMAllowedPrincipals` on the database or table.

Create a resource link to the shared table

When a resource is shared between accounts, and the shared resources are not put in the consumer accounts' Data Catalog. To make them available, and query the underlying data of a shared table using services like Athena, we need to create a resource link to the shared table. A resource link is a Data Catalog object that is a link to a local or shared database or table. For details, see Creating resource links. By creating a resource link, you can:

- Assign a different name to a database or table that aligns with your Data Catalog resource naming policies.
- Use services such as Athena and Redshift Spectrum to query shared databases or tables.

To create a resource link, complete the following steps:

1. If you are signed into your consumer account, sign out.
2. Sign in as the consumer account data lake administrator. Use the consumer account ID, IAM user name (default DatalakeAdminConsumer) and password that you specified during CloudFormation stack creation.
3. On the Lake Formation console (https://console.aws.amazon.com/lakeformation/), in the navigation pane, under **Data Catalog, Databases**, choose the shared database `lakeformation_tutorial_cross_account_database_tbac`.

   If you don't see the database, revisit the previous steps to see if everything is properly configured.
4. Choose **View Tables**.
5. Choose the shared table `amazon_reviews_table_tbac`.
6. On the **Actions** menu, choose **Create resource link**.
7. For **Resource link name**, enter a name (for this tutorial, `amazon_reviews_table_tbac_resource_link`).
8. Under **Database**, select the database that the resource link is created in (for this post, the CloudFormationn stack created the database `lakeformation_tutorial_cross_account_database_consumer`).

9. Choose **Create**.

   The resource link appears under **Data catalog**, **Tables**.

Create an LF-tag and assign it to the target database

Lake Formation tags reside in the same Data Catalog as the resources. This means that tags created in the producer account are not available to use when granting access to the resource links in the consumer account. You need to create a separate set of LF-tags in the consumer account to use LF tag-based access control when sharing the resource links in the consumer account.

1. Define the LF-tag in the consumer account. For this tutorial, we use key `Division` and values `sales`, `marketing`, and `analyst`.
2. Assign the LF-tag key `Division` and value `analyst` to the database `lakeformation_tutorial_cross_account_database_consumer`, where the resource link is created.

Grant LF-tag data permission to the consumer

As a final step, grant LF-tag data permission to the consumer.

1. In the navigation pane, under **Permissions**, **Data lake permissions**, choose **Grant**.
2. For **Principals**, choose **IAM users and roles**, and choose the user `DataAnalyst`.
3. For **LF-tags or catalog resources**, choose **Resources matched by LF-Tags** (recommended).
4. Choose **key** Division and **value** analyst.
5. For **Database permissions**, select **Describe** under **Database permissions**.
6. For **Table and column permissions**, select **Select** and **Describe** under **Table permissions**.
7. Choose **Grant**.
8. Repeat these steps for user `DataAnalyst`, where the LF-Tag key is `Confidentiality` and value is `public`.

   At this point, the data analyst user in the consumer account should be able to find the database and resource link, and query the shared table via the Athena console at https://console.aws.amazon.com/athena/. If not, confirm if the following are properly configured:

   - The resource link is created for the shared table
   - You granted the user access to the LF-Tag shared by the producer account

- You granted the user access to the LF-Tag associated to the resource link and database that the resource link is created in
- Check if you assigned the correct LF-Tag to the resource link, and to the database that the resource link is created in

Step 4: Implement the named resource method

To use the named resource method, we walk you through the following high-level steps:

1. Optionally, revoke permission for `IAMAllowedPrincipals` on the database, tables, and columns.
2. Grant data permission to the consumer account.
3. Accept a resource share from AWS Resource Access Manager.
4. Create a resource link for the shared table.
5. Grant data permission for the shared table to the consumer.
6. Grant data permission for the resource link to the consumer.

Revoke permission for `IAMAllowedPrincipals` on the database, tables, and columns (Optional)

- At the very beginning of this tutorial, we changed Lake Formation Data Catalog settings. If you skipped that part, this step is required. For instructions, see the optional step in the previous section.

Grant data permission to the consumer account

1. **Note**

   If you're signed in to producer account as another user, sign out first.

   Sign into the Lake Formation console at https://console.aws.amazon.com/lakeformation/ using the producer account data lake administrator using the AWS account ID, IAM user name (default is `DatalakeAdminProducer`), and password specified during CloudFormation stack creation.

2. On the **Permissions** page, under **Data lake Permissions** choose **Grant**.
3. Under **Principals**, choose **External accounts**, and enter one or more AWS account IDs or AWS organizations IDs. For more information see: AWS Organizations.

   Organizations that the producer account belongs to and AWS accounts within the same organization appear automatically. Otherwise, manually enter the account ID or organization ID.
4. For **LF-Tags or catalog resources**, choose `Named data catalog resources`.
5. Under **Databases**, choose the database `lakeformation_tutorial_cross_account_database_named_resource`.
6. Choose **Add LF-Tag**.
7. Under **Tables**, choose **All tables**.
8. For **Table column permissions**¸ choose **Select**, and **Describe** under **Table permissions**.
9. Select**Select** and **Describe**, under **Grantable Permissions**.
10. Optionally, for **Data permissions**, choose **Simple column-based access** if column-level permission management is required.
11. Choose **Grant**.

If you have not revoked permission for `IAMAllowedPrincipals`, you get a **Grant permissions** failed error. At this point, you should see the target table being shared via AWS RAM with the consumer account under **Permissions, Data permissions**.

Accept a resource share from AWS RAM

> **Note**
>
> This step is required only for AWS account-based sharing, not for organization-based sharing.

1. Sign into the AWS console at https://console.aws.amazon.com/connect/ using the consumer account data lake administrator using the IAM user name (default is DatalakeAdminConsumer) and password specified during CloudFormation stack creation.
2. On the AWS RAM console, in the navigation pane, under **Shared with me, Resource shares**, choose the shared Lake Formation resource. The **Status** should be **Pending**.

3. Choose **Action** and **Grant**.
4. Confirm the resource details, and choose **Accept resource share**.

   At this point, the consumer account data lake administrator should be able to find the shared resource on the Lake Formation console (https://console.aws.amazon.com/lakeformation/) under **Data Catalog**, **Databases**.

Create a resource link for the shared table

- Follow the instructions in Step 3: Implement cross-account sharing using the tag-based access control method (step 6) to create a resource link for a shared table. Name the resource link `amazon_reviews_table_named_resource_resource_link`. Create the resource link in the database `lakeformation_tutorial_cross_account_database_consumer`.

Grant data permission for the shared table to the consumer

To grant data permission for the shared table to the consumer, complete the following steps:

1. On the Lake Formationconsole (https://console.aws.amazon.com/lakeformation/), under **Permissions**, **Data lake permissions**, choose **Grant**.
2. For **Principals**, choose **IAM users and roles**, and choose the user `DataAnalyst`.
3. For **LF-Tags or catalog resources**, choose **Named data catalog resources**.
4. Under **Databases**, choose the database `lakeformation_tutorial_cross_account_database_named_resource`. If you don't see the database on the drop-down list, choose **Load more**.
5. Under **Tables**, choose the table `amazon_reviews_table_named_resource`.
6. For **Table and column permissions**, select **Select** and **Describe** under **Table permissions**.
7. Choose **Grant**.

Grant data permission for the resource link to the consumer

In addition to granting the data lake user permission to access the shared table, you also need to grant the data lake user permission to access the resource link.

1. On the Lake Formation console (https://console.aws.amazon.com/lakeformation/), under **Permissions**, **Data lake permissions**, choose **Grant**.

2. For **Principals**, choose **IAM users and roles**, and choose the user `DataAnalyst`.
3. For **LF-Tags or catalog resources**, choose **Named data catalog resources**.
4. Under **Databases**, choose the database `lakeformation_tutorial_cross_account_database_consumer`. If you don't see the database on the drop-down list, choose **Load more**.
5. Under **Tables**, choose the table `amazon_reviews_table_named_resource_resource_link`.
6. For **Resource link permissions**, select **Describe** under **Resource link permissions**.
7. Choose **Grant**.

   At this point, the data analyst user in the consumer account should be able to find the database and resource link, and query the shared table via the Athena console.

   If not, confirm if the following are properly configured:

   - The resource link is created for the shared table
   - You granted the user access to the table shared by the producer account
   - You granted the user access to the resource link and database for which the resource link is created

Step 5: Clean up AWS resources

To prevent unwanted charges to your AWS account, you can delete the AWS resources that you used for this tutorial.

1. Sign into Lake Formation console at https://console.aws.amazon.com/lakeformation/ using the producer account and delete or change the following:

   - AWS Resource Access Manager resource share
   - Lake Formation tags
   - CloudFormation stack
   - Lake Formation settings
   - AWS Glue Data Catalog

2. Sign into Lake Formation console at https://console.aws.amazon.com/lakeformation/ using the consumer account and delete or change the following:

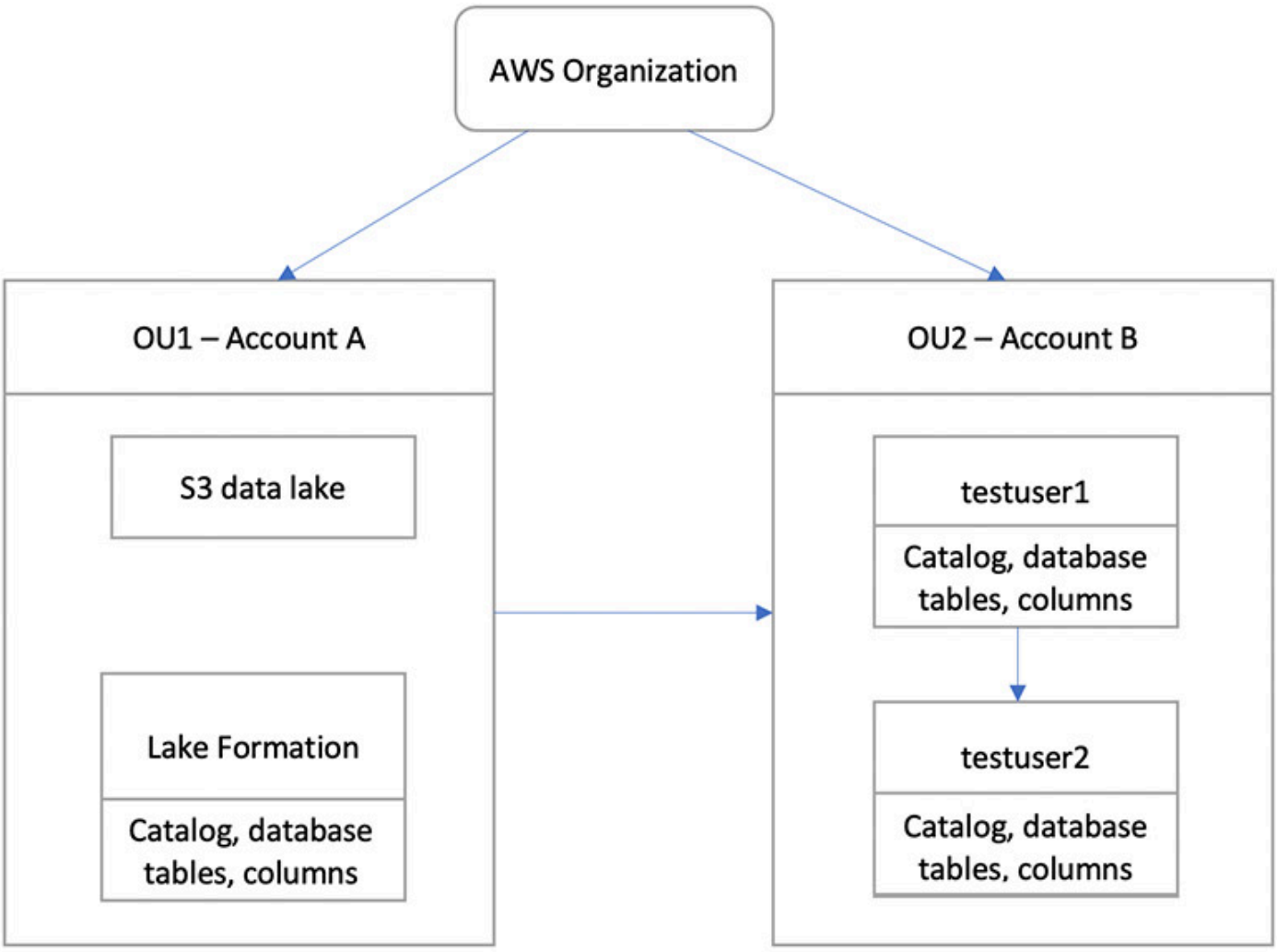
- Lake Formation tags
- CloudFormation stack

Sharing a data lake using Lake Formation fine-grained access control

This tutorial provides step-by-step instructions on how you can quickly and easily share datasets using Lake Formation when managing multiple AWS accounts with AWS Organizations. You define granular permissions to control access to sensitive data.

The following procedures also show how a data lake administrator of Account A can provide fine-grained access for Account B, and how a user in Account B, acting as a data steward, can grant fine-grained access to the shared table for other users in their account. Data stewards within each account can independently delegate access to their own users, giving each team or lines of business (LOB) autonomy.

The use case assumes you are using AWS Organizations to manage your AWS accounts. The user of Account A in one organizational unit (OU1) grants access to users of Account B in OU2. You can use the same approach when not using Organizations, such as when you only have a few accounts. The following diagram illustrates the fine-grained access control of datasets in a data lake. The data lake is available in the Account A. The data lake administrator of Account A provides fine-grained access for Account B. The diagram also shows that a user of Account B provides column-level access of the Account A data lake table to another user in Account B.



Topics

- Intended audience
- Prerequisites
- Step 1: Provide fine-grained access to another account
- Step 2: Provide fine-grained access to a user in the same account

Intended audience

This tutorial is intended for data stewards, data engineers, and data analysts. The following table lists the roles that are used in this tutorial:

Role	Description
IAM administrator	User who has the AWS managed policy: `AdministratorAccess` .
Data lake administrator	User who has the AWS managed policy: `AWSLakeFormationDataAdmin` attached to the role.
Data analyst	User who has the AWS managed policy: `AmazonAthenaFullAccess` attached.

Prerequisites

Before you start this tutorial, you must have an AWS account that you can use to sign in as an administrative user with correct permissions. For more information, see Complete initial AWS configuration tasks.

The tutorial assumes that you are familiar with IAM. For information about IAM, see the IAM User Guide.

You need the following resources for this tutorial:

- Two organizational units:
  - OU1 – Contains Account A
  - OU2 – Contains Account B
- An Amazon S3 data lake location (bucket) in Account A.
- A data lake administrator user in Account A. You can create a data lake administrator using the Lake Formation console (https://console.aws.amazon.com/lakeformation/) or the `PutDataLakeSettings` operation of the Lake Formation API.
- Lake Formation configured in Account A, and the Amazon S3 data lake location registered with Lake Formation in Account A.
- Two users in Account B with the following IAM managed policies:
  - testuser1 – has the AWS managed policies `AWSLakeFormationDataAdmin` attached.
  - testuser2 – Has the AWS managed policy `AmazonAthenaFullAccess` attached.

- A database testdb in the Lake Formation database for Account B.

Step 1: Provide fine-grained access to another account

Learn how a data lake administrator of Account A provides fine-grained access for Account B.

Grant fine-grained access to another account

1. Sign into AWS Management Console at https://console.aws.amazon.com/connect/ in Account A as a data lake administrator.
2. Open the Lake Formation console (https://console.aws.amazon.com/lakeformation/), and choose **Get started**.
3. in the navigation pane, choose **Databases**.
4. Choose **Create database**.
5. In the **Database** details section, select **Database**.
6. For **Name**, enter a name (for this tutorial, we use `sampledb01`).
7. Make sure that **Use only IAM access control for new tables in this database** is not selected. Leaving this unselected allows us to control access from Lake Formation.
8. Choose **Create database**.
9. On the **Databases** page, choose your database `sampledb01`.
10. On the **Actions** menu, choose **Grant**.
11. In the **Grant permissions** section, select **External account**.
12. For AWS account ID or AWS organization ID, enter the account ID for Account B in OU2.
13. For **Table**, choose the table you want Account B to have access to (for this post, we use table `acc_a_area`). Optionally, you can grant access to columns within the table, which we do in this post.
14. For **Include columns**¸ choose the columns you want Account B to have access to (for this post, we grant permissions to type, name, and identifiers).
15. For **Columns**, choose **Include columns**.
16. For **Table permissions**, select **Select**.
17. For **Grantable permissions**, select **Select**. Grantable permissions are required so admin users in Account B can grant permissions to other users in Account B.

18. Choose **Grant**.
19. In the navigation pane, choose **Tables**.
20. You could see one active connection in the AWS accounts and AWS organizations with access section.

Create a resource link

Integrated services like Amazon Athena can not directly access databases or tables across accounts. Hence, you need to create a resource link so that Athena can access resource links in your account to databases and tables in other accounts. Create a resource link to the table (`acc_a_area`) so Account B users can query its data with Athena.

1. Sign into the AWS console at https://console.aws.amazon.com/connect/ in Account B as `testuser1`.
2. On the Lake Formation console (https://console.aws.amazon.com/lakeformation/), in the navigation pane, choose **Tables**. You should see the tables that Account A has provided access.
3. Choose the table `acc_a_area`.
4. On the **Actions** menu, choose **Create resource link**.
5. For **Resource link name**, enter a name (for this tutorial, `acc_a_area_rl`).
6. For **Database**, choose your database (`testdb`).
7. Choose **Create**.
8. In the navigation pane, choose **Tables**.
9. Choose the table `acc_b_area_rl`.
10. On the **Actions** menu, choose **View data**.

    You are redirected to the Athena console, where you should see the database and table.

    You can now run a query on the table to see the column value for which access was provided to testuser1 from Account B.

Step 2: Provide fine-grained access to a user in the same account

This section shows how a user in Account B (`testuser1`), acting as a data steward, provides fine-grained access to another user in the same account (`testuser2`) to the column name in the shared table `aac_b_area_rl`.

Grant fine-grained access to a user in the same account

1. Sign into the AWS console at https://console.aws.amazon.com/connect/ in Account B as `testuser1`.
2. On the Lake Formation console, in the navigation pane, choose **Tables**.

   You can grant permissions on a table through its resource link. To do so, on the **Tables** page, select the resource link `acc_b_area_rl`, and on the **Actions** menu, choose **Grant on target**.
3. In the **Grant permissions** section, select **My account**.
4. For **IAM users and roles**, choose the user `testuser2`.
5. For **Column**, choose the column name.
6. For **Table permissions**, select **Select**.
7. Choose **Grant**.

   When you create a resource link, only you can view and access it. To permit other users in your account to access the resource link, you need to grant permissions on the resource link itself. You need to grant **DESCRIBE** or **DROP** permissions. On the **Tables page**, select your table again and on the **Actions** menu, choose **Grant**.
8. In the **Grant permissions** section, select **My account**.
9. For **IAM users and roles**, select the user `testuser2`.
10. For **Resource link permissions**, select **Describe**.
11. Choose **Grant**.
12. Sign into the AWS console in Account B as `testuser2`.

    On the Athena console (https://console.aws.amazon.com/athena/), you should see the database and table `acc_b_area_rl`. You can now run a query on the table to see the column value that `testuser2` has access to.

Onboarding to Lake Formation permissions

AWS Lake Formation uses the AWS Glue Data Catalog (Data Catalog) to store metadata for the Amazon S3 data lakes and external data sources such as Amazon Redshift in the form of catalogs, databases and tables. Metadata in the Data Catalog is organized in a three-level data hierarchy comprising catalogs, databases, and tables. It organizes data from various sources into logical containers called catalogs. Databases are collections of tables. The Data Catalog also contains resource links, which are links to shared databases and tables in external accounts, and are used for cross-account access to data in the data lake. Each AWS account has one Data Catalog per AWS Region.

Lake Formation provides a relational database management system (RDBMS) permissions model to grant or revoke access to catalogs, databases, tables, and columns in the Data Catalog with underlying data in Amazon S3.

Before you learn about the details of the Lake Formation permissions model, it is helpful to review the following background information:

- Data lakes managed by Lake Formation reside in designated locations in Amazon Simple Storage Service (Amazon S3). The Data Catalog also contains catalog objects. Each catalog represents data from sources like Amazon Redshift data warehouses, Amazon DynamoDB databases, and third-party data sources such as Snowflake, MySQL, and over 30 external data sources, which are integrated through federated connectors.
- Lake Formation maintains a Data Catalog that contains metadata about source data to be imported into your data lakes, such as data in logs and relational databases, and about data in your data lakes in Amazon S3. The Data Catalog also contains metadata about data from external data sources other than Amazon S3. The metadata is organized as catalogs, databases and tables. Metadata tables contain schema, location, partitioning, and other information about the data that they represent. Metadata databases are collections of tables.
- The Lake Formation Data Catalog is the same Data Catalog used by AWS Glue. You can use AWS Glue crawlers to create Data Catalog tables, and you can use AWS Glue extract, transform, and load (ETL) jobs to populate the underlying data in your data lakes.
- The catalogs, databases, and tables in the Data Catalog are referred to as *Data Catalog resources*. Tables in the Data Catalog are referred to as *metadata tables* to distinguish them from tables in data sources or tabular data in Amazon S3. The data that the metadata tables point to in Amazon S3 or in data sources is referred to as *underlying data*.

- A *principal* is a user or role, an Amazon Quick user or group, a user or group that authenticates with Lake Formation through a SAML provider, or for cross-account access control, an AWS account ID, organization ID, or organizational unit ID.
- AWS Glue crawlers create metadata tables, but you can also manually create metadata tables with the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI). When creating a metadata table, you must specify a location. When you create a database, the location is optional. Table locations can be Amazon S3 locations or data source locations such as an Amazon Relational Database Service (Amazon RDS) database. Database locations are always Amazon S3 locations.
- Services that integrate with Lake Formation, such as Amazon Athena and Amazon Redshift, can access the Data Catalog to obtain metadata and to check authorization for running queries. For a complete list of integrated services, see AWS service integrations with Lake Formation.

Topics

- Overview of Lake Formation permissions
- Lake Formation personas and IAM permissions reference
- Changing the default settings for your data lake
- Implicit Lake Formation permissions
- Lake Formation permissions reference
- Integrating IAM Identity Center
- Adding an Amazon S3 location to your data lake
- Hybrid access mode
- Creating objects in the AWS Glue Data Catalog
- Importing data using workflows in Lake Formation

Overview of Lake Formation permissions

There are two main types of permissions in AWS Lake Formation:

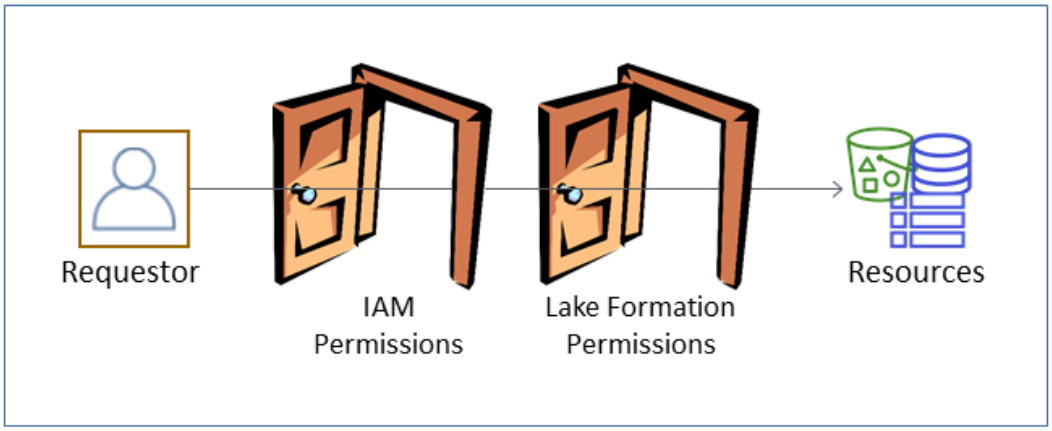
- Metadata access – Permissions on Data Catalog resources (*Data Catalog permissions*).

  These permissions enable principals to create, read, update, and delete metadata databases and tables in the Data Catalog.

- Underlying data access – Permissions on locations in Amazon Simple Storage Service (Amazon S3) (*data access permissions* and *data location permissions*).
  - Data lake permissions enable principals to read and write data to *underlying* Amazon S3 locations—data pointed to by Data Catalog resources.
  - Data location permissions enable principals to create and alter metadata databases and tables that point to specific Amazon S3 locations.

For both areas, Lake Formation uses a combination of Lake Formation permissions and AWS Identity and Access Management (IAM) permissions. The IAM permissions model consists of IAM policies. The Lake Formation permissions model is implemented as DBMS-style GRANT/REVOKE commands, such as `Grant SELECT on` *`tableName`* `to` *`userName`*.

When a principal makes a request to access Data Catalog resources or underlying data, for the request to succeed, it must pass permission checks by both IAM and Lake Formation.



Lake Formation permissions control access to Data Catalog resources, Amazon S3 locations, and the underlying data at those locations. IAM permissions control access to the Lake Formation and AWS Glue APIs and resources. So although you might have the Lake Formation permission to create a metadata table in the Data Catalog (`CREATE_TABLE`), your operation fails if you don't have the IAM permission on the `glue:CreateTable` API. (Why a `glue:` permission? Because Lake Formation uses the AWS Glue Data Catalog.)

> **Note**
>
> Lake Formation permissions apply only in the Region in which they were granted.

AWS Lake Formation requires that each principal (user or role) be authorized to perform actions on Lake Formation–managed resources. A principal is granted the necessary authorizations by the data lake administrator or another principal with the permissions to grant Lake Formation permissions.

When you grant a Lake Formation permission to a principal, you can optionally grant the ability to pass that permission to another principal.

You can use the Lake Formation API, the AWS Command Line Interface (AWS CLI), or the **Data permissions** and **Data locations** pages of the Lake Formation console to grant and revoke Lake Formation permissions.

Methods for fine-grained access control

With a data lake, the goal is to have fine-grained access control to data. In Lake Formation, this means fine-grained access control to Data Catalog resources and Amazon S3 locations. You can achieve fine-grained access control with one of the following methods.

Method	Lake Formation Permissions	IAM Permissions	Comments
Method 1	Open	Fine-grained	**This is the default method** for backward compatibility with AWS Glue. • *Open* means that the special permission `Super` is granted to the group `IAMAllowedPrincipals`, where `IAMAllowedPrincipals` is automatically created and includes any IAM users and roles that are allowed access to your Data Catalog resources by your IAM policies, and the `Super` permission enables a principal to perform every supported Lake Formation operation on the database or table on which it is granted. This effectively causes access to Data

Method	Lake Formation Permissions	IAM Permissions	Comments
			Catalog resources and Amazon S3 locations to be controlled solely by IAM policies. For more information, see Changing the default settings for your data lake and Upgrading AWS Glue data permissions to the AWS Lake Formation model. • *Fine-grained* means that IAM policies control all access to Data Catalog resources and to individual Amazon S3 buckets. On the Lake Formation console, this method appears as **Use only IAM access control**.

Method	Lake Formation Permissions	IAM Permissions	Comments
Method 2	Fine-grained	Coarse-grained	**This is the recommended method.** • *Fine-grained* access means granting limited Lake Formation permissions to individual principals on Data Catalog resources, Amazon S3 locations, and the underlying data in those locations. • *Coarse-grained* means broader permissions on individual operations and on access to Amazon S3 locations . For example, a coarse-grained IAM policy might include `"glue:*"` or `"glue:Create*"` rather than `"glue:CreateTables"` , leaving Lake Formation permissions to control whether or not a principal can create catalog objects. It also means giving principals access to the APIs that they need to do their work, but locking down other APIs and resources. For example, you might create an IAM policy that enables a principal to create Data Catalog resources and create and run workflows, but doesn't enable creation of AWS Glue connections or user-defined functions. See the examples later in this section.

Important

Be aware of the following:

- By default, Lake Formation has the **Use only IAM access control** settings enabled for compatibility with existing AWS Glue Data Catalog behavior. We recommend that you disable these settings after you transition to using Lake Formation permissions. For more information, see Changing the default settings for your data lake.
- Data lake administrators and database creators have implicit Lake Formation permissions that you must understand. For more information, see Implicit Lake Formation permissions.

Metadata access control

For access control for Data Catalog resources, the following discussion assumes fine-grained access control with Lake Formation permissions and coarse-grained access control with IAM policies.

There are two distinct methods for granting Lake Formation permissions on Data Catalog resources:

- **Named resource access control** – With this method, you grant permissions on specific databases or tables by specifying database or table names. The grants have this form:

  Grant *permissions* to *principals* on *resources* [with grant option].

  With the grant option, you can allow the grantee to grant the permissions to other principals.
- **Tag-based access control** – With this method, you assign one or more LF-Tags to Data Catalog databases, tables, and columns, and grant permissions on one or more LF-Tags to principals. Each LF-Tag is a key-value pair, such as `department=sales`. A principal that has LF-Tags that match the LF-Tags on a Data Catalog resource can access that resource. This method is recommended for data lakes with a large number of databases and tables. It's explained in detail in Lake Formation tag-based access control.

The permissions that a principal has on a resource is the union of the permissions granted by both the methods.

The following table summarizes the available Lake Formation permissions on Data Catalog resources. The column headings indicate the resource on which the permission is granted.

Catalog	Database	Table
CREATE_DATABASE	CREATE_TABLE	ALTER
	ALTER	DROP
	DROP	DESCRIBE
	DESCRIBE	SELECT*
		INSERT*
		DELETE*

For example, the CREATE_TABLE permission is granted on a database. This means that the principal is allowed to create tables in that database.

The permissions with an asterisk (*) are granted on Data Catalog resources, but they apply to the underlying data. For example, the DROP permission on a metadata table enables you to drop the table from the Data Catalog. However, the DELETE permission granted on the same table enables you to delete the table's underlying data in Amazon S3, using, for example, a SQL DELETE statement. With these permissions, you also can view the table on the Lake Formation console and retrieve information about the table with the AWS Glue API. Thus, SELECT, INSERT, and DELETE are both Data Catalog permissions and data access permissions.

When granting SELECT on a table, you can add a filter that includes or excludes one or more columns. This permits fine-grained access control on metadata table columns, limiting the columns that users of integrated services can see when running queries. This capability is not available using just IAM policies.

There is also a special permission named Super. The Super permission enables a principal to perform every supported Lake Formation operation on the database or table on which it is granted. This permission can coexist with the other Lake Formation permissions. For example, you can grant Super, SELECT, and INSERT on a metadata table. The principal can perform all supported actions on the table, and when you revoke Super, the SELECT and INSERT permissions remain.

For details on each permission, see Lake Formation permissions reference.

> **Important**
>
> To be able to see a Data Catalog table created by another user, you must be granted at least one Lake Formation permission on the table. If you are granted at least one permission on the table, you can also see the table's containing database.

You can grant or revoke Data Catalog permissions using the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI). The following is an example of an AWS CLI command that grants the user `datalake_user1` permission to create tables in the `retail` database.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "CREATE_TABLE" --resource '{ "Database": {"Name":"retail"}}'
```

The following is an example of a coarse-grained access control IAM policy that complements fine-grained access control with Lake Formation permissions. It permits all operations on any metadata database or table.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:*Database*",
                "glue:*Table*",
                "glue:*Partition*"
            ],
            "Resource": "*"
        }
    ]
}
```

The next example is also coarse-grained but somewhat more restrictive. It permits read-only operations on all metadata databases and tables in the Data Catalog in the designated account and Region.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTables",
                "glue:SearchTables",
                "glue:GetTable",
                "glue:GetDatabase",
                "glue:GetDatabases"
            ],
            "Resource": "arn:aws:glue:us-east-1:111122223333:*"
        }
    ]
}
```

Compare these policies to the following policy, which implements IAM-based fine-grained access control. It grants permissions only on a subset of tables in the customer relationship management (CRM) metadata database in the designated account and Region.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTables",
                "glue:SearchTables",
                "glue:GetTable",
                "glue:GetDatabase",
                "glue:GetDatabases"
```

```
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:catalog",
                "arn:aws:glue:us-east-1:111122223333:database/CRM",
                "arn:aws:glue:us-east-1:111122223333:table/CRM/P*"
            ]
        }
    ]
}
```

For more examples of coarse-grained access control policies, see Lake Formation personas and IAM permissions reference.

Underlying data access control

When an integrated AWS service requests access to data in an Amazon S3 location that is access-controlled by AWS Lake Formation, Lake Formation supplies temporary credentials to access the data.

To enable Lake Formation to control access to underlying data at an Amazon S3 location, you *register* that location with Lake Formation.

After you register an Amazon S3 location, you can start granting the following Lake Formation permissions:

- Data access permissions (`SELECT`, `INSERT`, and `DELETE`) on Data Catalog tables that point to that location.
- Data location permissions on that location.

Lake Formation data location permissions control the ability to create Data Catalog resources that point to particular Amazon S3 locations. Data location permissions provide an extra layer of security to locations within the data lake. When you grant the `CREATE_TABLE` or `ALTER` permission to a principal, you also grant data location permissions to limit the locations for which the principal can create or alter metadata tables.

Amazon S3 locations are buckets or prefixes under a bucket, but not individual Amazon S3 objects.

You can grant data location permissions to a principal by using the Lake Formation console, the API, or the AWS CLI. The general form of a grant is as follows:

```
grant DATA_LOCATION_ACCESS to principal on S3 location [with grant option]
```

If you include `with grant option`, the grantee can grant the permissions to other principals.

Recall that Lake Formation permissions always work in combination with AWS Identity and Access Management (IAM) permissions for fine-grained access control. For read/write permissions on underlying Amazon S3 data, IAM permissions are granted as follows:

When you register a location, you specify an IAM role that grants read/write permissions on that location. Lake Formation assumes that role when supplying temporary credentials to integrated AWS services. A typical role might have the following policy attached, where the registered location is the bucket `awsexamplebucket`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

Lake Formation provides a service-linked role that you can use during registration to automatically create policies like this. For more information, see Using service-linked roles for Lake Formation.

Therefore, registering an Amazon S3 location grants the required IAM `s3:` permissions on that location, where the permissions are specified by the role used to register the location.

> **Important**
>
> Avoid registering an Amazon S3 bucket that has **Requester pays** enabled. For buckets registered with Lake Formation, the role used to register the bucket is always viewed as the requester. If the bucket is accessed by another AWS account, the bucket owner is charged for data access if the role belongs to the same account as the bucket owner.

For read/write access to underlying data, in addition to Lake Formation permissions, principals also need the `lakeformation:GetDataAccess` IAM permission. With this permission, Lake Formation grants the request for temporary credentials to access the data.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lakeformation:GetDataAccess",
            "Resource": "*"
        }
    ]
}
```

In the above policy, you must set the Resource parameter to '*' (all. Specifying any other resource for this permission is not supported. This configuration ensures that Lake Formation can manage data access across your entire data lake environment efficiently.

> **Note**
>
> Amazon Athena requires the user to have the `lakeformation:GetDataAccess` permission. Other integrated services require their underlying execution role to have the `lakeformation:GetDataAccess` permission.

This permission is included in the suggested policies in the Lake Formation personas and IAM permissions reference.

To summarize, to enable Lake Formation principals to read and write underlying data with access controlled by Lake Formation permissions:

- Register the Amazon S3 locations that contain the data with Lake Formation.
- Principals who create Data Catalog tables that point to underlying data locations must have data location permissions.
- Principals who read and write underlying data must have Lake Formation data access permissions on the Data Catalog tables that point to the underlying data locations.
- Principals who read and write underlying data must have the `lakeformation:GetDataAccess` IAM permission when the underlying data location is registered with Lake Formation.

> **Note**
>
> The Lake Formation permissions model doesn't prevent access to Amazon S3 locations through the Amazon S3 API or console if you have access to them through IAM or Amazon S3 policies. You can attach IAM policies to principals to block this access.

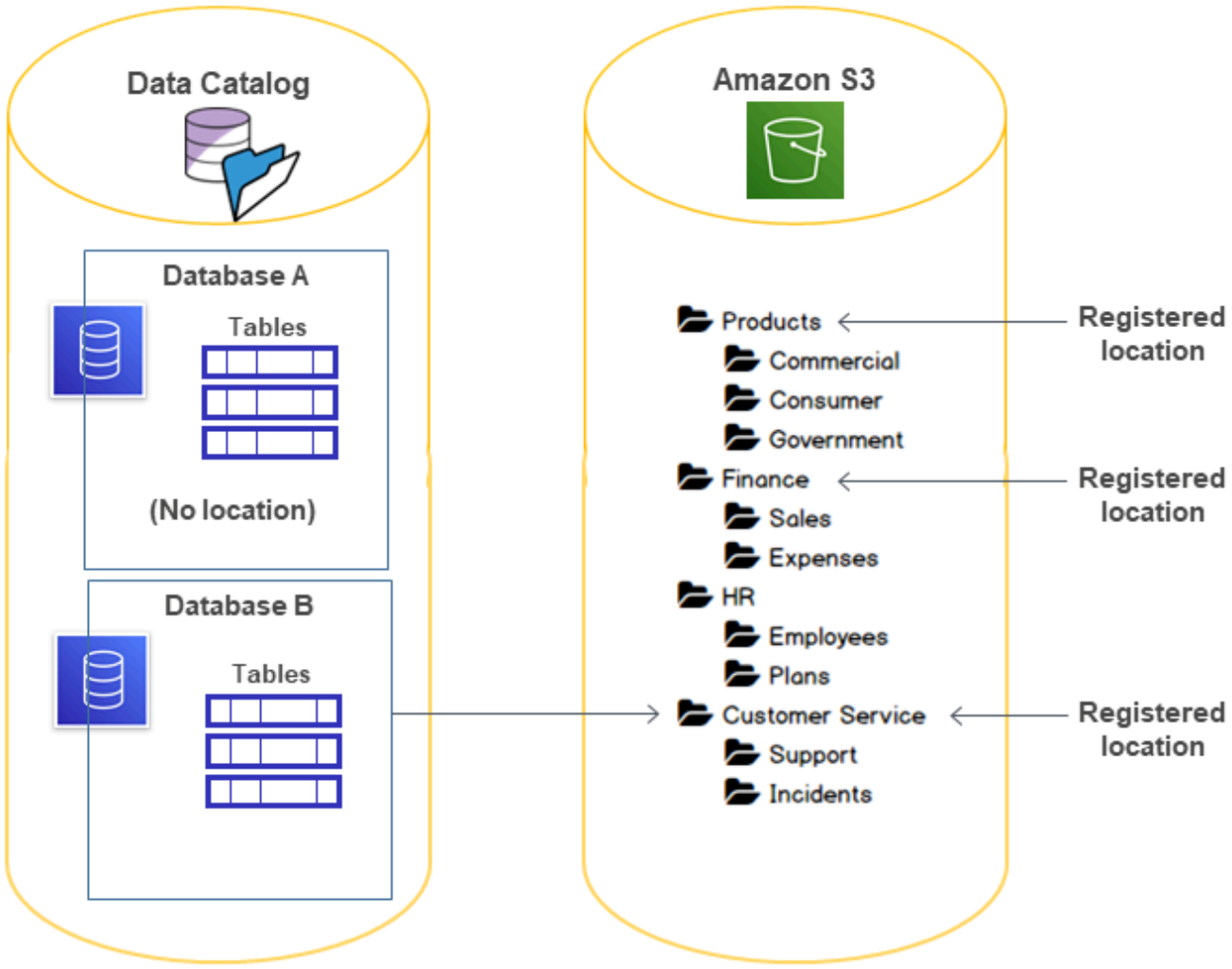
More on data location permissions

Data location permissions govern the outcome of create and update operations on Data Catalog databases and tables. The rules are as follows:

- A principal must have explicit or implicit data location permissions on an Amazon S3 location to create or update a database or table that specifies that location.
- The explicit permission `DATA_LOCATION_ACCESS` is granted using the console, API, or AWS CLI.

- Implicit permissions are granted when a database has a location property that points to a registered location, the principal has the `CREATE_TABLE` permission on the database, and the principal tries to create a table at that location or a child location.
- If a principal is granted data location permissions on a location, the principal has data location permissions on all child locations.
- A principal does not need data location permissions to perform read/write operations on the underlying data. It is sufficient to have the `SELECT` or `INSERT` data access permissions. Data location permissions apply only to creating Data Catalog resources that point to the location.
- Partition operations (`CreatePartition`, `BatchCreatePartition`, `UpdatePartition`) do not require data location permissions when the partition location is a child of the table location already registered with Lake Formation.

Consider the scenario shown in the following diagram.



In this diagram:

- The Amazon S3 buckets `Products`, `Finance`, and `Customer Service` are registered with Lake Formation.
- `Database A` has no location property, and `Database B` has a location property that points to the `Customer Service` bucket.
- User `datalake_user` has `CREATE_TABLE` on both databases.
- User `datalake_user` has been granted data location permissions only on the `Products` bucket.

The following are the results when user `datalake_user` tries to create a catalog table in a particular database at a particular location.

Location where `datalake_user` tries to create a table

Database and Location	Succeeds or Fails	Reason
Database A at `Finance/Sales`	Fails	No data location permission
Database A at `Products`	Succeeds	Has data location permission
Database A at `HR/Plans`	Succeeds	Location is not registered
Database B at `Customer Service/Incidents`	Succeeds	Database has location property at `Customer Service`

For more information, see the following:

- Adding an Amazon S3 location to your data lake
- Lake Formation permissions reference
- Lake Formation personas and IAM permissions reference

Lake Formation personas and IAM permissions reference

This section lists some suggested Lake Formation personas and their suggested AWS Identity and Access Management (IAM) permissions. For information about Lake Formation permissions, see the section called "Lake Formation permissions reference".

AWS Lake Formation personas

The following table lists the suggested AWS Lake Formation personas.

Lake Formation Personas

Persona	Description
IAM administrator (superuser)	(Required) User who can create IAM users and roles. Has the `AdministratorAccess` AWS managed policy. Has all

Persona	Description
	permissions on all Lake Formation resources. Can add data lake administrators. Cannot grant Lake Formation permissions if not also designated a data lake administrator.
Data lake administrator	(Required) User who can register Amazon S3 locations, access the Data Catalog, create databases, create and run workflows , grant Lake Formation permissions to other users, and view AWS CloudTrail logs. Has fewer IAM permissions than the IAM administrator, but enough to administer the data lake. Cannot add other data lake administrators.
Read only administrator	(Optional) User who can view principals, Data Catalog resources, permissions, and AWS CloudTrail logs, without the permissions to make updates.
Data engineer	(Optional) User who can create databases, create and run crawlers and workflows, and grant Lake Formation permissio ns on the Data Catalog tables that the crawlers and workflows create. We recommend that you make all data engineers database creators. For more information, see Creating a database.
Data analyst	(Optional) User who can run queries against the data lake using, for example, Amazon Athena. Has only enough permissions to run queries.
Workflow role	(Required) Role that runs a workflow on behalf of a user. You specify this role when you create a workflow from a blueprint.

Note

In Lake Formation, data lake administrators added after database creation can grant permissions but don't automatically have data access permissions such as SELECT or DESCRIBE. Administrators who create databases receive SUPER permissions on those databases. This behavior is intentional—while all administrators can grant themselves necessary permissions, these permissions aren't automatically applied to pre-existing

resources. Therefore, administrators must explicitly grant themselves access to databases that existed before they were assigned admin privileges.

AWS managed policies for Lake Formation

You can grant the AWS Identity and Access Management (IAM) permissions that are required to work with AWS Lake Formation by using AWS managed policies and inline policies. The following AWS managed policies are available for Lake Formation.

AWS managed policy:AWSLakeFormationDataAdmin

AWSLakeFormationDataAdmin policy grants administrative access to AWS Lake Formation and related services such as AWS Glue to manage data lakes.

You can attach `AWSLakeFormationDataAdmin` to your users, groups, and roles.

Permission details

- `CloudTrail` – Allows principals to view AWS CloudTrail logs. This is required to review any errors in the set up of the data lake.
- `Glue` – Allows principals to view, create, and update metadata tables and databases in Data Catalog. This includes API operations that start with `Get`, `List`, `Create`, `Update`, `Delete`, and `Search`. This is required to manage the metadata of the data lake tables.
- `IAM` – Allows principals to retrieve information about IAM users, roles, and policies attached to the roles. This is required for the data admin to review and list IAM users and roles to grant Lake Formation permissions.
- `Lake Formation` – Grants data lake admins required Lake Formation permissions to manage data lakes.
- S3 – Allows principals to retrieve information about Amazon S3 buckets and their locations in order to set up the data location for data lakes.

```
"Statement": [
        {
            "Sid": "AWSLakeFormationDataAdminAllow",
            "Effect": "Allow",
            "Action": [
                "lakeformation:*",
```

```
                "cloudtrail:DescribeTrails",
                "cloudtrail:LookupEvents",
                "glue:CreateCatalog",
  "glue:UpdateCatalog",
                "glue:DeleteCatalog",
  "glue:GetCatalog",
         "glue:GetCatalogs",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:CreateDatabase",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:GetConnections",
                "glue:SearchTables",
                "glue:GetTable",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTableVersions",
                "glue:GetPartitions",
                "glue:GetTables",
                "glue:ListWorkflows",
                "glue:BatchGetWorkflows",
                "glue:DeleteWorkflow",
                "glue:GetWorkflowRuns",
                "glue:StartWorkflowRun",
                "glue:GetWorkflow",
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets",
                "s3:GetBucketAcl",
                "iam:ListUsers",
                "iam:ListRoles",
                "iam:GetRole",
                "iam:GetRolePolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AWSLakeFormationDataAdminDeny",
            "Effect": "Deny",
            "Action": [
                "lakeformation:PutDataLakeSettings"
            ],
```

```
                "Resource": "*"
            }
        ]
}
```

> **Note**
>
> The `AWSLakeFormationDataAdmin` policy does not grant every required permission for data lake administrators. Additional permissions are needed to create and run workflows and register locations with the service linked role `AWSServiceRoleForLakeFormationDataAccess`. For more information, see Create a data lake administrator and Using service-linked roles for Lake Formation.

AWS managed policy:AWSLakeFormationCrossAccountManager

AWSLakeFormationCrossAccountManager policy provides cross account access to AWS Glue resources via Lake Formation, and grants read access to other required services such as AWS Organizations and AWS RAM.

You can attach `AWSLakeFormationCrossAccountManager` to your users, groups, and roles.

Permission details

This policy includes the following permissions.

- `Glue` – Allows principals to set or delete the Data Catalog resource policy for access control.
- `Organizations` – Allows principals to retrieve account and organizational unit (OU) information for an organization.
- `ram:CreateResourceShare` – Allows principals to create a resource share.
- `ram:UpdateResourceShare` –Allows principals to modify some properties of the specified resource share.
- `ram:DeleteResourceShare` – Allows principals to delete the specified resource share.
- `ram:AssociateResourceShare` – Allows principals to add the specified list of principals and list of resources to a resource share.
- `ram:DisassociateResourceShare` – Allows principals to remove the specified principals or resources from participating in the specified resource share.

- `ram:GetResourceShares`– Allows principals to retrieve details about the resource shares that you own or that are shared with you.
- `ram:RequestedResourceType` – Allows principals to retrieve the resource type (database, table or catalog).
- `AssociateResourceSharePermission` – Allows principals to add or replace the AWS RAM permission for a resource type included in a resource share. You can have exactly one permission associated with each resource type in the resource share.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Sid": "AllowCreateResourceShare",
            "Effect": "Allow",
            "Action": [
                "ram:CreateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": {
                    "ram:RequestedResourceType": [
                        "glue:Table",
                        "glue:Database",
                        "glue:Catalog"
                    ]
                }
            }
        },
        {
            "Sid": "AllowManageResourceShare",
            "Effect": "Allow",
            "Action": [
                "ram:UpdateResourceShare",
                "ram:DeleteResourceShare",
                "ram:AssociateResourceShare",
                "ram:DisassociateResourceShare",
                "ram:GetResourceShares"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringLike": {
                    "ram:ResourceShareName": [
                        "LakeFormation*"
                    ]
                }
            }
        },
        {
            "Sid": "AllowManageResourceSharePermissions",
            "Effect": "Allow",
            "Action": [
                "ram:AssociateResourceSharePermission"
            ],
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "ram:PermissionArn": [
                        "arn:aws:ram::aws:permission/AWSRAMLFEnabled*"
                    ]
                }
            }
        },
        {
            "Sid": "AllowXAcctManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "glue:PutResourcePolicy",
                "glue:DeleteResourcePolicy",
                "organizations:DescribeOrganization",
                "organizations:DescribeAccount",
                "ram:Get*",
                "ram:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowOrganizationsPermissions",
            "Effect": "Allow",
            "Action": [
                "organizations:ListRoots",
                "organizations:ListAccountsForParent",
                "organizations:ListOrganizationalUnitsForParent"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

AWS managed policy:AWSGlueConsoleFullAccess

AWSGlueConsoleFullAccess policy grants full access to AWS Glue resources when an identity that the policy is attached to uses the AWS Management Console. If you follow the naming convention for resources specified in this policy, users have full console capabilities. This policy is typically attached to users of the AWS Glue console.

In addition, AWS Glue and Lake Formation assume the service role `AWSGlueServiceRole` to allow access to related services, including Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), and Amazon CloudWatch.

AWS managed policy:LakeFormationDataAccessServiceRolePolicy

This policy is attached to a service-linked role named `ServiceRoleForLakeFormationDataAccess` that allows the service to perform actions on resources at your request. You can't attach this policy to your IAM identities.

This policy allows the Lake Formation integrated AWS services such as Amazon Athena or Amazon Redshift to use the service-linked role to discover Amazon S3 resources.

For more information see, Using service-linked roles for Lake Formation.

Permission details

This policy includes the following permission.

- `s3:ListAllMyBuckets` – Returns a list of all buckets owned by the authenticated sender of the request.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
```

```
  {
   "Sid": "LakeFormationDataAccessServiceRolePolicy",
   "Effect": "Allow",
   "Action": [
    "s3:ListAllMyBuckets"
   ],
   "Resource": [
    "arn:aws:s3:::*"
   ]
  }
 ]
}
```

Lake Formation updates to AWS managed policies

View details about updates to AWS managed policies for Lake Formation since this service began tracking these changes.

Change	Description	Date
Lake Formation updated `AWSLakeFormationCrossAccountManager` policy.	Lake Formation enhanced the AWSLakeFormationCrossAccountManager policy by replacing the `StringLike` condition operator with the `ArnLike` operator that allows IAM to perform the ARN format check.	January, 2025
Lake Formation updated `AWSLakeFormationDataAdmin` policy.	Lake Formation enhanced the AWSLakeFormationDataAdmin policy by adding the following AWS Glue Data Catalog CRUD APIs as part of the multi-catalog feature. • glue:CreateCatalog • glue:UpdateCatalog • glue:DeleteCatalog • glue:GetCatalog	December, 2024

Change	Description	Date
	• glue:GetCatalogs This managed policy change is to ensure that the Lake Formation administrator persona by default has IAM permission on these new operations.	
Lake Formation updated `AWSLakeFormationCrossAccountManager` policy.	Lake Formation enhanced the AWSLakeFormationCrossAccountManager policy by adding Sid elements to the policy statement.	March, 2024
Lake Formation updated `AWSLakeFormationDataAdmin` policy.	Lake Formation enhanced the AWSLakeFormationDataAdmin policy by adding a Sid element to the policy statement and removing a redundant action.	March, 2024
Lake Formation updated `LakeFormationDataAccessServiceRolePolicy` policy.	Lake Formation enhanced the LakeFormationDataAccessServiceRolePolicy policy by adding a Sid element to the policy statement.	February, 2024
Lake Formation updated `AWSLakeFormationCrossAccountManager` policy.	Lake Formation enhanced the AWSLakeFormationCrossAccountManager policy by adding a new permission to enable cross-account data sharing in hybrid access mode.	October, 2023

Change	Description	Date
Lake Formation updated `AWSLakeFormationCrossAccountManager` policy.	Lake Formation enhanced the AWSLakeFormationCrossAccountManager policy to create only one resource share per recipient account when the a resource is first shared. All resources shared thereafter with the same account are attached to the same resource share.	May 6, 2022
Lake Formation started tracking changes.	Lake Formation started tracking changes for its AWS managed policies.	May 6, 2022

Personas suggested permissions

The following are the suggested permissions for each persona. The IAM administrator is not included because that user has all permissions on all resources.

Topics

- Data lake administrator permissions
- Read only administrator permissions
- Data engineer permissions
- Data analyst permissions
- Workflow role permissions

Data lake administrator permissions

> **Important**
>
> In the following policies, replace *<account-id>* with a valid AWS account number, and replace *<workflow_role>* with the name of a role that has permissions to run a workflow, as defined in Workflow role permissions.

Policy Type	Policy
AWS managed policies	• `AWSLakeFormationDataAdmin` • `LakeFormationDataAccessServiceRolePolicy` (service-linked role policy) • `AWSGlueConsoleFullAccess` (Optional) • `CloudWatchLogsReadOnlyAccess` (Optional) • `AWSLakeFormationCrossAccountManager` (Optional) • `AmazonAthenaFullAccess` (Optional) For information about the optional AWS managed policies, see the section called "Create a data lake administrator".

Inline policy (for creating the Lake Formation service-linked role)	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": "iam:CreateServiceLinkedRole",
	"Resource": "*",
	"Condition": {
	"StringEquals": {
	"iam:AWSServiceName": "lakeformation.amazonaws.com"
	}
	}
	},
	{
	"Effect": "Allow",
	"Action": [
	"iam:PutRolePolicy"
	],
	"Resource": "arn:aws:iam:: *<account-id>* :role/aws-service-role/lakeformation.amazonaws.com/AWSServiceRoleForLakeFormationDataAccess"
	}
	]

Policy Type	Policy
	} (Optional) Inline policy (passrole policy for the workflow role). This is required only if the data lake administrator creates and runs workflows.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PassRolePermissions",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam:: 111122223333 :role/<workflow_role> "
            ]
        }
    ]
}
```

(Optional) Inline policy (if your account is granting or receiving cross-account Lake Formation permissions). This policy is for accepting or rejecting AWS RAM resource share invitations, and for enabling the granting of cross-account permissions to organizations. `ram:EnableSharingWithAwsOrganization` is required only for data lake administrators in the AWS Organizations management account.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ram:AcceptResourceShareInvitation",
                "ram:RejectResourceShareInvitation",
                "ec2:DescribeAvailabilityZones",
                "ram:EnableSharingWithAwsOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

Read only administrator permissions

Policy type	Policy
Inline policy (basic)	

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "lakeformation:GetEffectivePermissio
nsForPath",
            "lakeformation:ListPermissions",
            "lakeformation:ListDataCell
sFilter",
            "lakeformation:GetDataCellsFilter",
            "lakeformation:SearchDatabasesByLFTa
gs",
            "lakeformation:SearchTablesByLFTags"
,
            "lakeformation:GetLFTag",
            "lakeformation:ListLFTags",
            "lakeformation:GetResourceLFTags",
            "lakeformation:ListLakeFormationOpti
ns",
            "cloudtrail:DescribeTrails",
            "cloudtrail:LookupEvents",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:GetConnections",
            "glue:SearchTables",
            "glue:GetTable",
            "glue:GetTableVersions",
            "glue:GetPartitions",
            "glue:GetTables",
            "glue:GetWorkflow",
            "glue:ListWorkflows",
            "glue:BatchGetWorkflows",
            "glue:GetWorkflowRuns",
            "glue:GetWorkflow",
            "s3:ListBucket",
            "s3:GetBucketLocation",
```

Policy type	Policy

	"s3:ListAllMyBuckets",
	"s3:GetBucketAcl",
	"iam:ListUsers",
	"iam:ListRoles",
	"iam:GetRole",
	"iam:GetRolePolicy"
	],
	"Resource":"*"
	},
	{
	"Effect":"Deny",
	"Action":[
	"lakeformation:PutDataLakeSettings"
	],
	"Resource":"*"
	}
	]
	}

Data engineer permissions

> **Important**
>
> In the following policies, replace *<account-id>* with a valid AWS account number, and replace *<workflow_role>* with the name of the workflow role.

Policy Type	Policy
AWS managed policy	`AWSGlueConsoleFullAccess`
Inline policy (basic)	{ "Version":"2012-10-17", "Statement": [{ "Effect": "Allow", "Action": [

Policy Type	Policy
	"lakeformation:GetDataAccess", "lakeformation:GrantPermissions", "lakeformation:RevokePermissions", "lakeformation:BatchGrantPermissions", "lakeformation:BatchRevokePermissions", "lakeformation:ListPermissions", "lakeformation:AddLFTagsToResource", "lakeformation:RemoveLFTagsFromResource", "lakeformation:GetResourceLFTags", "lakeformation:ListLFTags", "lakeformation:GetLFTag", "lakeformation:SearchTablesByLFTags", "lakeformation:SearchDatabasesByLFTags", "lakeformation:GetWorkUnits", "lakeformation:GetWorkUnitResults", "lakeformation:StartQueryPlanning", "lakeformation:GetQueryState", "lakeformation:GetQueryStatistics"], "Resource": "*" }] }

Policy Type	Policy

Inline policy (for operations on governed tables, including operations within transactions)	{
	"Version":"2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"lakeformation:StartTransaction",
	"lakeformation:CommitTransaction",
	"lakeformation:CancelTransaction",
	"lakeformation:ExtendTransaction",
	"lakeformation:DescribeTransaction",
	"lakeformation:ListTransactions",
	"lakeformation:GetTableObjects",
	"lakeformation:UpdateTableObjects",
	"lakeformation:DeleteObjectsOnCancel
	"
	],
	"Resource": "*"
	}
	]
	}

Policy Type	Policy

Inline policy (for metadata access control using the Lake Formation tag-based access control (LF-TBAC) method)	{
	"Version":"2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"lakeformation:AddLFTagsToResource",
	"lakeformation:RemoveLFTagsFromResou
	rce",
	"lakeformation:GetResourceLFTags",
	"lakeformation:ListLFTags",
	"lakeformation:GetLFTag",
	"lakeformation:SearchTablesByLFTags"
	,
	"lakeformation:SearchDatabasesByLFTa
	gs"
	],
	"Resource": "*"
	}
	]
	}

Inline policy (passrole policy for the workflow role)	{
	"Version":"2012-10-17",
	"Statement": [
	{
	"Sid": "PassRolePermissions",
	"Effect": "Allow",
	"Action": [
	"iam:PassRole"
	],
	"Resource": [
	"arn:aws:iam:: *111122223
	333* :role/*<workflow_role>* "
	]
	}
	]
	}

Data analyst permissions

Policy Type	Policy
AWS managed policy	`AmazonAthenaFullAccess`
Inline policy (basic)	`{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["lakeformation:GetDataAccess", "glue:GetTable", "glue:GetTables", "glue:SearchTables", "glue:GetDatabase", "glue:GetDatabases", "glue:GetPartitions", "lakeformation:GetResourceLFTags", "lakeformation:ListLFTags", "lakeformation:GetLFTag", "lakeformation:SearchTablesByLFTags", "lakeformation:SearchDatabasesByLFTags"], "Resource": "*" }] }`
(Optional) Inline policy (for operations on governed tables, including operations within transactions)	`{ "Version":"2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "lakeformation:StartTransaction", "lakeformation:CommitTransaction", "lakeformation:CancelTransaction", "lakeformation:ExtendTransaction",`

Policy Type	Policy
	"lakeformation:DescribeTransaction", "lakeformation:ListTransactions", "lakeformation:GetTableObjects", "lakeformation:UpdateTableObjects", "lakeformation:DeleteObjectsOnCancel "], "Resource": "*" }] }

Workflow role permissions

This role has the permissions required to run a workflow. You specify a role with these permissions when you create a workflow.

> **Important**
>
> In the following policies, replace *<region>* with a valid AWS Region identifier (for example us-east-1), *<account-id>* with a valid AWS account number, *<workflow_role>* with the name of the workflow role, and *<your-s3-cloudtrail-bucket>* with the Amazon S3 path to your AWS CloudTrail logs.

Policy Type	Policy
AWS managed policy	AWSGlueServiceRole
Inline policy (data access)	{ "Version": "2012-10-17", "Statement": [{ "Sid": "Lakeformation", "Effect": "Allow", "Action": ["lakeformation:GetDataAccess",

Policy Type	Policy

	<pre>"lakeformation:GrantPermissions"
	],
	"Resource": "*"
	}
	]
	}</pre>

Inline policy (passrole policy for the workflow role)	<pre>{
	"Version":"2012-10-17",
	"Statement": [
	{
	"Sid": "PassRolePermissions",
	"Effect": "Allow",
	"Action": [
	"iam:PassRole"
	],
	"Resource": [
	"arn:aws:iam:: *111122223333* :role/*<workflow_role>* "
	]
	}
	]
	}</pre>

Inline policy (for ingesting data outside the data lake, for example, AWS CloudTrail logs)	<pre>{
	"Version":"2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": ["s3:GetObject", "s3:ListBucket"],
	"Resource": ["arn:aws:s3::: *<your-s3-cloudtrail-bucket>* /*"]
	}
	]
	}</pre>

Changing the default settings for your data lake

To maintain backward compatibility with AWS Glue, AWS Lake Formation has the following initial security settings:

- The `Super` permission is granted to the group `IAMAllowedPrincipals` on all existing AWS Glue Data Catalog resources.
- "Use only IAM access control" settings are enabled for new Data Catalog resources.

These settings effectively cause access to Data Catalog resources and Amazon S3 locations to be controlled solely by AWS Identity and Access Management (IAM) policies. Individual Lake Formation permissions are not in effect.

The `IAMAllowedPrincipals` group includes any IAM users and roles that are allowed access to your Data Catalog resources by your IAM policies. The `Super` permission enables a principal to perform every supported Lake Formation operation on the database or table on which it is granted.

To change security settings so that access to Data Catalog resources (databases and tables) is managed by Lake Formation permissions, do the following:

1. Change the default security settings for new resources. For instructions, see Change the default permission model or use hybrid access mode.
2. Change the settings for existing Data Catalog resources. For instructions, see Upgrading AWS Glue data permissions to the AWS Lake Formation model.

Changing the default security settings using the Lake Formation `PutDataLakeSettings` API operation

You can also change default security settings by using the Lake Formation PutDataLakeSettings API operation. This action takes as arguments an optional catalog ID and a DataLakeSettings structure.

To enforce metadata and underlying data access control by Lake Formation on new databases and tables, code the `DataLakeSettings` structure as follows.

> **Note**
>
> Replace *<AccountID>* with a valid AWS account ID and *<Username>* with a valid IAM user name. You can specify more than one user as a data lake administrator.

```
{
    "DataLakeSettings": {
        "DataLakeAdmins": [
            {
                "DataLakePrincipalIdentifier":
 "arn:aws:iam::<AccountId>:user/<Username>"
            }
        ],
        "CreateDatabaseDefaultPermissions": [],
        "CreateTableDefaultPermissions": []
    }
}
```

You can also code the structure as follows. Omitting the `CreateDatabaseDefaultPermissions` or `CreateTableDefaultPermissions` parameter is equivalent to passing an empty list.

```
{
    "DataLakeSettings": {
        "DataLakeAdmins": [
            {
                "DataLakePrincipalIdentifier":
 "arn:aws:iam::<AccountId>:user/<Username>"
            }
        ]
    }
}
```

This action effectively revokes all Lake Formation permissions from the `IAMAllowedPrincipals` group on new databases and tables. When you create a database, you can override this setting.

To enforce metadata and underlying data access control only by IAM on new databases and tables, code the `DataLakeSettings` structure as follows.

```
{
```

```
    "DataLakeSettings": {
        "DataLakeAdmins": [
            {
                "DataLakePrincipalIdentifier":
 "arn:aws:iam::<AccountId>:user/<Username>"
            }
        ],
        "CreateDatabaseDefaultPermissions": [
            {
                "Principal": {
                    "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                },
                "Permissions": [
                    "ALL"
                ]
            }
        ],
        "CreateTableDefaultPermissions": [
            {
                "Principal": {
                    "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                },
                "Permissions": [
                    "ALL"
                ]
            }
        ]
    }
}
```

This grants the Super Lake Formation permission to the `IAMAllowedPrincipals` group on new databases and tables. When you create a database, you can override this setting.

> **Note**
>
> In the preceding `DataLakeSettings` structure, the only permitted value for `DataLakePrincipalIdentifier` is `IAM_ALLOWED_PRINCIPALS`, and the only permitted value for `Permissions` is `ALL`.

Implicit Lake Formation permissions

AWS Lake Formation grants the following implicit permissions to data lake administrators, database creators, and table creators.

Data lake administrators

- Have `Describe` access to all resources in the Data Catalog except for resources shared from another account directly to a different principal. This access cannot be revoked from an administrator.
- Have data location permissions everywhere in the data lake.
- Can grant or revoke access to any resources in the Data Catalog to any principal (including self). This access cannot be revoked from an administrator.
- Can create databases in the Data Catalog.
- Can grant the permission to create a database to another user.

> **Note**
>
> Data lake administrators can register Amazon S3 locations only if they have IAM permissions to do so. The suggested data lake administrator policies in this guide grant those permissions. Also, data lake administrators do not have implicit permissions to drop databases or alter/drop tables created by others. However, they can grant themselves permissions to do so.

For more information about data lake administrators, see Create a data lake administrator.

Catalog creators

- Have all catalog permissions on catalogs that they create, have permissions on databases and tables that they create in the catalog, and can grant other principals in the same AWS account permission to create databases and tables in the catalog. A catalog creator who also has the `AWSLakeFormationCrossAccountManager` AWS managed policy can grant permissions on the catalog to other AWS accounts or organizations.

  Data lake administrators can use the Lake Formation console or API to designate catalog creators.

> **Note**
>
> Catalog creators do not implicitly have permissions on databases and tables that others create in the catalog.

For more information on creating catalogs, see Bringing your data into the AWS Glue Data Catalog.

Database creators

- Have all database permissions on databases that they create, have permissions on tables that they create in the database, and can grant other principals in the same AWS account permission to create tables in the database. A database creator who also has the `AWSLakeFormationCrossAccountManager` AWS managed policy can grant permissions on the database to other AWS accounts or organizations.

  Data lake administrators can use the Lake Formation console or API to designate database creators.

  > **Note**
  >
  > Database creators do not implicitly have permissions on tables that others create in the database.

  For more information, see Creating a database.

Table creators

- Have all permissions on tables that they create.
- Can grant permissions on all tables that they create to principals in the same AWS account.
- Can grant permissions on all tables that they create to other AWS accounts or organizations if they have the `AWSLakeFormationCrossAccountManager` AWS managed policy.
- Can view the databases that contain the tables that they create.

Lake Formation permissions reference

To perform AWS Lake Formation operations, principals need both Lake Formation permissions and AWS Identity and Access Management (IAM) permissions. You typically grant IAM permissions

using *coarse-grained* access control policies, as described in the section called "Overview of Lake Formation permissions". You can grant Lake Formation permissions by using the console, the API, or the AWS Command Line Interface (AWS CLI).

To learn how to grant or revoke Lake Formation permissions, see the section called "Granting data permissions" and the section called "Granting data location permissions".

> **Note**
>
> The examples in this section show how to grant permissions to principals in the same AWS account. For examples of cross-account grants, see the section called "Cross-account data sharing".

Lake Formation permissions per resource type

Following are the valid Lake Formation permissions available for each type of resource:

Resource	Permission
`Catalog`	`ALL` (`Super`), Super user
	`ALTER`
	`CREATE_DATABASE`
	`DESCRIBE`
	`DROP`
`Database`	`ALL` (`Super`)
	`ALTER`
	`CREATE_TABLE`
	`DESCRIBE`
	`DROP`

Resource	Permission
Table	ALL (Super)
	ALTER
	DELETE
	DESCRIBE
	DROP
	INSERT
	SELECT
View	ALL (Super)
	SELECT
	DESCRIBE
	DROP
Data Catalog	CREATE_DATABASE
Amazon S3 location	DATA_LOCATION_ACCESS
LF-Tags	DROP
	ALTER
LF-Tag values	ASSOCIATE
	DESCRIBE
	GrantWithLFTagExpression
LF-Tag policy - Database	ALL (Super)
	ALTER

Resource	Permission
	CREATE_TABLE
	DESCRIBE
	DROP
LF-Tag policy - Table	ALL (Super)
	ALTER
	DESCRIBE
	DELETE
	DROP
	INSERT
	SELECT
Resource link - Database or Table	DESCRIBE
	DROP
Table with data filters	DESCRIBE
	DROP
	SELECT
Table with column filter	SELECT

Topics

- Lake Formation grant and revoke AWS CLI commands
- Lake Formation permissions

Lake Formation grant and revoke AWS CLI commands

Each permission description in this section includes examples of granting the permission using an AWS CLI command. The following are the synopses of the Lake Formation **grant-permissions** and **revoke-permissions** AWS CLI commands.

```
grant-permissions
[--catalog-id <value>]
--principal <value>
--resource <value>
--permissions <value>
[--permissions-with-grant-option <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]
```

```
revoke-permissions
[--catalog-id <value>]
--principal <value>
--resource <value>
--permissions <value>
[--permissions-with-grant-option <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]
```

For detailed descriptions of these commands, see grant-permissions and revoke-permissions in the *AWS CLI Command Reference*. This section provides additional information on the `--principal` option.

The value of the `--principal` option is one of the following:

- Amazon Resource Name (ARN) for an AWS Identity and Access Management (IAM) user or role
- ARN for a user or group that authenticates through a SAML provider, such as Microsoft Active Directory Federation Service (AD FS)
- ARN for an Amazon Quick user or group
- For cross-account permissions, an AWS account ID, an organization ID, or an organizational unit ID
- For IAM Identity Center user or group, IAM Identity Center user or group ARN.

The following are syntax and examples for all `--principal` types.

Principal is an IAM user

Syntax:

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:user/<user-name>
```

Example:

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/
datalake_user1
```

Principal is an IAM role

Syntax:

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:role/<role-name>
```

Example:

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:role/workflowrole
```

Principal is a user authenticating through a SAML provider

Syntax:

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:saml-
provider/<SAMLproviderName>:user/<user-name>
```

Examples:

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:saml-provider/
idp1:user/datalake_user1
```

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:saml-provider/
AthenaLakeFormationOkta:user/athena-user@example.com
```

Principal is a group authenticating through a SAML provider

Syntax:

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:saml-
provider/<SAMLproviderName>:group/<group-name>
```

Examples:

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:saml-provider/
idp1:group/data-scientists
```

```
--principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:saml-provider/
AthenaLakeFormationOkta:group/my-group
```

Principal is an Amazon Quick Enterprise Edition user

Syntax:

```
--principal DataLakePrincipalIdentifier=arn:aws:quicksight:<region>:<account-
id>:user/<namespace>/<user-name>
```

> **Note**
>
> For *<namespace>*, you must specify `default`.

Example:

```
--principal DataLakePrincipalIdentifier=arn:aws:quicksight:us-
east-1:111122223333:user/default/bi_user1
```

Principal is an Amazon Quick Enterprise Edition group

Syntax:

```
--principal DataLakePrincipalIdentifier=arn:aws:quicksight:<region>:<account-
id>:group/<namespace>/<group-name>
```

> **Note**
>
> For *<namespace>*, you must specify `default`.

Example:

```
--principal DataLakePrincipalIdentifier=arn:aws:quicksight:us-
east-1:111122223333:group/default/data_scientists
```

Principal is an AWS account

Syntax:

```
--principal DataLakePrincipalIdentifier=<account-id>
```

Example:

```
--principal DataLakePrincipalIdentifier=111122223333
```

Principal is an organization

Syntax:

```
--principal DataLakePrincipalIdentifier=arn:aws:organizations::<account-
id>:organization/<organization-id>
```

Example:

```
--principal
 DataLakePrincipalIdentifier=arn:aws:organizations::111122223333:organization/o-
abcdefghijkl
```

Principal is an organizational unit

Syntax:

```
--principal DataLakePrincipalIdentifier=arn:aws:organizations::<account-
id>:ou/<organization-id>/<organizational-unit-id>
```

Example:

```
--principal DataLakePrincipalIdentifier=arn:aws:organizations::111122223333:ou/o-
abcdefghijkl/ou-ab00-cdefghij
```

Principal is an IAM Identity Center identity user or group

Example:User

```
--principal DataLakePrincipalIdentifier=arn:aws:identitystore:::user/<UserID>
```

Example:Group:

```
--principal DataLakePrincipalIdentifier=arn:aws:identitystore:::group/<GroupID>
```

Principal is an IAM group - `IAMAllowedPrincipals`

Lake Formation sets `Super` permission on all databases and tables in the Data Catalog to a group called `IAMAllowedPrincipals` by default. If this group permission exists on a database or a table, all principals in your account will have access to the resource through the IAM principal policies for AWS Glue. It provides backward compatibility when you start using Lake Formation permissions to secure the Data Catalog resources that were earlier protected by IAM policies for AWS Glue.

When you use Lake Formation to manage permissions for your Data Catalog resources, you need to first revoke the `IAMAllowedPrincipals` permission on the resources, or opt in the principals and the resources to hybrid access mode for Lake Formation permissions to work.

Example:

```
--principal DataLakePrincipalIdentifier=IAM_Allowed_Principals
```

Principal is an IAM group - `ALLIAMPrincipals`

When you grant permissions to `ALLIAMPrincipals` group on a Data Catalog resource, every principal in the account gets access to the Data Catalog resource using Lake Formation permissions and IAM permissions.

Example:

```
--principal DataLakePrincipalIdentifier=123456789012:IAMPrincipals
```

Lake Formation permissions

This section contains the available Lake Formation permissions that you can grant to principals.

ALTER

Permission	Granted on this resource	Grantee also needs
ALTER	DATABASE	glue:UpdateDatabase
ALTER	TABLE	glue:UpdateTable
ALTER	LF-Tag	lakeformation:UpdateLFTag

A principal with this permission can alter metadata for a database or table in the Data Catalog. For tables, you can change the column schema and add column parameters. You cannot alter columns in the underlying data that a metadata table points to.

If the property that is being altered is a registered Amazon Simple Storage Service (Amazon S3) location, the principal must have data location permissions on the new location.

Example

The following example grants the ALTER permission to user datalake_user1 on the database retail in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "ALTER" --resource '{ "Database": {"Name":"retail"}}'
```

Example

The following example grants ALTER to user datalake_user1 on the table inventory in the database retail.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "ALTER" --resource '{ "Table": {"DatabaseName":"retail",
 "Name":"inventory"}}'
```

CREATE_DATABASE

Permission	Granted on this resource	Grantee also needs
CREATE_DATABASE	Data Catalog	glue:CreateDatabase

A principal with this permission can create a metadata database or resource link in the Data Catalog. The principal can also create tables in the database.

Example

The following example grants CREATE_DATABASE to user datalake_user1 in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "CREATE_DATABASE" --resource '{ "Catalog": {}}'
```

When a principal creates a database in the Data Catalog, no permissions to underlying data are granted. The following additional metadata permissions are granted (along with the ability to grant these permissions to others):

- CREATE_TABLE in the database
- ALTER database
- DROP database

When creating a database, the principal can optionally specify an Amazon S3 location. Depending on whether the principal has data location permissions, the CREATE_DATABASE permission might not be sufficient to create databases in all cases. It is important to keep the following three cases in mind.

Create database use case	Permissions needed
The location property is unspecified.	CREATE_DATABASE is sufficient.

Create database use case	Permissions needed
The location property is specified, and the location is not managed by Lake Formation (is not registered).	CREATE_DATABASE is sufficient.
The location property is specified, and the location is managed by Lake Formation (is registered).	CREATE_DATABASE is required plus data location permissions on the specified location.

CREATE_TABLE

Permission	Granted on this resource	Grantee also needs
CREATE_TABLE	DATABASE	glue:CreateTable

A principal with this permission can create a metadata table or resource link in the Data Catalog within the specified database.

Example

The following example grants the user datalake_user1 permission to create tables in the retail database in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "CREATE_TABLE" --resource '{ "Database": {"Name":"retail"}}'
```

When a principal creates a table in the Data Catalog, all Lake Formation permissions on the table are granted to the principal, with the ability to grant these permissions to others.

Cross-account Grants

If a database owner account grants CREATE_TABLE to a recipient account, and a user in the recipient account successfully creates a table in the owner account's database, the following rules apply:

- The user and data lake administrators in the recipient account have all Lake Formation permissions on the table. They can grant permissions on the table to other principals in their account. They can't grant permissions to principals in the owner account or any other accounts.
- Data lake administrators in the owner account can grant permissions on the table to other principals in their account.

Data Location Permissions

When you attempt to create a table that points to an Amazon S3 location, depending on whether you have data location permissions, the `CREATE_TABLE` permission might not be sufficient to create a table. It's important to keep the following three cases in mind.

Create table use case	Permissions needed
The specified location is not managed by Lake Formation (is not registered).	`CREATE_TABLE` is sufficient.
The specified location is managed by Lake Formation (is registered), and the containin g database has no location property or has a location property that is not an Amazon S3 prefix of the table location.	`CREATE_TABLE` is required plus data location permissions on the specified location.
The specified location is managed by Lake Formation (is registered), and the containing database has a location property that points to a location that is registered and is an Amazon S3 prefix of the table location.	`CREATE_TABLE` is sufficient.

`DATA_LOCATION_ACCESS`

Permission	Granted on this resource	Grantee also needs
`DATA_LOCATION_ACCESS`	Amazon S3 location	(Amazon S3 permissions on the location, which must be

Permission	Granted on this resource	Grantee also needs
		specified by the role used to register the location.)

This is the only data location permission. A principal with this permission can create a metadata database or table that points to the specified Amazon S3 location. The location must be registered. A principal who has data location permissions on a location also has location permissions on child locations.

Example

The following example grants data location permissions on `s3://products/retail` to user `datalake_user1` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
 {"ResourceArn":"arn:aws:s3:::products/retail"}}'
```

`DATA_LOCATION_ACCESS` is not needed to query or update underlying data. This permission applies only to creating Data Catalog resources.

For more information about data location permissions, see Underlying data access control.

DELETE

Permission	Granted on this resource	Grantee also needs
DELETE	TABLE	(No additional IAM permissio ns are needed if the location is registered.)

A principal with this permission can insert, update, and read underlying data at the Amazon S3 location specified by the table. The principal can also view the table on the Lake Formation console and retrieve information about the table with the AWS Glue API.

Example

The following example grants the DELETE permission to the user `datalake_user1` on the table `inventory` in the database `retail` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "DELETE" --resource '{ "Table": {"DatabaseName":"retail",
 "Name":"inventory"}}'
```

This permission applies only to data in Amazon S3, and not to data in other data stores such as Amazon Relational Database Service (Amazon RDS).

DESCRIBE

Permission	Granted on this resource	Grantee also needs
DESCRIBE	Table resource link	glue:GetTable
	Database resource link	glue:GetDatabase
DESCRIBE	DATABASE	glue:GetDatabase
DESCRIBE	TABLE	glue:GetTable
DESCRIBE	LF-Tag	glue:GetTable glue:GetDatabase lakeformation:GetResourceLFTags lakeformation:ListLFTags lakeformation:GetLFTag lakeformation:SearchTablesByLFTags

Permission	Granted on this resource	Grantee also needs
		`lakeformation:SearchDatabasesByLFTags`

A principal with this permission can view the specified database, table, or resource link. No other Data Catalog permissions are implicitly granted, and no data access permissions are implicitly granted. Databases and tables appear in the query editors of integrated services, but no queries can be made against them unless other Lake Formation permissions (for example, `SELECT`) are granted.

For example, a user who has `DESCRIBE` on a database can see the database and all database metadata (description, location, and so on). However, the user can't find out which tables the database contains, and can't drop, alter, or create tables in the database. Similarly, a user who has `DESCRIBE` on a table can see the table and table metadata (description, schema, location, and so on), but can't drop, alter, or run queries against the table.

The following are some additional rules for `DESCRIBE`:

- If a user has other Lake Formation permissions on a database, table, or resource link, `DESCRIBE` is implicitly granted.
- If a user has `SELECT` on only a subset of columns for a table (partial `SELECT`), the user is restricted to seeing just those columns.
- You can't grant `DESCRIBE` to a user who has partial select on a table. Conversely, you can't specify column inclusion or exclusion lists for tables that `DESCRIBE` is granted on.

Example

The following example grants the `DESCRIBE` permission to the user `datalake_user1` on the table resource link `inventory-link` in the database `retail` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "DESCRIBE" --resource '{ "Table": {"DatabaseName":"retail",
 "Name":"inventory-link"}}'
```

DROP

Permission	Granted on this resource	Grantee also needs
DROP	DATABASE	glue:DeleteDatabase
DROP	TABLE	glue:DeleteTable
DROP	LF-Tag	lakeformation:DeleteLFTag
DROP	Database resource link Table resource link	glue:DeleteDatabase glue:DeleteTable

A principal with this permission can drop a database, table, or resource link in the Data Catalog. You can't grant DROP on a database to an external account or organization.

> **Warning**
>
> Dropping a database drops all tables in the database.

Example

The following example grants the DROP permission to the user `datalake_user1` on the database `retail` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "DROP" --resource '{ "Database": {"Name":"retail"}}'
```

Example

The following example grants DROP to the user `datalake_user1` on the table `inventory` in the database `retail`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
```

```
 --permissions "DROP" --resource '{ "Table": {"DatabaseName":"retail",
 "Name":"inventory"}}'
```

Example

The following example grants DROP to the user `datalake_user1` on the table resource link `inventory-link` in the database `retail`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "DROP" --resource '{ "Table": {"DatabaseName":"retail", "Name":"inventory-
link"}}'
```

INSERT

Permission	Granted on this resource	Grantee also needs
INSERT	TABLE	(No additional IAM permissions are needed if the location is registered.)

A principal with this permission can insert, update, and read underlying data at the Amazon S3 location specified by the table. The principal can also view the table in the Lake Formation console and retrieve information about the table with the AWS Glue API.

Example

The following example grants the `INSERT` permission to the user `datalake_user1` on the table `inventory` in the database `retail` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "INSERT" --resource '{ "Table": {"DatabaseName":"retail",
 "Name":"inventory"}}'
```

This permission applies only to data in Amazon S3, and not to data in other data stores such as Amazon RDS.

SELECT

Permission	Granted on this resource	Grantee also needs
SELECT	• TABLE	(No additional IAM permissions are needed if the location is registered.)

A principal with this permission can view a table in the Data Catalog, and can query the underlying data in Amazon S3 at the location specified by the table. The principal can view the table in the Lake Formation console and retrieve information about the table with the AWS Glue API. If column filtering was applied when this permission was granted, the principal can view the metadata only for the included columns and can query data only from the included columns.

> **Note**
>
> It is the responsibility of the integrated analytics service to apply the column filtering when processing a query.

Example

The following example grants the SELECT permission to the user `datalake_user1` on the table `inventory` in the database `retail` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "SELECT" --resource '{ "Table": {"DatabaseName":"retail",
 "Name":"inventory"}}'
```

This permission applies only to data in Amazon S3, and not to data in other data stores such as Amazon RDS.

You can filter (restrict the access to) specific columns with an optional inclusion list or an exclusion list. An inclusion list specifies the columns that can be accessed. An exclusion list specifies the columns that can't be accessed. In the absence of an inclusion or exclusion list, all table columns are accessible.

The results of `glue:GetTable` return only the columns that the caller has permission to view. Integrated services such as Amazon Athena and Amazon Redshift honor column inclusion and exclusion lists.

Example

The following example grants `SELECT` to the user `datalake_user1` on the table `inventory` using an inclusion list.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "SELECT"  --resource '{ "TableWithColumns": {"DatabaseName":"retail",
 "Name":"inventory", "ColumnNames": ["prodcode","location","period","withdrawals"]}}'
```

Example

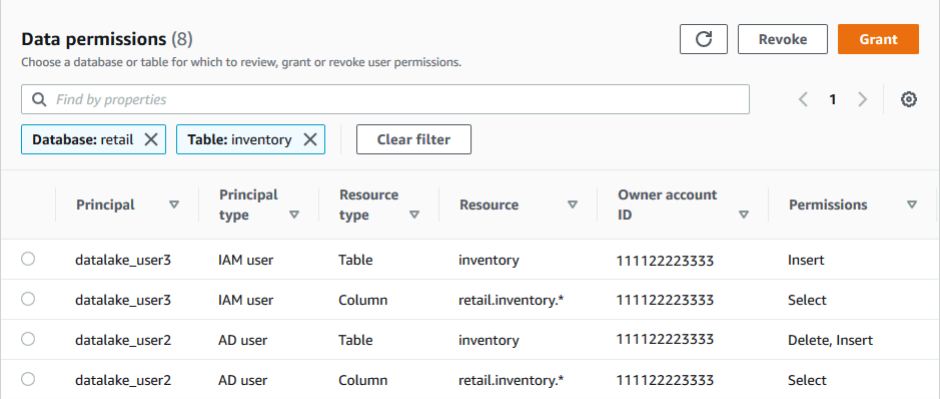
This next example grants `SELECT` on the `inventory` table using an exclusion list.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "SELECT"  --resource '{ "TableWithColumns": {"DatabaseName":"retail",
 "Name":"inventory", "ColumnWildcard": {"ExcludedColumnNames": ["intkey",
 "prodcode"]}}}'
```

The following restrictions apply to the `SELECT` permission:

- When granting `SELECT`, you can't include the grant option if column filtering is applied.
- You cannot restrict access control on columns that are partition keys.
- A principal with the `SELECT` permission on a subset of columns in a table cannot be granted the `ALTER`, `DROP`, `DELETE`, or `INSERT` permission on that table. Similarly, a principal with the `ALTER`, `DROP`, `DELETE`, or `INSERT` permission on a table cannot be granted the `SELECT` permission with column filtering.

The `SELECT` permission always appears on the **Data permissions** page of the Lake Formation console as a separate row. This following image shows that `SELECT` is granted to the users `datalake_user2` and `datalake_user3` on all columns in the `inventory` table.



Super

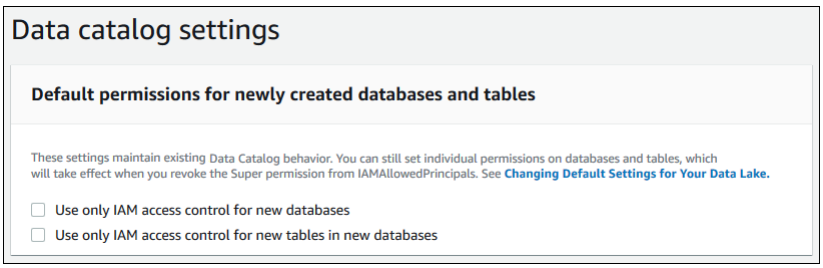
Permission	Granted on this resource	Grantee also needs
`Super`	`DATABASE`	`glue:*Database*`
`Super`	`TABLE`	`glue:*Table*`, `glue:*Partition*`

This permission allows a principal to perform every supported Lake Formation operation on the database or table. You can't grant `Super` on a database to an external account.

This permission can coexist with the other Lake Formation permissions. For example, you can grant the `Super`, `SELECT`, and `INSERT` permissions on a metadata table. The principal can then perform all supported operations on the table. When you revoke `Super`, the `SELECT` and `INSERT` permissions remain, and the principal can perform only select and insert operations.

Instead of granting `Super` to an individual principal, you can grant it to the group `IAMAllowedPrincipals`. The `IAMAllowedPrincipals` group is automatically created and includes all IAM users and roles that are permitted access to your Data Catalog resources by your IAM policies. When `Super` is granted to `IAMAllowedPrincipals` for a Data Catalog resource, access to the resource is effectively controlled solely by IAM policies.

You can have the `Super` permission to be automatically granted to `IAMAllowedPrincipals` for new catalog resources by taking advantage of options on the **Settings** page of the Lake Formation console.



- To grant `Super` to `IAMAllowedPrincipals` for all new databases, select **Use only IAM access control for new databases**.
- To grant `Super` to `IAMAllowedPrincipals` for all new tables in new databases, select **Use only IAM access control for new tables in new databases**.

> **Note**
>
> This option causes the check box **Use only IAM access control for new tables in this database** in the **Create database** dialog box to be selected by default. It does nothing more than that. It is the check box in the **Create database** dialog box that enables the grant of `Super` to `IAMAllowedPrincipals`.

These **Settings** page options are enabled by default. For more information, see the following:

- the section called “Changing the default settings for your data lake”
- the section called “Upgrading AWS Glue data permissions to the Lake Formation model”

SUPER_USER

Permission	Granted on this resource	Grantee also needs
`Super user`	`Catalog`	`glue:GetCatalog`

You can grant the `Super user` permission only to specific principals on catalogs within the default Data Catalog. You can't grant `Super user` permission on the default catalog or on other resource types such as databases and tables or to principals in external accounts. The `Super user` permission permission allows a principal to perform every supported Lake Formation operation on the databases and tables within the granted catalog.

With the `Super user` permission, the principal (grantee) is able to perform the following actions on the resources (catalogs, databases, and tables) within the catalog:

- `CREATE_DATABASE`, `DESCRIBE` permissions on the catalog.
- `DROP`, `ALTER`, `CREATE_TABLE`, `DESCRIBE` (effectively SUPER) permissions on all databases within the catalog.
- `DROP`, `ALTER`, `DESCRIBE`, `SELECT`, `INSERT`, `DELETE` (effectively SUPER) permissions on all tables within all databases within the catalog.
- `All` (effectively SUPER) permissions on catalogs within the catalog.
- Grantable (the ability to grant these permissions to other principals) permissions on all catalogs, databases, and tables within the catalog.

With the `Super user` permission on a catalog resource, the grantee is not allowed to perform or delegate `ALTER` and `DROP` actions on the catalog.

ASSOCIATE

Permission	Granted on this resource	Grantee also needs

`ASSOCIATE`	`LF-Tag`	`glue:GetDatabase`
		`glue:GetTable`
		`lakeformation:AddLFTagsToResource"`
		`lakeformation:RemoveLFTagsFromResource"`
		`lakeformation:GetResourceLFTags`
		`lakeformation:ListLFTags`
		`lakeformation:GetLFTag`

Permission	Granted on this resource	Grantee also needs

		`lakeformation:SearchTablesByLFTags`
		
		`lakeformation:SearchDatabasesByLFTags`

A principal with this permission on a LF-Tag can assign the LF-Tag to a Data Catalog resource. Granting `ASSOCIATE` implicitly grants `DESCRIBE`.

Example

This example grants to user `datalake_user1` the `ASSOCIATE` permission on the LF-Tag with the key `module`. It grants permissions to view and assign all values for that key, as indicated by the asterisk (*)..

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/
datalake_user1 --permissions "ASSOCIATE" --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

Integrating IAM Identity Center

With AWS IAM Identity Center, you can connect to identity providers (IdPs) and centrally manage access for users and groups across AWS analytics services. You can integrate identity providers such as Okta, Ping, and Microsoft Entra ID (formerly Azure Active Directory) with IAM Identity Center for users in your organization to access data using a single-sign on experience. IAM Identity Center also supports connecting additional third-party identity providers.

For more information see, Supported identity providers in the AWS IAM Identity Center User Guide.

You can configure AWS Lake Formation as an enabled application in IAM Identity Center, and data lake administrators can grant fine-grained permissions to authorized users and groups on AWS Glue Data Catalog resources.

Users from your organization can sign in to any Identity Center enabled application using your organization's identity provider, and query datasets applying Lake Formation permissions. With this integration, you can manage access to AWS services, without creating multiple IAM roles.

Trusted identity propagation is an AWS IAM Identity Center feature that administrators of connected AWS services can use to grant and audit access to service data. Access to this data is based on user attributes such as group associations. Setting up trusted identity propagation requires collaboration between the administrators of connected AWS services and the IAM Identity Center administrators. For more information, see Prerequisites and considerations.

For limitations, see IAM Identity Center integration limitations.

Topics

- Prerequisites for IAM Identity Center integration with Lake Formation
- Connecting Lake Formation with IAM Identity Center
- Updating IAM Identity Center integration
- Deleting a Lake Formation connection with IAM Identity Center
- Granting permissions to users and groups
- Including IAM Identity Center user context in CloudTrail logs

Prerequisites for IAM Identity Center integration with Lake Formation

The following are the prerequisites for integrating IAM Identity Center with Lake Formation.

1. Enable IAM Identity Center – Enabling IAM Identity Center is a prerequisite to support authentication and identity propagation.
2. Choose your identity source – After you enable IAM Identity Center, you must have an identify provider to manage users and groups. You can either use the built-in Identity Center directory as an identity source or use external IdP, such as Microsoft Entra ID or Okta.

   For more information, see Manage your identity source and Connect to an external identity provider in the AWS IAM Identity Center User Guide.
3. Create an IAM role – The role that creates IAM Identity Center connection requires permissions to create and modify application configuration in Lake Formation and IAM Identity Center as in the following inline policy.

You need to add permissions per IAM best practices. Specific permissions are detailed in the procedures that follow. For more information, see Getting started with IAM Identity Center.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:CreateLakeFormationIdentityCenterConfiguration",
                "sso:CreateApplication",
                "sso:PutApplicationAssignmentConfiguration",
                "sso:PutApplicationAuthenticationMethod",
                "sso:PutApplicationGrant",
                "sso:PutApplicationAccessScope"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

If you're sharing Data Catalog resources with external AWS accounts or organizations, you must have the AWS Resource Access Manager (AWS RAM) permissions for creating resource shares. For more information about the permissions required to share resources, see Cross-account data sharing prerequisites.

The following inline policies contain specific permissions required to view, update, and delete properties of Lake Formation integration with IAM Identity Center.

- Use the following inline policy to allow an IAM role to view a Lake Formation integration with IAM Identity Center.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [

 "lakeformation:DescribeLakeFormationIdentityCenterConfiguration",
                "sso:DescribeApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- Use the following inline policy to allow an IAM role to update a Lake Formation integration with IAM Identity Center. The policy also includes optional permissions required to share resources with external accounts.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:UpdateLakeFormationIdentityCenterConfiguration",

 "lakeformation:DescribeLakeFormationIdentityCenterConfiguration",
                "sso:DescribeApplication",
                "sso:UpdateApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- Use the following inline policy to allow an IAM role to delete a Lake Formation integration with IAM Identity Center.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:DeleteLakeFormationIdentityCenterConfiguration",
                "sso:DeleteApplication"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- For IAM permissions required to grant or revoke data lake permissions for IAM Identity Center users and groups, see IAM permissions required to grant or revoke Lake Formation permissions.

Permissions description

- `lakeformation:CreateLakeFormationIdentityCenterConfiguration` – Creates the Lake Formation IdC configuration.
- `lakeformation:DescribeLakeFormationIdentityCenterConfiguration` – Describes an existing IdC configuration.
- `lakeformation:DeleteLakeFormationIdentityCenterConfiguration` – Gives the ability to delete an existing Lake Formation IdC configuration.
- `lakeformation:UpdateLakeFormationIdentityCenterConfiguration` – Used to change an existing Lake Formation configuration.
- `sso:CreateApplication` – Used to create an IAM Identity Center application.
- `sso:DeleteApplication` – Used to delete an IAM Identity Center application.
- `sso:UpdateApplication` – Used to update an IAM Identity Center application.
- `sso:PutApplicationGrant` – Used to change the trusted token issuer information.
- `sso:PutApplicationAuthenticationMethod` – Grants Lake Formation authentication access.

- `sso:GetApplicationGrant` – Used to list trusted token issuer information.
- `sso:DeleteApplicationGrant` – Deletes the trust token issuer information.
- `sso:PutApplicationAccessScope` – Adds or updates the list of authorized targets for an IAM Identity Center access scope for an application.
- `sso:PutApplicationAssignmentConfiguration` – Used to configure how users gain access to an application.

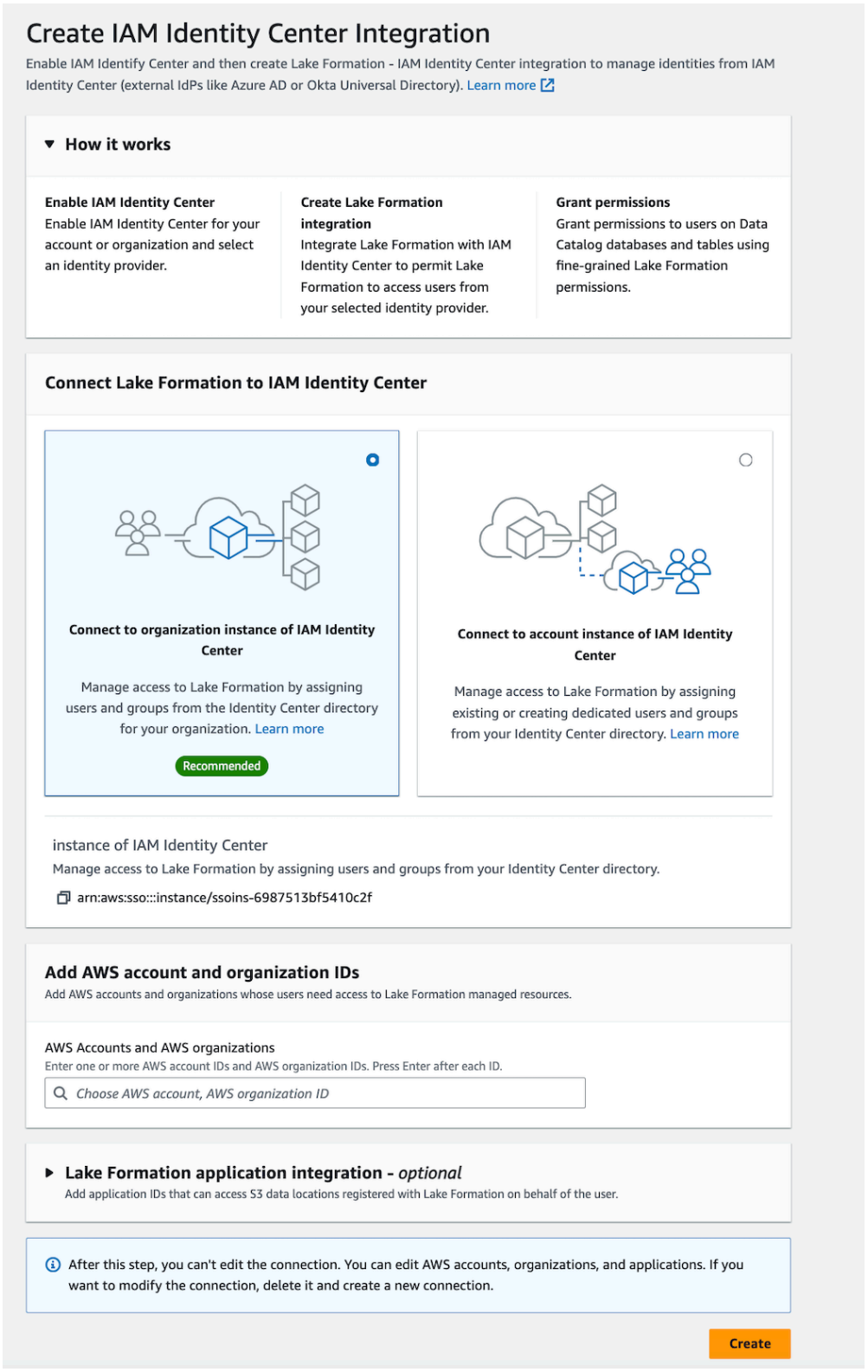
Connecting Lake Formation with IAM Identity Center

Before you can use IAM Identity Center to manage identities to grant access to Data Catalog resources using Lake Formation, you must complete the following steps. You can create the IAM Identity Center integration using the Lake Formation console or AWS CLI.

AWS Management Console

To connect Lake Formation with IAM Identity Center

1. Sign in to the AWS Management Console, and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the left navigation pane, select **IAM Identity Center integration**.



3. (Optional) Enter one or more valid AWS account IDs, organization IDs, and/or organizational unit IDs to allow external accounts to access the Data Catalog resources. When IAM Identity Center users or groups try to access Lake Formation managed Data

Catalog resources, Lake Formation assumes an IAM role to authorize metadata access. If the IAM role belongs to an external account that does not have an AWS Glue resource policy and an AWS RAM resource share, the IAM Identity Center users and groups won't be able to access the resource even if they've Lake Formation permissions.

Lake Formation uses the AWS Resource Access Manager (AWS RAM) service to share the resource with external accounts and organizations. AWS RAM sends an invitation to the grantee account to accept or reject the resource share.

For more information, see Accepting a resource share invitation from AWS RAM.

> **Note**
>
> Lake Formation permits IAM roles from external accounts to act as carrier roles on behalf of IAM Identity Center users and groups for accessing Data Catalog resources, but permissions can only be granted on Data Catalog resources within the owning account. If you try to grant permissions to IAM Identity Center users and groups on Data Catalog resources in an external account, Lake Formation throws the following error - "Cross-account grants are not supported for the principal."

4. (Optional) On the **Create Lake Formation integration** screen, specify the ARNs of third-party applications that can access data in Amazon S3 locations registered with Lake Formation. Lake Formation vends scoped-down temporary credentials in the form of AWS STS tokens to registered Amazon S3 locations based on the effective permissions, so that authorized applications can access data on behalf of users.
5. (Optional) On the **Create Lake Formation integration** screen, check mark the Amazon Redshift Connect checkbox in Trusted Identity Propagation to enable Amazon Redshift Federated Permissions discovery via IDC. Lake Formation propagates identity to downstream based on the effective permissions, so that authorized applications can access data on behalf of users.
6. Select **Submit**.

   After the Lake Formation administrator finishes the steps and creates the integration, the IAM Identity Center properties appear in the Lake Formation console. Completing these tasks makes Lake Formation an IAM Identity Center enabled application. The properties in the console include the integration status. The integration status says `Success` when it's completed. This status indicates if IAM Identity Center configuration is completed.

AWS CLI

- The following example shows how to create Lake Formation integration with IAM Identity Center. You can also specify the `Status` (`ENABLED`, `DISABLED`) of the applications.

```
aws lakeformation create-lake-formation-identity-center-configuration \
    --catalog-id <123456789012> \
    --instance-arn <arn:aws:sso:::instance/ssoins-112111f12ca1122p> \
    --share-recipients '[{"DataLakePrincipalIdentifier": "<123456789012>"},
                         {"DataLakePrincipalIdentifier": "<555555555555>"}]' \
    --external-filtering '{"AuthorizedTargets": ["<app arn1>", "<app arn2>"],
 "Status": "ENABLED"}'
```

- The following example shows how to view a Lake Formation integration with IAM Identity Center.

```
aws lakeformation describe-lake-formation-identity-center-configuration
 --catalog-id <123456789012>
```

- The following example shows how to enable `Redshift:Connect` Authorization. Authorization can be ENABLED or DISABLED.

```
aws lakeformation  create-lake-formation-identity-center-configuration \
--instance-arn <arn:aws:sso:::instance/ssoins-112111f12ca1122p> \
--service-integrations '[{
  "Redshift": [{
    "RedshiftConnect": {
      "Authorization": "ENABLED"
    }
  }]
}]'
```

- Use the `describe-lake-formation-identity-center-configuration` command to describe the lake formation identity center application. `Redshift:Connect` service integration is essential for cross-service and cross-cluster IdC identity propagation:

```
aws lakeformation describe-lake-formation-identity-center-configuration --catalog-
id <123456789012>
```

Response:

```
{
    "CatalogId": "CATALOG ID",
    "InstanceArn": "INSTANCE ARN",
    "ApplicationArn": "APPLICATION ARN",
    "ShareRecipients": [],
    "ServiceIntegrations": [
        {
            "Redshift": [
                {
                    "RedshiftConnect": {
                        "Authorization": "ENABLED"
                    }
                }
            ]
        }
    ]
}
```

Using IAM Identity Center across multiple AWS Regions

Lake Formation supports IAM Identity Center in multiple AWS Regions. You can extend IAM Identity Center from your primary AWS Region to additional Regions for improved performance through proximity to users and reliability. When a new Region is added in IAM Identity Center, you can create Lake Formation Identity Center applications in the new Region without replicating identities from the primary Region. For more details to get started with IAM Identity Center in multiple Regions, see Multi-Region IAM Identity Center in the *IAM Identity Center User Guide*.

Updating IAM Identity Center integration

After creating the connection, you can add third-party applications for the IAM Identity Center integration to integrate with Lake Formation, and get access to Amazon S3 data on behalf of the users. You can also remove existing applications from the IAM Identity Center integration. You can add or remove applications using Lake Formation console, AWS CLI, and using UpdateLakeFormationIdentityCenterConfiguration operation.

> **Note**
>
> After creating IAM Identity Center integration, you can't update the instance ARN.

AWS Management Console

To update an existing IAM Identity Center connection with Lake Formation

1. Sign in to the AWS Management Console, and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the left navigation pane, select **IAM Identity Center integration**.
3. Select **Add** on the **IAM Identity Center integration** page.
4. Enter one or more valid AWS account IDs, organization IDs, and/or organizational unit IDs to allow external accounts to access the Data Catalog resources.
5. On the **Add applications** screen, enter the application IDs of the third-party applications that you want to integrate with Lake Formation.
6. Select **Add**.
7. (Optioanlly) On the **IAM Identity Center integration** page you can either enabled trusted identity propagation for Amazon Redshift connect or disable it. Lake Formation propagates identity to downstream based on the effective permissions, so that authorized applications can access data on behalf of users.

AWS CLI

You can add or remove third-party applications for the IAM Identity Center integration by running the following AWS CLI command. When you set external filtering status to ENABLED, it enables the IAM Identity Center to provide identity management for third-party applications to access data managed by Lake Formation. You can also enable or disable the IAM Identity Center integration by setting the application status.

```
aws lakeformation update-lake-formation-identity-center-configuration \
 --external-filtering '{"AuthorizedTargets": ["<app arn1>", "<app arn2>"], "Status":
 "ENABLED"}'\
 --share-recipients '[{"DataLakePrincipalIdentifier": "<444455556666>"}
                     {"DataLakePrincipalIdentifier": "<777788889999>"}]' \
 --application-status ENABLED
```

If you have an existing LF IDC application, but wish to add the `Redshift:Connect` authorization, you can use the following to update your Lake Formation IDC Application. Authorization can be ENABLED or DISABLED.

```
aws lakeformation update-lake-formation-identity-center-configuration \
--service-integrations '[{

  "Redshift": [{
    "RedshiftConnect": {
      "Authorization": "ENABLED"
    }
  }]
}]'
```

Deleting a Lake Formation connection with IAM Identity Center

If you would like to delete an existing IAM Identity Center integration, you can do it using Lake Formation console, AWS CLI, or DeleteLakeFormationIdentityCenterConfiguration operation.

AWS Management Console

To delete an existing IAM Identity Center connection with Lake Formation

1. Sign in to the AWS Management Console, and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the left navigation pane, select **IAM Identity Center integration**.
3. Select **Delete** on the **IAM Identity Center integration** page.
4. On the **Confirm integration** screen, confirm the action, and select **Delete**.

AWS CLI

You can delete IAM Identity Center integration by running the following AWS CLI command.

```
aws lakeformation delete-lake-formation-identity-center-configuration \
    --catalog-id <123456789012>
```

Granting permissions to users and groups

Your data lake administrator can grant permissions to IAM Identity Center users and groups on Data Catalog resources (databases, tables, and views) to allow easy data access. To grant or revoke data lake permissions, the grantor requires permissions for the following IAM Identity Center actions.

- DescribeUser
- DescribeGroup
- DescribeInstance

You can grant permissions by using the Lake Formation console, the API, or the AWS CLI.

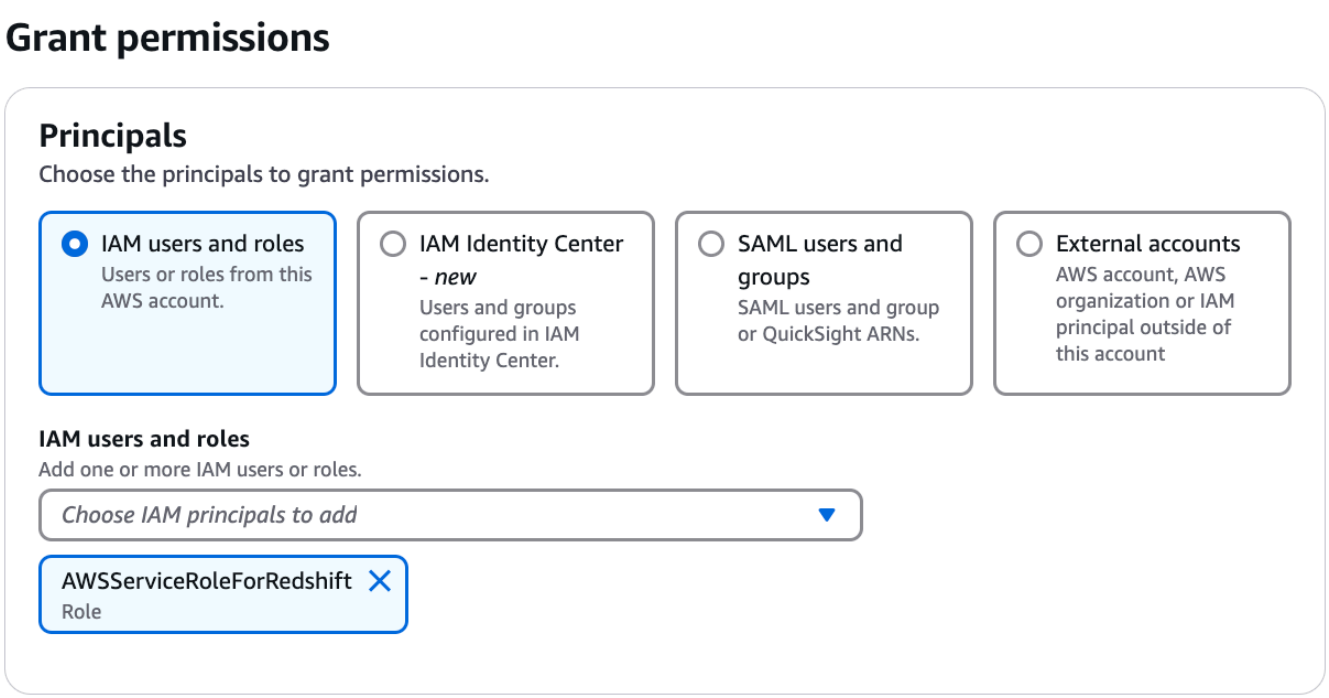
For more information on granting permissions, see the section called “Granting data permissions”.

> **Note**
>
> You can only grant permissions on resources in your account. To cascade permissions to users and groups on resources shared with you, you must use AWS RAM resources shares.

AWS Management Console

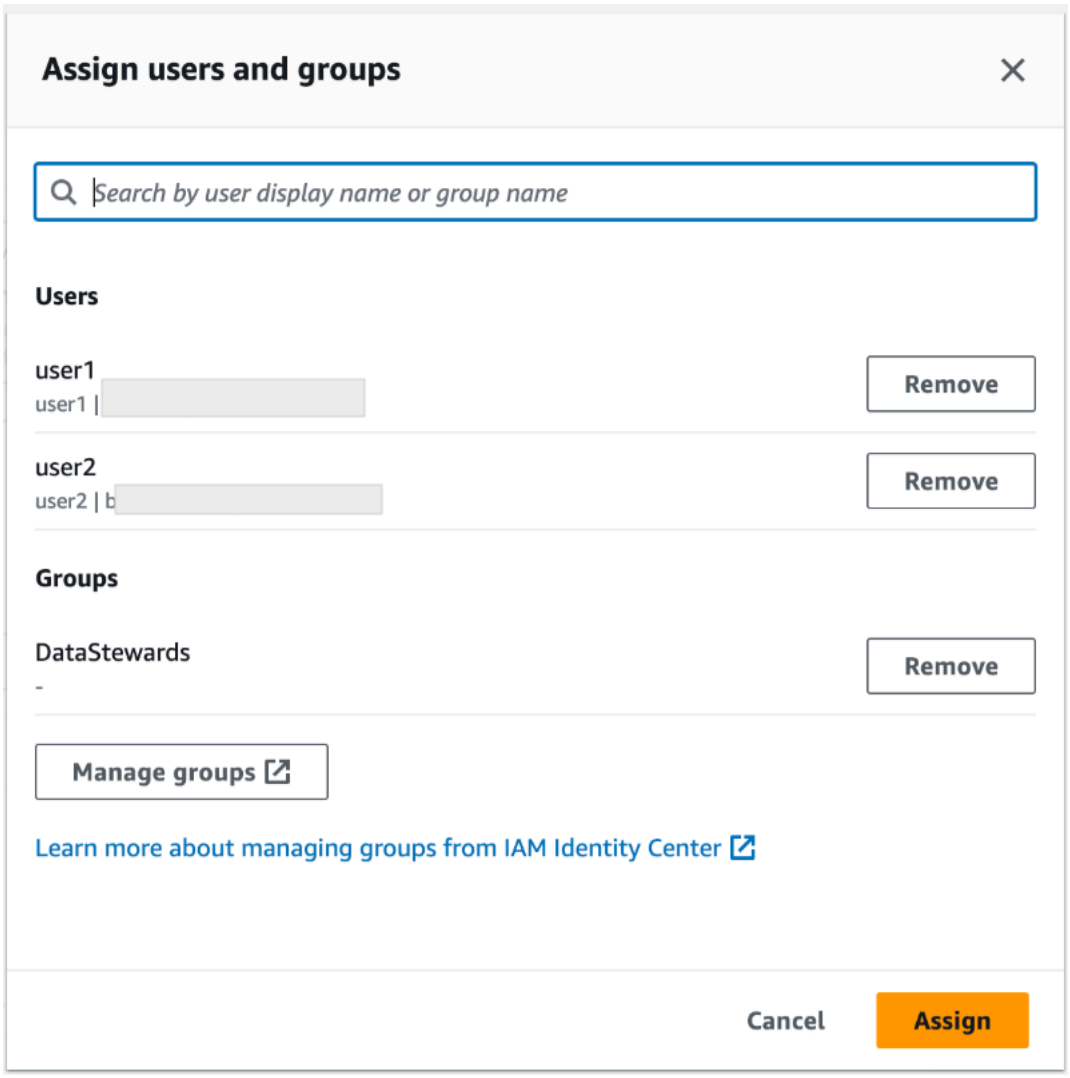
To grant permissions to users and groups

1. Sign in to the AWS Management Console, and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. Select **Data lake permissions** under **Permissions** in the Lake Formation console.
3. Select **Grant**.
4. On the **Grant data lake permissions** page, choose, **IAM Identity Center** users and groups.
5. Select **Add** to choose the users and groups to grant permissions.



6. On the **Assign users and groups** screen, choose the users and/or groups to grant permissions.

   Select **Assign**.



7. Next, choose the method to grant permissions.

   For instructions on granting permissions using named resources method, see Granting data permissions using the named resource method.

   For instructions on granting permission using LF-Tags, see Granting data lake permissions using the LF-TBAC method.

8. Choose the Data Catalog resources on which you want to grant permissions.
9. Choose the Data Catalog permissions to grant.
10. Select **Grant**.

AWS CLI

The following example shows how to grant IAM Identity Center user SELECT permission on a table.

```
aws lakeformation grant-permissions \
--principal DataLakePrincipalIdentifier=arn:aws:identitystore:::user/<UserId> \
--permissions "SELECT" \
--resource '{ "Table": { "DatabaseName": "retail", "TableWildcard": {} } }'
```

To retrieve `UserId` from IAM Identity Center, see GetUserId operation in the IAM Identity Center API Reference.

Including IAM Identity Center user context in CloudTrail logs

Lake Formation uses credential vending functionality to provide temporary access to Amazon S3 data. By default, when an IAM Identity Center user submits a query to an integrated analytics service, the CloudTrail logs only include the IAM role assumed by the service to provide short term access. If you use a user-defined role to register the Amazon S3 data location with Lake Formation, you can opt in to include the IAM Identity Center user's context in the CloudTrail events, and then track the users that access your resources.

> **Important**
>
> To include object-level Amazon S3 API requests in the CloudTrail, you need to enable CloudTrail event logging for Amazon S3 bucket and objects. For more inormation, see Enabling CloudTrail event logging for Amazon S3 buckets and objects in the Amazon S3 User Guide.

To enable credential vending auditing on data lake locations registered with user-defined roles

1. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the left-side navigation, expand **Administration**, and choose **Data Catalog settings**.
3. Under **Enhanced auditing**, choose **Propagate provided context.**
4. Choose **Save**.

You can also enable the enhanced auditing option by setting the `Parameters` attribute in the PutDataLakeSettings operation. By default, the `SET_CONTEXT"` parameter value is set to "true".

```
{
    "DataLakeSettings": {
        "Parameters": {"SET_CONTEXT": "true"},
    }
}
```

The following is an excerpt from a CloudTrail event with the enhanced auditing option. This log includes both the IAM Identity Center user's session context and the user-defined IAM role assumed by Lake Formation to access the Amazon S3 data location. See the `onBehalfOf` parameter in the following excerpt.

```
{
         "eventVersion":"1.09",
         "userIdentity":{
            "type":"AssumedRole",
            "principalId":"AROAW7F7MOX4OYE6FLIFN:access-grants-
e653760c-4e8b-44fd-94d9-309e035b75ab",
            "arn":"arn:aws:sts::123456789012:assumed-role/accessGrantsTestRole/access-
grants-e653760c-4e8b-44fd-94d9-309e035b75ab",
            "accountId":"123456789012",
            "accessKeyId":"ASIAW7F7MOX4CQLD4JIZN",
            "sessionContext":{
               "sessionIssuer":{
                  "type":"Role",
                  "principalId":"AROAW7F7MOX4OYE6FLIFN",
                  "arn":"arn:aws:iam::123456789012:role/accessGrantsTestRole",
                  "accountId":"123456789012",
                  "userName":"accessGrantsTestRole"
               },
               "attributes":{
                  "creationDate":"2023-08-09T17:24:02Z",
                  "mfaAuthenticated":"false"
               }
            },
            "onBehalfOf":{
                "userId": "<identityStoreUserId>",
                "identityStoreArn": "arn:aws:identitystore::<restOfIdentityStoreArn>"
            }
         },
```

```
        "eventTime":"2023-08-09T17:25:43Z",
        "eventSource":"s3.amazonaws.com",
        "eventName":"GetObject",
    ....
```

Adding an Amazon S3 location to your data lake

To add a data location as storage in your data lake, you *register* the location (**Data lake location**) with AWS Lake Formation. You can then use Lake Formation permissions for fine-grained access control to AWS Glue Data Catalog objects that point to this location and to the underlying data in the location.

Lake Formation also allows to register a data location in hybrid access mode and provide you the flexibility to selectively enable Lake Formation permissions for databases and tables in your Data Catalog. With the Hybrid access mode, you have an incremental path that allows you to set Lake Formation permissions for a specific set of users without interrupting the permission policies of other existing users or workloads.

For more information on setting up hybrid access mode, see Hybrid access mode

When you register a location, that Amazon S3 path and all folders under that path are registered.

For example, suppose that you have an Amazon S3 path organization like the following:

`/mybucket/accounting/sales/`

If you register `S3://mybucket/accounting`, the `sales` folder is also registered and under Lake Formation management.

For more information about registering locations, see Underlying data access control.

> **Note**
>
> Lake Formation permissions are recommended for structured data (arranged in tables with rows and columns). If your data contains object-based unstructured data, consider using Amazon S3 access grants to manage data access.

Topics

- Requirements for roles used to register locations

- Registering an Amazon S3 location
- Registering an encrypted Amazon S3 location
- Registering an Amazon S3 location in another AWS account
- Registering an encrypted Amazon S3 location across AWS accounts
- Deregistering an Amazon S3 location
- Accessing Amazon S3 locations

Requirements for roles used to register locations

You must specify an AWS Identity and Access Management (IAM) role when you register an Amazon Simple Storage Service (Amazon S3) location. AWS Lake Formation assumes that role when accessing the data in that location.

You can use one of the following role types to register a location:

- The Lake Formation service-linked role. This role grants the required permissions on the location. Using this role is the simplest way to register the location. For more information, see Using service-linked roles for Lake Formation and Service-linked role limitations.
- A user-defined role. Use a user-defined role when you need to grant more permissions than the service-linked role provides.

  You must use a user-defined role in the following circumstances:

  - When registering a location in another account.

    For more information, see the section called “Registering an Amazon S3 location in another AWS account” and the section called “Registering an encrypted Amazon S3 location across AWS accounts”.
  - If you used an AWS managed CMK (`aws/s3`) to encrypt the Amazon S3 location.

    For more information, see Registering an encrypted Amazon S3 location.
  - If you plan to access the location using Amazon EMR.

    If you already registered a location with the service-linked role and want to begin accessing the location with Amazon EMR, you must deregister the location and reregister it with a user-defined role. For more information, see the section called “Deregistering an Amazon S3 location”.

Using service-linked roles for Lake Formation

AWS Lake Formation uses an AWS Identity and Access Management (IAM) *service-linked role*. A service-linked role is a unique type of IAM role that is linked directly to Lake Formation. The service-linked role is predefined by Lake Formation and includes all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Lake Formation easier because you don't have to create a role and manually add the necessary permissions. Lake Formation defines the permissions of its service-linked role, and unless defined otherwise, only Lake Formation can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy can't be attached to any other IAM entity.

This service-linked role trusts the following services to assume the role:

- `lakeformation.amazonaws.com`

When you use a service-linked role in account A to register an Amazon S3 location that is owned by account B, the Amazon S3 bucket policy (a resource-based policy) in account B must grant access permissions to the service-linked role in account A.

For information about using service-linked role to register a data location, see Service-linked role limitations.

> **Note**
>
> Service control policies (SCPs) don't affect service-linked roles.
> For more information, see Service control policies (SCPs) in the *AWS Organizations user guide*.

Service-linked role permissions for Lake Formation

Lake Formation uses the service-linked role named `AWSServiceRoleForLakeFormationDataAccess`. This role provides a set of Amazon Simple Storage Service (Amazon S3) permissions that enable the Lake Formation integrated service (such as Amazon Athena) to access registered locations. When you register a data lake location, you must provide a role that has the required Amazon S3 read/write permissions on that location. Instead of creating a role with the required Amazon S3 permissions, you can use this service-linked role.

The first time that you name the service-linked role as the role with which to register a path, the service-linked role and a new IAM policy are created on your behalf. Lake Formation adds the path to the inline policy and attaches it to the service-linked role. When you register subsequent paths with the service-linked role, Lake Formation adds the path to the existing policy.

While signed in as a data lake administrator, register a data lake location. Then, in the IAM console, search for the role `AWSServiceRoleForLakeFormationDataAccess` and view its attached policies.

For example, after you register the location `s3://my-kinesis-test/logs`, Lake Formation creates the following inline policy and attaches it to `AWSServiceRoleForLakeFormationDataAccess`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "LakeFormationDataAccessPermissionsForS3",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::my-kinesis-test/logs/*"
            ]
        },
        {
            "Sid": "LakeFormationDataAccessPermissionsForS3ListBucket",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::my-kinesis-test"
            ]
```

```
        }
    ]
}
```

Creating a service-linked role for Lake Formation

You don't need to manually create a service-linked role. When you register an Amazon S3 location with Lake Formation in the AWS Management Console, the AWS CLI, or the AWS API, Lake Formation creates the service-linked role for you.

> **Important**
>
> This service-linked role can appear in your account if you completed an action in another service that uses the features supported by this role. To learn more, see A new role appeared in my IAM account.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you register an Amazon S3 location with Lake Formation, Lake Formation creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **Lake Formation** use case. In the AWS CLI or the AWS API, create a service-linked role with the `lakeformation.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for Lake Formation

Lake Formation does not allow you to edit the `AWSServiceRoleForLakeFormationDataAccess` service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Deleting a service-linked role for Lake Formation

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don’t have an unused entity that is not actively monitored

or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the Lake Formation service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete Lake Formation resources used by the Lake Formation

- If you've used the service-linked role to register Amazon S3 locations with Lake Formation, before deleting the service-linked role, you need to deregister the location and reregister it using a custom role.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForLakeFormationDataAccess` service-linked role. For more information, see Deleting a Service-Linked Role in the *IAM User Guide*.

The following are the requirements for a user-defined role:

- When creating the new role, on the **Create role** page of the IAM console, choose **AWS service**, and then under **Choose a use case**, choose **Lake Formation**.

  If you create the role using a different path, ensure that the role has a trust relationship with `lakeformation.amazonaws.com`. For more information, see Modifying a role trust policy (Console).
- The role must have an inline policy that grants Amazon S3 read/write permissions on the location. The following is a typical policy.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket"
            ]
        }
    ]
}
```

- Add the following trust policy to the IAM role to allow the Lake Formation service to assume the role and vend temporary credentials to the integrated analytical engines.

  To include IAM Identity Center user context in the CloudTrail logs, the trust policy must have the permission for the `sts:SetContext` action.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DataCatalogViewDefinerAssumeRole1",
            "Effect": "Allow",
            "Principal": {
               "Service": [
                    "lakeformation.amazonaws.com"
                ]
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetContext"
            ]
        }
```

```
    ]
}
```

- The data lake administrator who registers the location must have the `iam:PassRole` permission on the role.

  The following is an inline policy that grants this permission. Replace *<account-id>* with a valid AWS account number, and replace *<role-name>* with the name of the role.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PassRolePermissions",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/<role-name>"
            ]
        }
    ]
}
```

- To permit Lake Formation to add logs in CloudWatch Logs and publish metrics, add the following inline policy.

  > **Note**
  >
  > Writing to CloudWatch Logs incurs a charge.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "Sid1",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": [
                 "arn:aws:logs:us-east-1:111122223333:log-group:/aws-
lakeformation-acceleration/*",
                 "arn:aws:logs:us-east-1:111122223333:log-group:/aws-
lakeformation-acceleration/*:log-stream:*"
            ]
        }
    ]
}
```

Registering an Amazon S3 location

You must specify an AWS Identity and Access Management (IAM) role when you register an Amazon Simple Storage Service (Amazon S3) location. Lake Formation assumes that role when it grants temporary credentials to integrated AWS services that access the data in that location.

> **Important**
>
> Avoid registering an Amazon S3 bucket that has **Requester pays** enabled. For buckets registered with Lake Formation, the role used to register the bucket is always viewed as the requester. If the bucket is accessed by another AWS account, the bucket owner is charged for data access if the role belongs to the same account as the bucket owner.

You can use the AWS Lake Formation console, Lake Formation API, or AWS Command Line Interface (AWS CLI) to register an Amazon S3 location.

Before you begin

Review the requirements for the role used to register the location.

To register a location (console)

> **Important**
>
> The following procedures assume that the Amazon S3 location is in the same AWS account as the Data Catalog and that the data in the location is not encrypted. Other sections in this chapter cover cross-account registration and registration of encrypted locations.

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator or as a user with the `lakeformation:RegisterResource` IAM permission.
2. In the navigation pane, under **Administration**, select **Data lake locations**.
3. Choose **Register location**, and then choose **Browse** to select an Amazon Simple Storage Service (Amazon S3) path.
4. (Optional, but strongly recommended) Select **Review location permissions** to view a list of all existing resources in the selected Amazon S3 location and their permissions.

   Registering the selected location might result in your Lake Formation users gaining access to data already at that location. Viewing this list helps you ensure that existing data remains secure.
5. For **IAM role**, choose either the `AWSServiceRoleForLakeFormationDataAccess` service-linked role (the default) or a custom IAM role that meets the requirements in the section called "Requirements for roles used to register locations".

   You can update a registered location or other details only when you register it using a custom IAM role. To edit a location registered using a service-linked role, you should deregister the location and register it again.
6. Choose **Enable Data Catalog Federation** option to allow Lake Formation to assume a role and vend temporary credentials to integrated AWS services to access tables under federated databases. If a location is registered with Lake Formation, and you want to use the same location for a table under a federated database, you need to register the same location with the **Enable Data Catalog Federation** option.
7. Choose **Hybrid access mode** to not enable Lake Formation permissions by default. When you register Amazon S3 location in hybrid access mode, you can enable Lake Formation permissions by opting in principals for databases and tables under that location.

For more information on setting up hybrid access mode, see Hybrid access mode.

8. Choose **Register S3 path with Owner Account** and provide the S3 bucket owner account to allow Lake Formation to verify the registration role's access on the Amazon S3 location and enable Lake Formation to vend temporary credentials for Amazon S3 files under the registered location.
9. Select **Register location**.

To register a location (AWS CLI)

1. **Register a new location with Lake Formation**

   This example uses a service-linked role to register the location. You can use the `--role-arn` argument instead to supply your own role.

   Replace *<s3-path>* with a valid Amazon S3 path, account number with a valid AWS account, and *<s3-access-role>* with an IAM role that has permissions to register a data location.

   > **Note**
   >
   > You can't edit properties of a registered location if it is registered using a service-linked role.

   ```
   aws lakeformation register-resource \
    --resource-arn arn:aws:s3:::<s3-path> \
    --use-service-linked-role
   ```

   The following example uses a custom role to register the location.

   ```
   aws lakeformation register-resource \
    --resource-arn arn:aws:s3:::<s3-path> \
    --role-arn arn:aws:iam::<123456789012>:role/<s3-access-role>
   ```

2. **To update a location registered with Lake Formation**

   You can edit a registered location only if it is registered using a custom IAM role. For a location registered with service-linked role, you should deregister the location and register it again. For more information, see the section called “Deregistering an Amazon S3 location”.

```
aws lakeformation update-resource \
 --role-arn arn:aws:iam::<123456789012>:role/<s3-access-role>\
 --resource-arn arn:aws:s3:::<s3-path>
```

```
aws lakeformation update-resource \
 --resource-arn arn:aws:s3:::<s3-path> \
 --use-service-linked-role
```

3. **Register a data location in hybrid access mode with federation**

```
aws lakeformation register-resource \
 --resource-arn arn:aws:s3:::<s3-path> \
 --role-arn arn:aws:iam::<123456789012>:role/<s3-access-role> \
 --hybrid-access-enabled
```

```
aws lakeformation register-resource \
 --resource-arn arn:aws:s3:::<s3-path> \
 --role-arn arn:aws:iam::<123456789012>:role/<s3-access-role> \
 --with-federation
```

```
aws lakeformation update-resource \
 --resource-arn arn:aws:s3:::<s3-path> \
 --role-arn arn:aws:iam::<123456789012>:role/<s3-access-role> \
 --hybrid-access-enabled
```

4. **Register a location with expected owner account**

   Use the `--expected-resource-owner-account` flag to allow Lake Formation to verify the registration role's access on the Amazon S3 location and vend temporary credentials for Amazon S3 files under the registered location.

```
aws lakeformation register-resource \
```

```
 --resource-arn arn:aws:s3:::<s3-path> \
 --role-arn arn:aws:iam::<123456789012>:role/<s3-access-role> \
 --expected-resource-owner-account <123456789012>
```

```
aws lakeformation update-resource \
 --resource-arn arn:aws:s3:::<s3-path> \
 --role-arn arn:aws:iam::<123456789012>:role/<s3-access-role> \
 --expected-resource-owner-account <123456789012>
```

For more information, see RegisterResource API operation.

Note

Once you register an Amazon S3 location, any AWS Glue table pointing to the location (or any of its child locations) will return the value for the `IsRegisteredWithLakeFormation` parameter as `true` in the `GetTable` call. There is a known limitation that Data Catalog API operations such as `GetTables` and `SearchTables` do not update the value for the `IsRegisteredWithLakeFormation` parameter, and return the default, which is false. It is recommended to use the `GetTable` API to view the correct value for the `IsRegisteredWithLakeFormation` parameter.

Registering an encrypted Amazon S3 location

Lake Formation integrates with AWS Key Management Service (AWS KMS) to enable you to more easily set up other integrated services to encrypt and decrypt data in Amazon Simple Storage Service (Amazon S3) locations.

Both customer managed AWS KMS keys and AWS managed keys are supported. Currently, client-side encryption/decryption is supported only with Athena.

You must specify an AWS Identity and Access Management (IAM) role when you register an Amazon S3 location. For encrypted Amazon S3 locations, either the role must have permission to encrypt and decrypt data with the AWS KMS key, or the KMS key policy must grant permissions on the key to the role.

> **Important**
>
> Avoid registering an Amazon S3 bucket that has **Requester pays** enabled. For buckets registered with Lake Formation, the role used to register the bucket is always viewed as the requester. If the bucket is accessed by another AWS account, the bucket owner is charged for data access if the role belongs to the same account as the bucket owner.

Lake Formation uses a service-linked role to register your data locations. However, this role has several limitations. Due to these constraints, we recommend creating and using a custom IAM role instead for more flexibility and control. The custom role you create to register the location must meets the requirements specified in the section called "Requirements for roles used to register locations".

> **Important**
>
> If you used an AWS managed key to encrypt the Amazon S3 location, you can't use the Lake Formation service-linked role. You must use a custom role and add IAM permissions on the key to the role. Details are provided later in this section.

The following procedures explain how to register an Amazon S3 location that is encrypted with either a customer managed key or an AWS managed key.

- Registering a location encrypted with a customer managed key
- Registering a location encrypted with an AWS managed key

Before You Begin

Review the requirements for the role used to register the location.

To register an Amazon S3 location encrypted with a customer managed key

> **Note**
>
> If the KMS key or Amazon S3 location are not in the same AWS account as the Data Catalog, follow the instructions in the section called "Registering an encrypted Amazon S3 location across AWS accounts" instead.

1. Open the AWS KMS console at https://console.aws.amazon.com/kms and log in as an AWS Identity and Access Management (IAM) administrative user or as a user who can modify the key policy of the KMS key used to encrypt the location.
2. In the navigation pane, choose **Customer managed keys**, and then choose the name of the desired KMS key.
3. On the KMS key details page, choose the **Key policy** tab, and then do one of the following to add your custom role or the Lake Formation service-linked role as a KMS key user:

   - **If the default view is showing** (with **Key administrators**, **Key deletion**, **Key users**, and **Other AWS accounts** sections) – Under the **Key users** section, add your custom role or the Lake Formation service-linked role `AWSServiceRoleForLakeFormationDataAccess`.
   - **If the key policy (JSON) is showing** – Edit the policy to add your custom role or the Lake Formation service-linked role `AWSServiceRoleForLakeFormationDataAccess` to the object "Allow use of the key," as shown in the following example.

   > **Note**
   >
   > If that object is missing, add it with the permissions shown in the example. The example uses the service-linked role.

```
        ...
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:role/aws-service-role/
lakeformation.amazonaws.com/AWSServiceRoleForLakeFormationDataAccess",
                    "arn:aws:iam::111122223333:user/keyuser"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
```

```
            "Resource": "*"
        },
        ...
```

4. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator or as a user with the `lakeformation:RegisterResource` IAM permission.
5. In the navigation pane, under **Administration**, choose **Data lake locations**.
6. Choose **Register location**, and then choose **Browse** to select an Amazon Simple Storage Service (Amazon S3) path.
7. (Optional, but strongly recommended) Choose **Review location permissions** to view a list of all existing resources in the selected Amazon S3 location and their permissions.

   Registering the selected location might result in your Lake Formation users gaining access to data already at that location. Viewing this list helps you ensure that existing data remains secure.
8. For **IAM role**, choose either the `AWSServiceRoleForLakeFormationDataAccess` service-linked role (the default) or your custom role that meets the the section called "Requirements for roles used to register locations".
9. Choose **Register location**.

For more information about the service-linked role, see Service-linked role permissions for Lake Formation.

To register an Amazon S3 location encrypted with an AWS managed key

> **Important**
>
> If the Amazon S3 location is not in the same AWS account as the Data Catalog, follow the instructions in the section called "Registering an encrypted Amazon S3 location across AWS accounts" instead.

1. Create an IAM role to use to register the location. Ensure that it meets the requirements listed in the section called "Requirements for roles used to register locations".

2. Add the following inline policy to the role. It grants permissions on the key to the role. The `Resource` specification must designate the Amazon Resource Name (ARN) of the AWS managed key. You can obtain the ARN from the AWS KMS console. To get the correct ARN, ensure that you log in to the AWS KMS console with the same AWS account and Region as the AWS managed key that was used to encrypt the location.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ]
}
```

   You can use KMS key aliases instead of the key ID - `arn:aws:kms:region:account-id:key/alias/your-key-alias`

   For more information, see Aliases in AWS KMS section in the AWS Key Management Service Developer Guide.

3. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator or as a user with the `lakeformation:RegisterResource` IAM permission.
4. In the navigation pane, under **Administration**, choose **Data lake locations**.
5. Choose **Register location**, and then choose **Browse** to select an Amazon S3 path.
6. (Optional, but strongly recommended) Choose **Review location permissions** to view a list of all existing resources in the selected Amazon S3 location and their permissions.

Registering the selected location might result in your Lake Formation users gaining access to data already at that location. Viewing this list helps you ensure that existing data remains secure.

7. For **IAM role**, choose the role that you created in Step 1.
8. Choose **Register location**.

Registering an Amazon S3 location in another AWS account

AWS Lake Formation enables you to register Amazon Simple Storage Service (Amazon S3) locations across AWS accounts. For example, if the AWS Glue Data Catalog is in account A, a user in account A can register an Amazon S3 bucket in account B.

Registering an Amazon S3 bucket in AWS account B using an AWS Identity and Access Management (IAM) role in AWS account A requires the following permissions:

- The role in account A must grant permissions on the bucket in account B.
- The bucket policy in account B must grant access permissions to the role in Account A.

> **Important**
>
> Avoid registering an Amazon S3 bucket that has **Requester pays** enabled. For buckets registered with Lake Formation, the role used to register the bucket is always viewed as the requester. If the bucket is accessed by another AWS account, the bucket owner is charged for data access if the role belongs to the same account as the bucket owner.
> You can't use the Lake Formation service-linked role to register a location in another account. You must use a user-defined role instead. The role must meet the requirements in the section called “Requirements for roles used to register locations”. For more information about the service-linked role, see Service-linked role permissions for Lake Formation.

Before you begin

Review the requirements for the role used to register the location.

To register a location in another AWS account

> **Note**
>
> If the location is encrypted, follow the instructions in the section called "Registering an encrypted Amazon S3 location across AWS accounts" instead.

The following procedure assumes that a principal in account 1111-2222-3333, which contains the Data Catalog, wants to register the Amazon S3 bucket `awsexamplebucket1`, which is in account 1234-5678-9012.

1. In account 1111-2222-3333, sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Create a new role or view an existing role that meets the requirements in the section called "Requirements for roles used to register locations". Ensure that the role grants Amazon S3 permissions on `awsexamplebucket1`.
3. Open the Amazon S3 console at https://console.aws.amazon.com/s3/. Sign in with account 1234-5678-9012.
4. In the **Bucket name** list, choose the bucket name, `awsexamplebucket1`.
5. Choose **Permissions**.
6. On the **Permissions** page, choose **Bucket Policy**.
7. In the **Bucket policy editor**, paste the following policy. Replace *<role-name>* with the name of your role.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect":"Allow",
            "Principal": {
                "AWS":"arn:aws:iam::111122223333:role/<role-name>"
            },
            "Action":"s3:ListBucket",
            "Resource":"arn:aws:s3:::awsexamplebucket1"
```

```
        },
        {
            "Effect":"Allow",
            "Principal": {
                "AWS":"arn:aws:iam::111122223333:role/<role-name>"
            },
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource":"arn:aws:s3:::awsexamplebucket1/*"
        }
    ]
}
```

8. Choose **Save**.
9. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in to account 1111-2222-3333 as the data lake administrator or as a user with sufficient permissions to register locations.
10. In the navigation pane, under **Administration**, choose **Data lake locations**.
11. On **Data lake locations** page, choose **Register location**.
12. On the **Register location page**, for **Amazon S3 path**, enter the bucket name `s3://awsexamplebucket1`.

> **Note**
>
> You must type the bucket name because cross-account buckets do not appear in the list when you choose **Browse**.

13. For **IAM role**, choose your role.
14. Choose **Register location**.

Registering an encrypted Amazon S3 location across AWS accounts

AWS Lake Formation integrates with AWS Key Management Service (AWS KMS) to enable you to more easily set up other integrated services to encrypt and decrypt data in Amazon Simple Storage Service (Amazon S3) locations.

Both customer managed keys and AWS managed keys are supported. Client-side encryption/decryption is not supported.

> **Important**
>
> Avoid registering an Amazon S3 bucket that has **Requester pays** enabled. For buckets registered with Lake Formation, the role used to register the bucket is always viewed as the requester. If the bucket is accessed by another AWS account, the bucket owner is charged for data access if the role belongs to the same account as the bucket owner.

This section explains how to register an Amazon S3 location under the following circumstances:

- The data in the Amazon S3 location is encrypted with a KMS key created in AWS KMS.
- The Amazon S3 location is not in the same AWS account as the AWS Glue Data Catalog.
- The KMS key either is or is not in the same AWS account as the Data Catalog.

Registering an AWS KMS–encrypted Amazon S3 bucket in AWS account B using an AWS Identity and Access Management (IAM) role in AWS account A requires the following permissions:

- The role in account A must grant permissions on the bucket in account B.
- The bucket policy in account B must grant access permissions to the role in Account A.
- If the KMS key is in account B, the key policy must grant access to the role in account A, and the role in account A must grant permissions on the KMS key.

In the following procedure, you create a role in the AWS account that contains the Data Catalog (account A in the previous discussion). Then, you use this role to register the location. Lake Formation assumes this role when accessing underlying data in Amazon S3. The assumed role has the required permissions on the KMS key. As a result, you don't have to grant permissions on the KMS key to principals accessing underlying data with ETL jobs or with integrated services such as Amazon Athena.

> **Important**
>
> You can't use the Lake Formation service-linked role to register a location in another account. You must use a user-defined role instead. The role must meet the requirements in

the section called "Requirements for roles used to register locations". For more information about the service-linked role, see Service-linked role permissions for Lake Formation.

Before You Begin

Review the requirements for the role used to register the location.

To register an encrypted Amazon S3 location across AWS accounts

1. In the same AWS account as the Data Catalog, sign into the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Create a new role or view an existing role that meets the requirements in the section called "Requirements for roles used to register locations". Ensure that the role includes a policy that grants Amazon S3 permissions on the location.
3. If the KMS key is not in the same account as the Data Catalog, add to the role an inline policy that grants the required permissions on the KMS key. The following is an example policy. Replace Region and account ID with the region and account number of the KMS key. Replace *<key-id>* with the key ID.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
         ],
        "Resource": "arn:aws:kms:us-east-1:111122223333:key/<key-id>"
        }
    ]
}
```

4. On the Amazon S3 console, add a bucket policy granting the required Amazon S3 permissions to the role. The following is an example bucket policy. Replace the account ID with the AWS account number of the Data Catalog, *<role-name>* with the name of your role, and *<bucket-name>* with the name of the bucket.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect":"Allow",
            "Principal": {
                "AWS":"arn:aws:iam::111122223333:role/<role-name>"
            },
            "Action":"s3:ListBucket",
            "Resource":"arn:aws:s3:::<bucket-name>"
        },
        {
            "Effect":"Allow",
            "Principal": {
                "AWS":"arn:aws:iam::111122223333:role/<role-name>"
            },
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource":"arn:aws:s3:::<bucket-name>/*"
        }
    ]
}
```

5. In AWS KMS, add the role as a user of the KMS key.

   a. Open the AWS KMS console at https://console.aws.amazon.com/kms. Then, sign in as an administrator user or as a user who can modify the key policy of the KMS key used to encrypt the location.

   b. In the navigation pane, choose **Customer managed keys**, and then choose the name of the KMS key.

c. On the KMS key details page, under the **Key policy** tab, if the JSON view of the key policy is not showing, choose **Switch to policy view**.

d. In the **Key policy** section, choose **Edit**, and add the Amazon Resource Name (ARN) of the role to the `Allow use of the key` object, as shown in the following example.

> **Note**
>
> If that object is missing, add it with the permissions shown in the example.

```
...
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::<catalog-account-id>:role/<role-name>"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
...
```

For more information, see Allowing Users in Other Accounts to Use a KMS key in the *AWS Key Management Service Developer Guide*.

6. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign into the Data Catalog AWS account as the data lake administrator.
7. In the navigation pane, under **Administration**, choose **Data lake locations**.
8. Choose **Register location**.

9. On the **Register location page**, for **Amazon S3 path**, enter the location path as **`s3://<bucket>/<prefix>`**. Replace *`<bucket>`* with the name of the bucket and *`<prefix>`* with the rest of the path for the location.

   > **Note**
   >
   > You must type the path because cross-account buckets do not appear in the list when you choose **Browse**.

10. For **IAM role**, choose the role from Step 2.
11. Choose **Register location**.

Deregistering an Amazon S3 location

You can deregister an Amazon Simple Storage Service (Amazon S3) location if you no longer want it to be managed by Lake Formation. Deregistering a location does not affect Lake Formation data location permissions that are granted on that location. You can reregister a location that you deregistered, and the data location permissions remain in effect. You can use a different role to reregister the location.

To deregister a location (console)

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator or as a user with the `lakeformation:RegisterResource` IAM permission.
2. In the navigation pane, under **Administration**, choose **Data lake locations**.
3. Select a location, and on the **Actions** menu, choose **Remove**.
4. When prompted for confirmation, choose **Remove**.

Accessing Amazon S3 locations

AWS Lake Formation enables you to read and write the underlying data files in Amazon Simple Storage Service (Amazon S3) for tables registered in the AWS Glue Data Catalog (Data Catalog). This provides you with a single set of permissions for both SQL queries and direct file access using your existing Lake Formation table grants. Lake Formation extends its credential vending functionality to Amazon S3 locations registered in the Data Catalog. When your application

requests access to the Amazon S3 files underlying a Data Catalog table, Lake Formation evaluates the caller's existing table-level permissions and, if authorized, returns short-lived, scoped Amazon S3 credentials for that table's registered location.

Enable direct access to Amazon S3 locations

To enable this feature, you must complete the following steps.

Prerequisites

Complete the following prerequisites before enabling the feature flag:

- **Specify table location** – Specify the path to your data in Amazon S3 using the LOCATION property in the table attributes. For more information about the location format, see Table location in Amazon S3 in the *Amazon Athena User Guide*.
- **Register Amazon S3 locations** – Catalog your tables in the Data Catalog for your Amazon S3 locations and register them with Lake Formation by providing the Amazon S3 bucket owner account. When you register a location, that Amazon S3 path and all folders under that path are registered. For more information, see Registering an Amazon S3 location.
- **Grant Lake Formation permissions** – Grant Lake Formation SELECT (or SUPER) permissions on tables to your data scientists and applications using the Lake Formation console or APIs. If you already use Lake Formation for table access with services like Athena or Amazon EMR, this is already configured.
- **Enable Application Integration with Full Table Access** – Enable Application Integration with Full Table Access to allow credential vending to the registered table location. For more information, see Application integration for full table access.

Enable the feature flag

Enable the `fs.s3a.lakeformation.access.grants.enabled` flag on supported engines:

```
fs.s3a.lakeformation.access.grants.enabled = true
```

- **Amazon EMR on Amazon EC2** – EMR 7.13 or later
- **Amazon EMR on EKS** – EMR 7.13 or later
- **Amazon EMR Serverless** – EMR 7.13 or later

Once configured, you can immediately read or write data files from EMR using standard APIs.

For more information on Amazon EMR integration, see Lake Formation path-based credential vending in the *Amazon EMR Management Guide*.

How it works

Lake Formation-based Amazon S3 location access follows this flow:

1. A principal or role requests access to Amazon S3 data files through a service such as an Amazon EMR Spark job, or data processing pipeline.
2. The AWS SDK plugin integrated with EMR intercepts the Amazon S3 request and calls the Lake Formation GetTemporaryDataLocationCredentials API.
3. Lake Formation checks if the Amazon S3 path corresponds to a registered location and cataloged table, and evaluates the caller's permissions on the associated AWS Glue table.
4. If the caller has SELECT or SUPER permissions on the corresponding table with full table access, Lake Formation vends temporary credentials scoped to that Amazon S3 location. SELECT grants **READ** credentials; SUPER grants **READWRITE** credentials.
5. The plugin uses these credentials to complete the Amazon S3 request, and Amazon S3 provides the data to the application.
6. All credential vending operations are logged to CloudTrail, providing an audit trail for data access.

This process is transparent to you and you will simply use existing Lake Formation permissions with standard file-based APIs.

```
# Read raw data (Lake Formation-based S3 Location access)
raw_df = spark.read.json("s3://finance-datalake/raw/transactions/dt=2024-03-21/")

# Read governed data (Lake Formation-based S3 Location access)
transactions_df = spark.read.parquet("s3://data-lake/transactions/year=2026/")

# Write processed data (Lake Formation-based S3 Location access)
processed_df.write \
    .mode("append") \
    .partitionBy("transaction_date") \
    .parquet("s3://finance-datalake/processed/transactions/")

print("ETL complete. Records written:", processed_df.count())
```

Permission requirements

The feature currently vends credentials only when the caller has **full table access**, meaning SELECT on all columns without row or column filters. If a table has row-level or column-level filters applied, callers must continue using trusted engines like Athena, Amazon EMR, AWS Glue, or Amazon Redshift that can enforce those filters. This ensures security boundaries remain consistent.

Cross-account access

Lake Formation simplified access to Amazon S3 locations works with cross-account sharing capabilities. When you share a table with another AWS account through Lake Formation, recipients can access the underlying Amazon S3 data files using their Lake Formation permissions, subject to the same full table access requirements.

The feature supports Lake Formation's resource link mechanism for cross-account access. When a consumer account has permissions on a shared table, Lake Formation vends credentials scoped to the registered account's Amazon S3 location, enabling seamless cross-account data access without requiring separate Amazon S3 bucket policies or cross-account IAM roles.

Nested Amazon S3 locations

When multiple tables point to nested locations in the same bucket, Lake Formation applies the following behavior:

- When accessing `s3://bucket`, you receive permissions corresponding to the table registered at the bucket level.
- When accessing `s3://bucket/folder1`, you receive permissions corresponding to the table registered at that specific path.
- For access to folders without a registered table (for example, `s3://bucket/folder2`), you receive permissions from the nearest parent registered location.
- If multiple tables are registered at the same location, Lake Formation returns an error due to conflicting permissions.

Auditing and compliance

All credential vending operations are logged to CloudTrail, providing an audit trail for data access. When Lake Formation vends credentials through the `GetTemporaryDataLocationCredentials` API, CloudTrail records:

- The principal (user or role)

- Timestamp
- Amazon S3 location
- Associated AWS Glue table

Subsequent Amazon S3 API calls made with those credentials are also logged to CloudTrail as Amazon S3 data events, with context linking them back to the Lake Formation permission grant. This provides auditors with complete visibility into who accessed what data, when, and through which permission path—all in a single CloudTrail log stream.

Supported AWS services

Lake Formation simplified access to Amazon S3 locations works with:

- Amazon EMR on Amazon EC2 (EMR 7.13 or later)
- Amazon EMR on EKS (EMR 7.13 or later)
- Amazon EMR Serverless (EMR 7.13 or later)

Open source plugin for third-party services

You can also integrate your Apache Spark or Trino applications using APIs or through an open source plugin provided by AWS. For more information, see aws-lakeformation-accessgrants-plugin-java-v2 on GitHub.

Considerations

Note the following considerations when using credential vending for Amazon S3 locations:

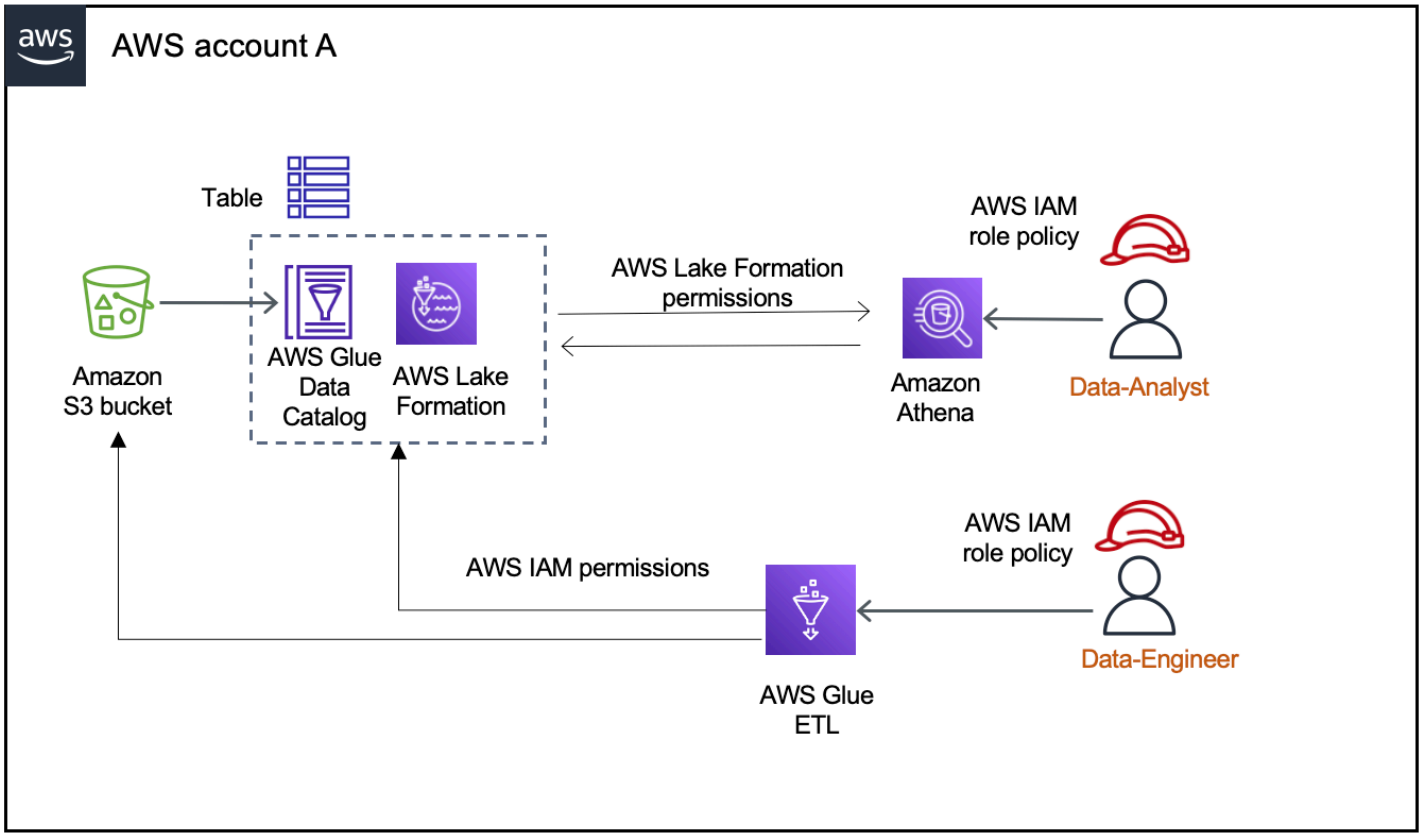
- Credential vending for Amazon S3 locations is not supported cross-region.
- Credential vending is supported for Amazon S3 locations included as the table's primary location.
- The plugin supports Apache Hive, Apache Hudi, and Delta Lake table formats. Apache Iceberg is not currently supported.
- The plugin is not currently supported with Amazon EMR Spark Fine-Grained Access Control (FGAC) mode.

Hybrid access mode

AWS Lake Formation *hybrid access mode* supports two permission pathways to the same AWS Glue Data Catalog objects.
In the first pathway, Lake Formation allows you to select specific principals, and grant them Lake Formation permissions to access catalogs, databases, tables, and views by opting in. The second pathway allows all other principals to access these resources through the default IAM principal policies for Amazon S3 and AWS Glue actions.

When registering an Amazon S3 location with Lake Formation, you have the option to either enforce Lake Formation permissions for all resources at this location or use hybrid access mode. The hybrid access mode enforces only `CREATE_TABLE`, `CREATE_PARTITION`, `UPDATE_TABLE` permissions by default. When an Amazon S3 location is in the hybrid mode, you can enable Lake Formation permissions by opting in principals for the Data Catalog objects under that location. It means both Lake Formation permissions and IAM permissions can control access to that data. This means that opted in principals will require both Lake Formation permissions and IAM permissions to access the data, while non-opted-in principals will continue to access data using only IAM permissions.

Thus, hybrid access mode provides the flexibility to selectively enable Lake Formation for catalogs, databases, and tables in your Data Catalog for a specific set of users without interrupting the access for other existing users or workloads.



For considerations and limitations, see Hybrid access mode considerations and limitations.

Terms and definitions

Here are the definitions of Data Catalog resources based on how you set up access permissions:

Lake Formation resource

A resource that is registered with Lake Formation. Users require Lake Formation permissions to access the resource.

AWS Glue resource

A resources that is not registered with Lake Formation. Users require only IAM permissions to access the resource because it has `IAMAllowedPrincipals` group permissions. Lake Formation permissions are not enforced.

For more information on `IAMAllowedPrincipals` group permissions, see Metadata permissions.

Hybrid resource

A resources that is registered in hybrid access mode. Based on the users accessing the resource, the resource dynamically switch between being a Lake Formation resource or an AWS Glue resource.

Common hybrid access mode use cases

You can use hybrid access mode to provide access in single account and cross-account data sharing scenarios:

Single account scenarios

- **Convert an AWS Glue resource to a hybrid resource** – In this scenario, you are not currently using Lake Formation but want to adopt Lake Formation permissions for Data Catalog objects. When you register the Amazon S3 location in hybrid access mode, you can grant Lake Formation permissions to users who opt in specific databases and tables pointing to that location.
- **Convert a Lake Formation resource to a hybrid resource** – Currently, you are using Lake Formation permissions to control access for a Data Catalog database but want to provide access to new principals using IAM permissions for Amazon S3 and AWS Glue without interrupting the existing Lake Formation permissions.

  When you update a data location registration to hybrid access mode, new principals can access the Data Catalog database pointing the Amazon S3 location using IAM permissions policies without interrupting existing users' Lake Formation permissions.

  Before updating the data location registration to enable hybrid access mode, you need to first opt in principals that are currently accessing the resource with Lake Formation permissions. This is to prevent potential interruption to the current workflow.
  You need to also grant `Super` permission on the tables in the database to the `IAMAllowedPrincipal` group.

Cross-account data sharing scenarios

- **Share AWS Glue resources using hybrid access mode** – In this scenario, the producer account has tables in a database that are currently shared with a consumer account using IAM permissions policies for Amazon S3 and AWS Glue actions. The data location of the database is not registered with Lake Formation.

Before registering the data location in hybrid access mode, you need to update the **Cross account version settings** to version 4. Version 4 provides the new AWS RAM permission policies required for cross-account sharing when `IAMAllowedPrincipal` group has `Super` permission on the resource. For those resources with `IAMAllowedPrincipal` group permissions, you can grant Lake Formation permissions to external accounts and opt them in to use Lake Formation permissions. The data lake administrator in the recipient account can grant Lake Formation permissions to principals in the account and opt them in to enforce the Lake Formation permissions.

- **Share Lake Formation resources using hybrid access mode** – Currently, the producer account has tables in a database that are shared with a consumer account enforcing Lake Formation permissions. The data location of the database is registered with Lake Formation.

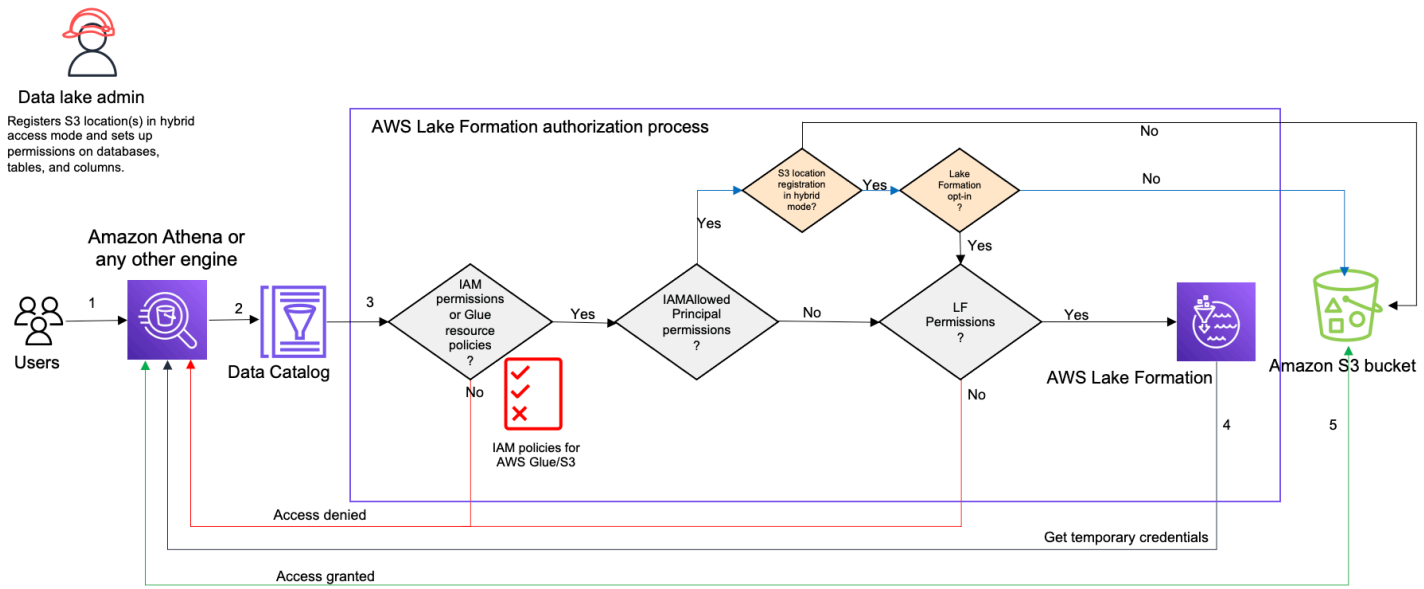
  In this case, you can update the Amazon S3 location registration to hybrid access mode, and share the data from Amazon S3 and metadata from Data Catalog using Amazon S3 bucket policies and Data Catalog resource policies to principals in the consumer account. You need to re-grant the existing Lake Formation permissions and opt in the principals before updating the Amazon S3 location registration. Also, you need to grant `Super` permission on the tables in the database to the `IAMAllowedPrincipals` group.

Topics

- How hybrid access mode works
- Setting up hybrid access mode - common scenarios
- Removing principals and resources from hybrid access mode
- Viewing principals and resources in hybrid access mode
- Additional resources

How hybrid access mode works

The following diagram shows how Lake Formation authorization works in hybrid access mode when you query the Data Catalog resources.



Before accessing data in your data lake, a data lake administrator or a user with administrative permissions sets up individual Data Catalog table user policies to allow or deny access to tables in your Data Catalog. Then, a principal who has the permissions to perform `RegisterResource` operation registers the Amazon S3 location of the table with Lake Formation in hybrid access mode. If a data location is not registered with Lake Formation, the administrator grants Lake Formation permissions to specific users on the Data Catalog databases and tables and opts them in to use Lake Formation permissions for those databases and tables in hybrid access mode.

1. **Submits a query** - A principal submits a query or an ETL script using an integrated service such as Amazon Athena, AWS Glue, Amazon EMR, or Amazon Redshift Spectrum.
2. **Requests data** - The integrated analytical engine identifies the table that is being requested and sends the metadata request to the Data Catalog (`GetTable`, `GetDatabase`).
3. **Checks permissions** - The Data Catalog verifies the querying principal's access permissions with Lake Formation.
   a. If the table doesn't have `IAMAllowedPrincipals` group permissions attached, Lake Formation permissions are enforced.
   b. If the principal has opted in to use Lake Formation permissions in the hybrid access mode, and the table has `IAMAllowedPrincipals` group permissions attached, Lake Formation permissions are enforced. The query engine applies the filters it received from Lake Formation and returns the data to the user.

c. If the table location is not registered with Lake Formation and the principal has not opted in to use Lake Formation permissions in hybrid access mode, the Data Catalog checks if the table has `IAMAllowedPrincipals` group permissions attached to it. If this permission exists on the table, all principals in the account gets `Super` or `All` permissions on the table.

   Lake Formation credential vending is not available, even when opted in, unless the data location is registered with Lake Formation.

4. **Get credentials** – The Data Catalog checks and lets the engine know if the table location is registered with Lake Formation or not. If the underlying data is registered with Lake Formation, the analytical engine requests Lake Formation for temporary credentials to access data in the Amazon S3 bucket.
5. **Get data** – If the principal is authorized to access the table data, Lake Formation provides temporary access to the integrated analytical engine. Using the temporary access, the analytical engine fetches the data from Amazon S3, and performs necessary filtering such as column, row, or cell filtering. When the engine finishes running the job, it returns the results back to the user. This process is called credential vending. For more information, see *Integrating with Lake Formation*.
6. If the data location of the table is not registered with Lake Formation, the second call from the analytic engine is made directly to Amazon S3. The concerned Amazon S3 bucket policy and IAM user policy are evaluated for data access. Whenever you use IAM policies, make sure that you follow IAM best practices. For more information, see Security best practices in IAM in the IAM User Guide.

Setting up hybrid access mode - common scenarios

As with Lake Formation permissions, you generally have two types of scenarios in which you can use hybrid access mode to manage data access: Provide access to principals within one AWS account and provide access to an external AWS account or principal.

This section provides instructions for setting up hybrid access mode in the following scenarios:

Manage permissions in hybrid access mode within one AWS account

- Converting an AWS Glue resource to a hybrid resource – You are currently providing access to tables in a database for all principals in your account using IAM permissions for Amazon S3 and AWS Glue but want to adopt Lake Formation to manage permissions incrementally.

- Converting a Lake Formation resource to a hybrid resource – You are currently using Lake Formation to manage access for tables in a database for all principals in your account but want to use Lake Formation only for specific principals. You want to provide access to new principals by using IAM permissions for AWS Glue and Amazon S3 on the same database and tables.

Manage permissions in hybrid access mode across AWS accounts

- Sharing an AWS Glue resource using hybrid access mode – You're currently not using Lake Formation to manage permissions for a table but want to apply Lake Formation permissions to provide access for principals in another account.
- Sharing a Lake Formation resource using hybrid access mode – You're using Lake Formation to manage access for a table but want to provide access for principals in another account by using IAM permissions for AWS Glue and Amazon S3 on the same database and tables.

Setting up hybrid access mode – High-level steps

1. Register the Amazon S3 data location with Lake Formation by selecting **Hybrid access mode**.
2. Principals must have `DATA_LOCATION` permission on a data lake location to create Data Catalog tables or databases that point to that location.
3. Set the **Cross-account version setting** to Version 4.
4. Grant fine-grained permissions to specific IAM users or roles on databases and tables. At the same time, make sure to set `Super` or `All` permissions to the `IAMAllowedPrincipals` group on the database and all or selected tables in the database.
5. Opt in the principals and resources. Other principals in the account can continue accessing the databases and tables using IAM permission policies for AWS Glue and Amazon S3 actions.
6. Optionally clean up IAM permission policies for Amazon S3 for the principals that are opted in to use Lake Formation permissions.

Prerequisites for setting up hybrid access mode

The following are the prerequisites for setting up hybrid access mode:

> **Note**
>
> We recommend that a Lake Formation administrator registers the Amazon S3 location in hybrid access mode, and opt in principals and resources.

1. Grant data location permission (`DATA_LOCATION_ACCESS`) to create Data Catalog resources that point to the Amazon S3 locations. Data location permissions control the ability to create Data Catalog catalogs, databases and tables that point to particular Amazon S3 locations.
2. To share Data Catalog resources with another account in hybrid access mode (without removing `IAMAllowedPrincipals` group permissions from the resource), you need to update the **Cross account version settings** to Version 4 or higher. To update the version using Lake Formation console, choose **Version 4** or **Version 5** under **Cross account version settings** on the **Data Catalog settings** page.

   You can also use the `put-data-lake-settings` AWS CLI command to set the `CROSS_ACCOUNT_VERSION` parameter to version 4 or 5:

   ```
   aws lakeformation put-data-lake-settings --region us-east-1 --data-lake-settings
    file://settings
   {
   "DataLakeAdmins": [
           {
   "DataLakePrincipalIdentifier": "arn:aws:iam::<111122223333>:user/<user-name>"
           }
       ],
       "CreateDatabaseDefaultPermissions": [],
       "CreateTableDefaultPermissions": [],
       "Parameters": {
   "CROSS_ACCOUNT_VERSION": "5"
       }
   }
   ```

3. To grant cross-account permissions in hybrid access mode, the grantor must have the required IAM permissions for AWS Glue and AWS RAM services. The AWS managed policy `AWSLakeFormationCrossAccountManager` grants the required permissions.

To enable cross-account data sharing in hybrid access mode, we've updated the `AWSLakeFormationCrossAccountManager` managed policy by adding two new IAM permissions:

- ram:ListResourceSharePermissions
- ram:AssociateResourceSharePermission

> **Note**
>
> If you are not using the AWS managed policy for the grantor role, add the above policies to your custom policies.

Amazon S3 bucket location and user access

When you create a catalog, database or a table in the AWS Glue Data Catalog, you can specify the Amazon S3 bucket location of the underlying data and register it with Lake Formation. The tables below describe how permissions work for AWS Glue and Lake Formation users (principals) based on the Amazon S3 data location of the table or database.

Amazon S3 location registered with Lake Formation

Amazon S3 location of a database	AWS Glue users	Lake Formation users
Registered with Lake Formation (in hybrid access mode or in Lake Formation mode)	Have read/write access to the Amazon S3 data location by inheriting permissions from the IAMAllowedPrincipals group (super access) permissions.	Inherit permissions to create tables from their granted CREATE TABLE permission.
No associated Amazon S3 location	Require explicit DATA LOCATION permission for running CREATE TABLE and INSERT TABLE statements.	Require explicit DATA LOCATION permission for running CREATE TABLE and INSERT TABLE statements.

IsRegisteredWithLakeFormation table property

The `IsRegisteredWithLakeFormation` property of a table indicates whether the data location of the table is registered with Lake Formation for the requester. If the permission mode of the location is registered as Lake Formation, then the `IsRegisteredWithLakeFormation` property is `true` for all users accessing the data location because all users are considered as opted in for that table. If the location is registered in hybrid access mode, then the value is set to `true` only for users who have opted in for that table.

How `IsRegisteredWithLakeFormation` works

Permission mode	Users/Roles	`IsRegisteredWithLakeFormation`	Description
Lake Formation	All	True	When a location is registered with Lake Formation, the `IsRegisteredWithLakeFormation` property will be set to true for all users. This means that the permissions defined in Lake Formation apply to the registered location. Credential vending will be done by Lake Formation.
Hybrid access mode	Opted in	True	For users who have opted in to using Lake Formation for data access and governance for a table, the `IsRegisteredWithLakeFormation`

Permission mode	Users/Roles	`IsRegisteredWithLakeFormation`	Description
			property will be set to `true` for that table. They are subject to the permission policies defined in Lake Formation for the registered location.
Hybrid access mode	Not opted in	False	For users who have not opted in to using Lake Formation permissions, the `IsRegisteredWithLakeFormation` property is set to `false`. They are not subject to the permission policies defined in Lake Formation for the registered location. Instead, users will follow the Amazon S3 permissions policies.

Converting an AWS Glue resource to a hybrid resource

Follow these steps to register an Amazon S3 location in hybrid access mode and on-board new Lake Formation users without interrupting the existing Data Catalog users' data access.

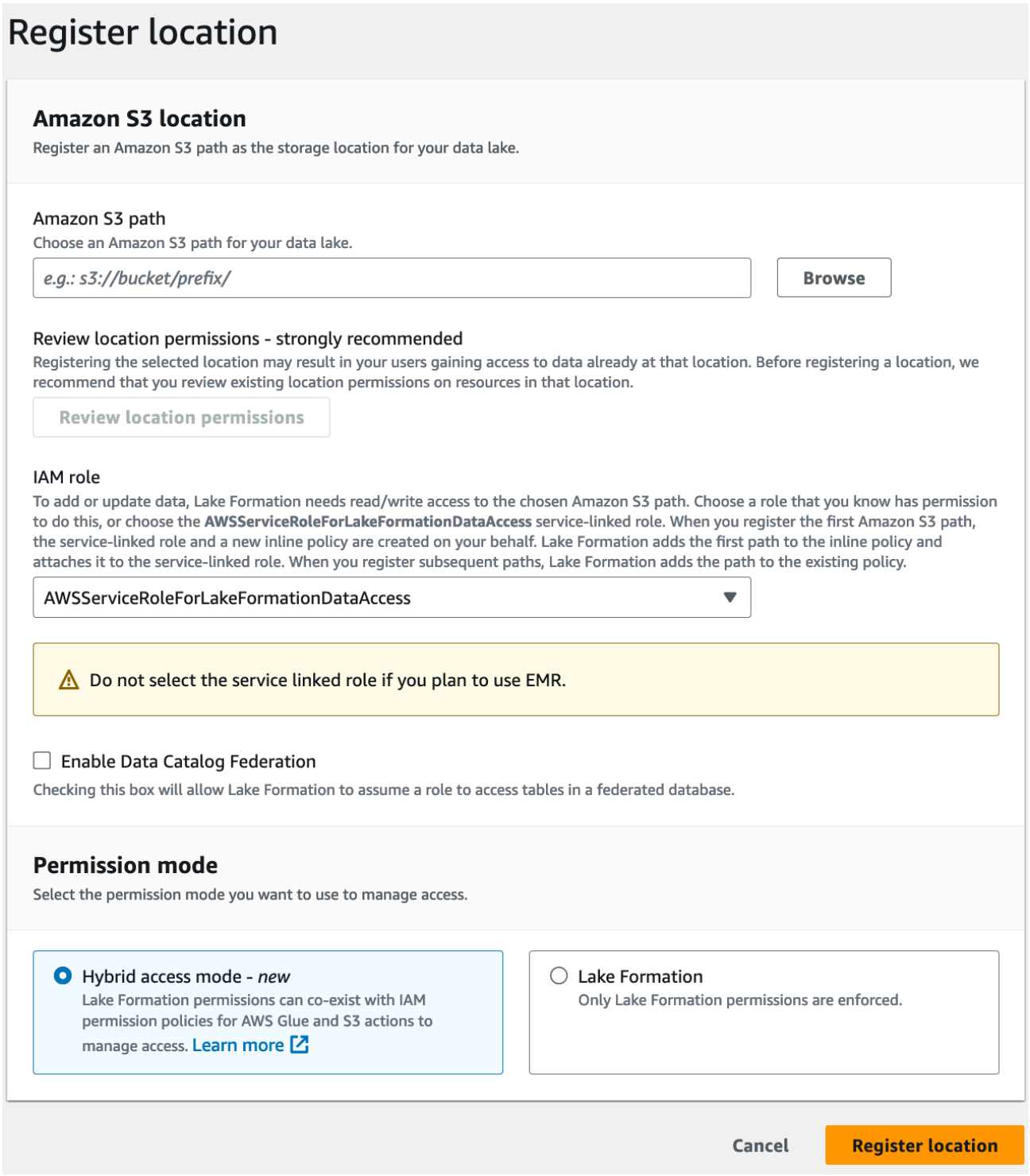
Scenario description - The data location is not registered with Lake Formation, and users' access to the Data Catalog database and tables is determined by IAM permissions policies for Amazon S3 and AWS Glue actions.
The `IAMAllowedPrincipals` group by default has `Super` permissions on all tables in the database.

To enable hybrid access mode for a data location that is not registered with Lake Formation

1. **Register an Amazon S3 location enabling hybrid access mode.**

   Console

   1. Sign in to the Lake Formation console as a data lake administrator.
   2. In the navigation pane, choose **Data lake locations** under **Administration**.
   3. Choose **Register location**.



4. On the **Register location** window, choose the **Amazon S3** path that you want to register with Lake Formation.
5. For **IAM role**, choose either the `AWSServiceRoleForLakeFormationDataAccess` service-linked role (the default) or a custom IAM role that meets the requirements in Requirements for roles used to register locations.

6. Choose **Hybrid access mode** to apply fine-grained Lake Formation access control policies to opt-in principals and Data Catalog databases and tables pointing to the registered location.

   Choose Lake Formation to allow Lake Formation to authorize access requests to the registered location.

7. Choose **Register location**.

AWS CLI

Following is an example for registering a data location with Lake Formation with HybridAccessEnabled:true/false. Default value for the `HybridAccessEnabled` parameter is false. Replace Amazon S3 path, role name, and AWS account id with valid values.

```
aws lakeformation register-resource --cli-input-json file:file path
json:
    {
        "ResourceArn": "arn:aws:s3:::s3-path",
        "UseServiceLinkedRole": false,
        "RoleArn": "arn:aws:iam::<123456789012>:role/<role-name>",
        "HybridAccessEnabled": true
    }
```

2. **Grant permissions and opt in principals to use Lake Formation permissions for resources in hybrid access mode**

Before you opt in principals and resources in hybrid access mode, verify that `Super` or `All` permissions to `IAMAllowedPrincipals` group exists on the databases and tables that have location registered with Lake Formation in hybrid access mode.

> **Note**
>
> You can't grant the `IAMAllowedPrincipals` group permission on `All tables` within a database. You need to select each table separately from the drop-down menu, and grant permissions. Also, when you create new tables in the database, you can choose to use the `Use only IAM access control for new tables in new databases` in the **Data Catalog Settings**. This option grants `Super` permission to the

`IAMAllowedPrincipals` group automatically when you create new tables within the database.

Console

1. On the Lake Formation console, under **Data Catalog**, choose **Catalogs**, **Databases**, or **Tables**.
2. Select a catalog, a database, or a table from the list, and choose **Grant** from the **Actions** menu.
3. Choose principals to grant permissions on the database, tables, and columns using named resource method or LF-Tags.

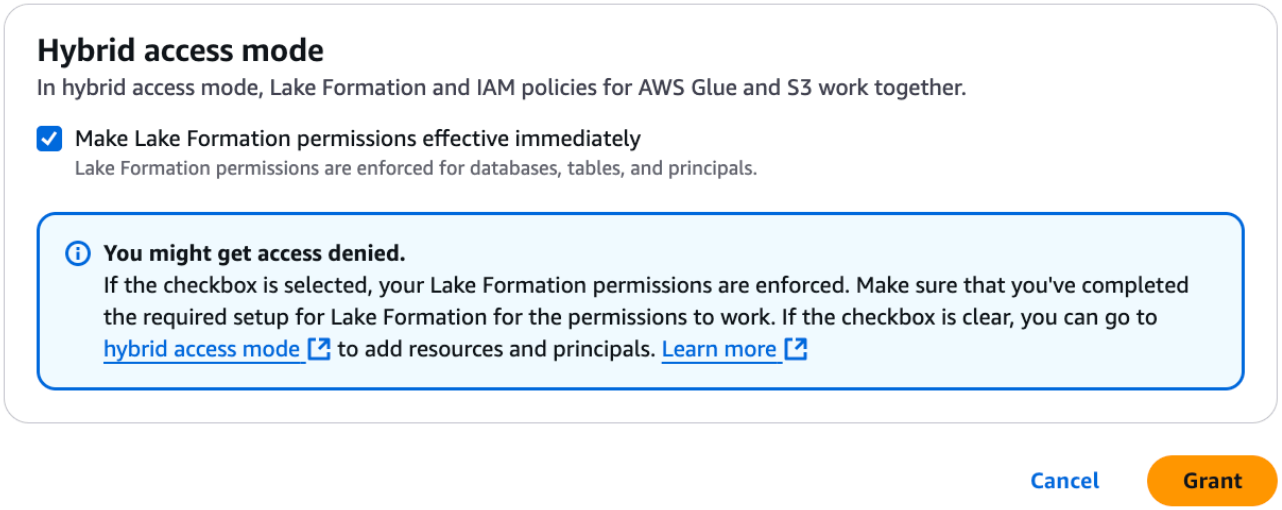
   Alternatively, choose **Data permissions**, select the principals to grant permissions from the list, and choose **Grant**.

   For more details on granting data permissions, see Granting permissions on Data Catalog resources.

   > **Note**
   >
   > If you're granting a principal Create table permission, you also need to grant data location permissions (`DATA_LOCATION_ACCESS`) to the principal. This permission is not needed to update tables.
   > For more information, see Granting data location permissions.

4. When you use **Named resource method** to grant permissions, the option to opt in principals and resources is available on the lower section of the **Grant data permission** page.

   Choose **Make Lake Formation permissions effective immediately** to enable Lake Formation permissions for the principals and resources.



5. Choose **Grant**.

   When you opt in principal A on table A that is pointing to a data location, it allows principal A to have access to this table's location using Lake Formation permissions if the data location is registered in hybrid mode.

AWS CLI

Following is an example for opting in a principal and a table in hybrid access mode. Replace the role name, AWS account id, database name, and table name with valid values.

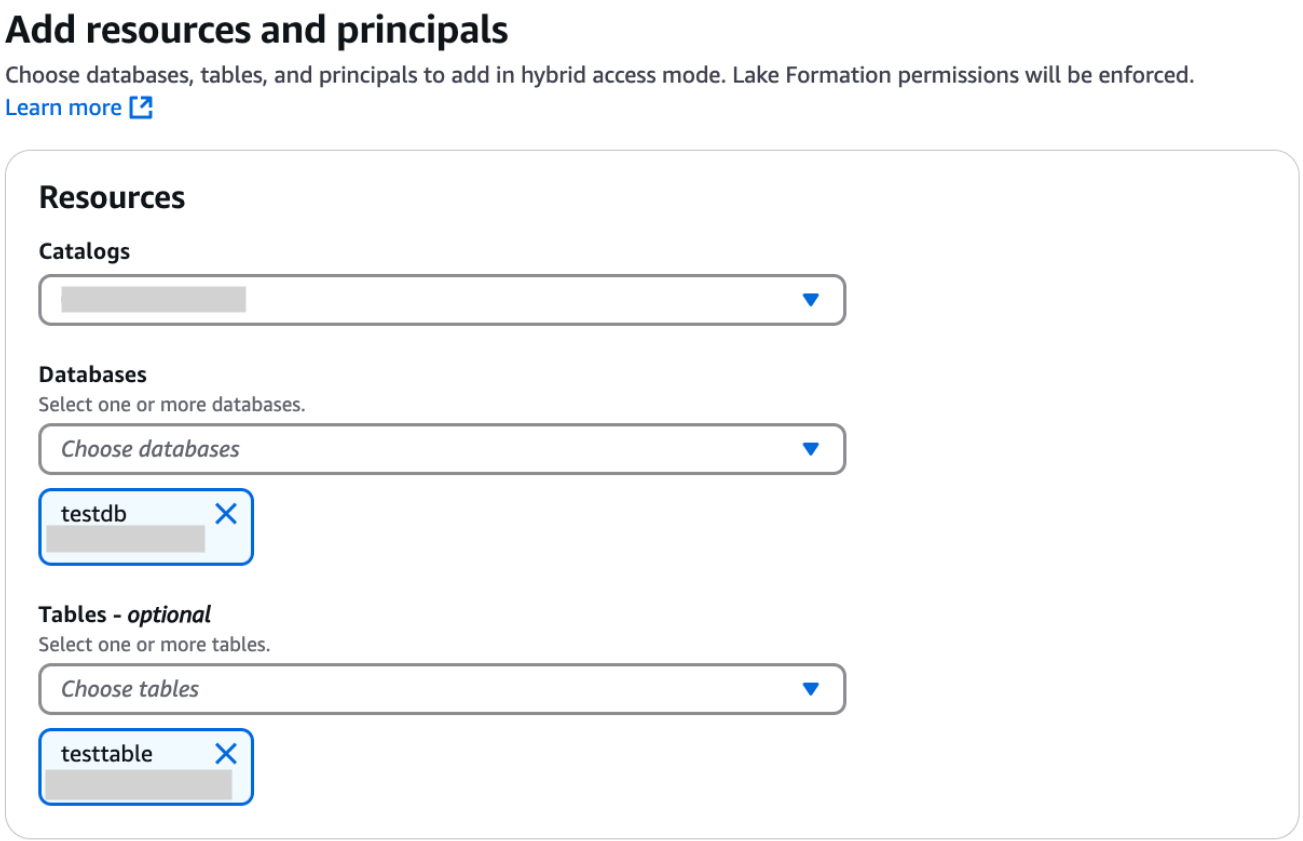
```
aws lakeformation create-lake-formation-opt-in --cli-input-json file://file path
json:
  {
        "Principal": {
            "DataLakePrincipalIdentifier":
 "arn:aws:iam::<123456789012>:role/<hybrid-access-role>"
        },
        "Resource": {
            "Table": {
                "CatalogId": "<123456789012>",
                "DatabaseName": "<hybrid_test>",
                "Name": "<hybrid_test_table>"
            }
        }
    }
```

a. (Optional) If you choose LF-Tags to grant permissions, you can opt in principals to use Lake Formation permissions in a separate step. You can do this by choosing **Hybrid access mode** under **Permissions** from the left navigation bar.

b. On the lower section of the **Hybrid access mode** page, choose **Add** to add resources and principals to hybrid access mode.

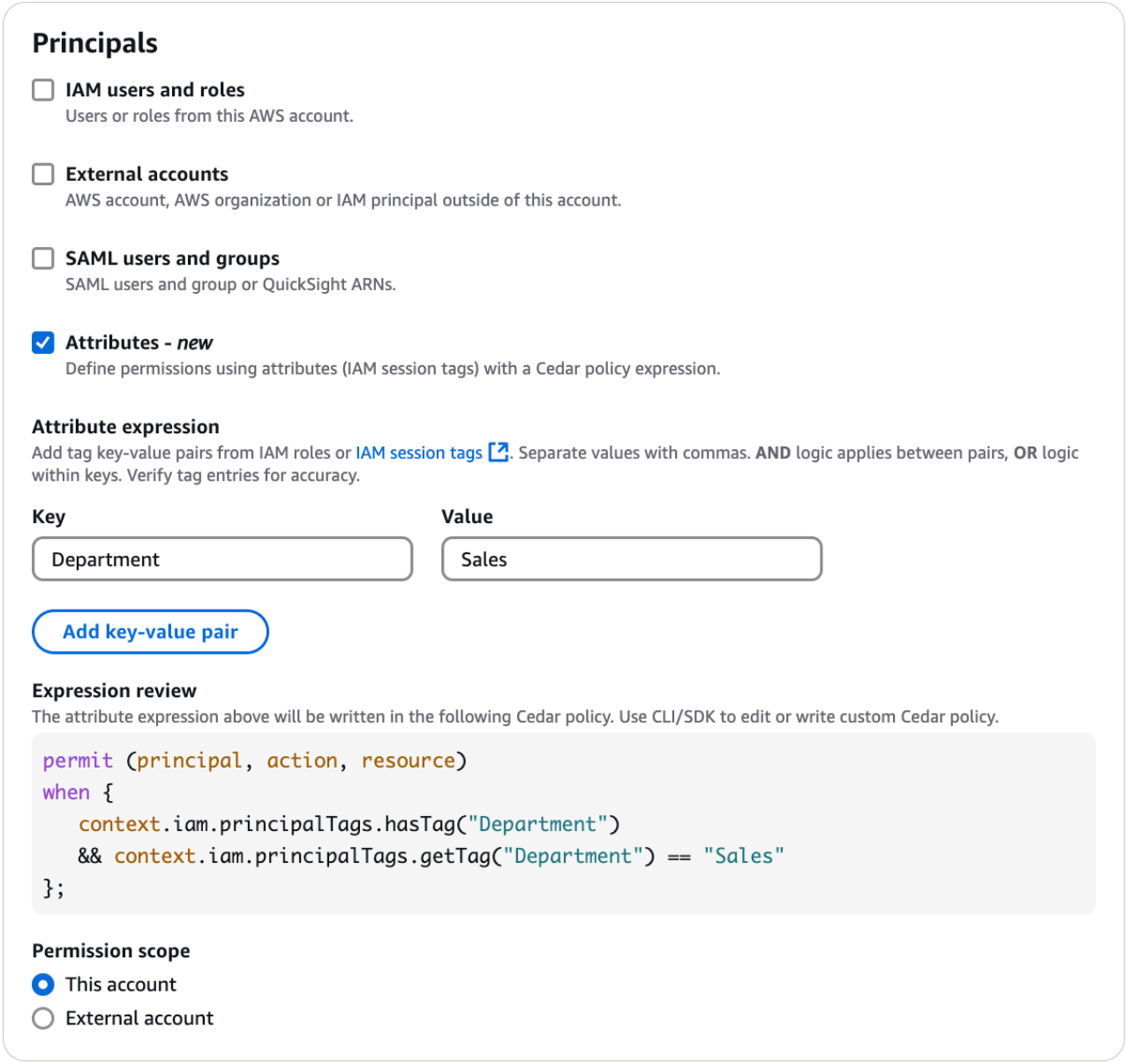
c. On the **Add resources and principals** page, choose the catalogs, databases and tables registered in hybrid access mode.

You can choose `All tables` under a database to grant access.



d. Choose principals opt in to use Lake Formation permissions in hybrid access mode.

- **Principals** – You can choose IAM users and roles in the same account or in another account. You can also choose SAML users and groups.
- **Attributes** – Select attributes to grant permissions based on attributes.



- Enter the key-value pair to create a grant based on attributes. Review the Cedar policy expression on the console. For more information about Cedar, see [What is Cedar? | Cedar Policy Language Reference GuideLink].
- Choose **Add**.

  All IAM roles/users with matching attributes are granted access.

e. Choose **Add**.

Converting a Lake Formation resource to a hybrid resource

In cases where you're currently using Lake Formation permissions for your Data Catalog databases and tables, you can edit the location registration properties to enable hybrid access mode. This allows you to provide new principals access to the same resources using IAM permission policies for Amazon S3 and AWS Glue actions without interrupting existing Lake Formation permissions.

Scenario description - The following steps assume that you've a data location registered with Lake Formation, and you've set up permissions for principals on databases, tables, or columns pointing to that location. If the location was registered with a service linked role, you can't update the location parameters and enable hybrid access mode. The `IAMAllowedPrincipals` group by default has Super permissions on the database and all its tables.

> **Important**
>
> Don't update a location registration to hybrid access mode without opting in the principals that are accessing data in this location.

Enabling hybrid access mode for a data location registered with Lake Formation

1.
> **Warning**
>
> We don't recommend converting a Lake Formation managed data location to hybrid access mode to avoid interrupting the permission policies of other existing users or workloads.

   Opt in the existing principals who have Lake Formation permissions.

   1. List and review the permissions you've granted to principals on catalogs, databases and tables. For more information, see Viewing database and table permissions in Lake Formation.
   2. Choose **Hybrid access mode** under **Permissions** from the left navigation bar, and choose **Add**.
   3. On the **Add principals and resources** page, choose the catalogs, databases, and tables from the Amazon S3 data location that you want to use in hybrid access mode. Choose the principals that already have Lake Formation permissions.

4. Choose **Add** to opt in the principals to use Lake Formation permissions in hybrid access mode.

2. Update the Amazon S3 bucket/prefix registration by choosing **Hybrid access mode** option.

   Console

   1. Sign in to the Lake Formation console as the data lake administrator.
   2. In the navigation pane, under **Register and Ingest**, choose **Data lake locations**.
   3. Select a location, and on the **Actions**menu, choose **Edit**.
   4. Choose **Hybrid access mode**.
   5. Choose **Save**.
   6. Under Data Catalog, select the database or table and grant `Super` or `All` permissions to the virtual group called `IAMAllowedPrincipals`.
   7. Verify that your existing Lake Formation users' access is not interrupted when you updated the location registration properties. Sign in to Athena console as a Lake Formation principal and run a sample query on a table that is pointing to the updated location.

      Similarly, verify the access of AWS Glue users who are using IAM permissions policies to access the database and tables.

   AWS CLI

   Following is an example for registering a data location with Lake Formation with HybridAccessEnabled:true/false. Default value for the `HybridAccessEnabled` parameter is false. Replace Amazon S3 path, role name, and AWS account id with valid values.

   ```
   aws lakeformation update-resource --cli-input-json file://file path
   json:
   {
       "ResourceArn": "arn:aws:s3:::<s3-path>",
       "RoleArn": "arn:aws:iam::<123456789012>:role/<test>",
       "HybridAccessEnabled": true
   }
   ```

Sharing an AWS Glue resource using hybrid access mode

Share data with another AWS account or a principal in another AWS account enforcing Lake Formation permissions without interrupting existing Data Catalog users' IAM based access.

Scenario description - The producer account has a Data Catalog database that has access controlled using IAM principal policies for Amazon S3 and AWS Glue actions. The data location of the database is not registered with Lake Formation. The `IAMAllowedPrincipals` group, by default, has `Super` permissions on the database and all its tables.

Granting cross-account Lake Formation permissions in hybrid access mode

1. **Producer account set up**

    1. Sign in to the Lake Formation console using a role that has `lakeformation:PutDataLakeSettings` IAM permission.
    2. Go to **Data Catalog settings**, and choose `Version 4` for the **Cross account version settings**.

        If you're currently using version 1 or 2, see Updating cross-account data sharing version settings instructions on updating to version 3.

        There are no permission policy changes required when upgrading from version 3 to 4.
    3. Register the Amazon S3 location of the database or table that you're planning to share in hybrid access mode.
    4. Verify that `Super` permission to the `IAMAllowedPrincipals` group exists on the databases and tables of which you registered the data location in hybrid access mode in the above step.
    5. Grant Lake Formation permissions to AWS organizations, organizational units (OUs), or directly with an IAM principal in another account.
    6. If you're granting permissions directly to an IAM principal, opt in the principal from the consumer account to enforce Lake Formation permissions in hybrid access mode by enabling the option **Make Lake Formation permissions effective immediately**.

        If you're granting cross-account permissions to another AWS account, when you opt in the account, Lake Formation permissions are enforced only for the admins of that account. The recipient account data lake administrator need to cascade down the permissions and opt in

the principals in the account to enforce Lake Formation permissions for the shared resources that are in hybrid access mode.

If you choose **Resources matched by LF-Tags** option to grant cross-account permissions, you need to first complete granting permissions step. You can opt in principals and resources to hybrid access mode as a separate step by choosing **Hybrid access mode** under Permissions on the left-navigation bar of the Lake Formation console. Then choose **Add** to add the resources and principals that you want to enforce Lake Formation permissions.

2. **Consumer account set up**

   1. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as a data lake administrator.
   2. Go to https://console.aws.amazon.com/ram/home, and accept the resource share invitation. The **Shared with me** tab in the AWS RAM console displays the database and tables that are shared with your account.
   3. Create a resource link to the shared database and/or table in Lake Formation.
   4. Grant `Describe` permission on resource link and `Grant on target` permission (on the original shared resource) to the IAM principals in your (consumer) account.
   5. Grant Lake Formation permissions on the database or table shared with you to the principals in your account. Opt in the principals and resources to enforce Lake Formation permissions in hybrid access mode by enabling the option **Make Lake Formation permissions effective immediately**.
   6. Test the principal's Lake Formation permissions by running sample Athena queries. Test the existing access of your AWS Glue users with IAM principal policies for Amazon S3 and AWS Glue actions.

      (Optional) Remove the Amazon S3 bucket policy for data access and IAM principal policies for AWS Glue and Amazon S3 data access for the principals that you configured to use Lake Formation permissions.

Sharing a Lake Formation resource using hybrid access mode

Allow new Data Catalog users in an external account to access Data Catalog databases and tables using IAM based policies without interrupting the existing Lake Formation cross-account sharing permissions.

Scenario description - The producer account has Lake Formation managed database and tables that are shared with an external (consumer) account at account-level or IAM principal-level. The data location of the database is registered with Lake Formation. The `IAMAllowedPrincipals` group does not have `Super` permissions on the database and its tables.

Granting cross-account access to new Data Catalog users via IAM based policies without interrupting existing Lake Formation permissions

1. **Producer account set up**

   1. Sign in to the Lake Formation console using a role that `lakeformation:PutDataLakeSettings`.
   2. Under **Data Catalog settings**, choose `Version 4` for the **Cross account version settings**.

      If you're currently using version 1 or 2, see Updating cross-account data sharing version settings instructions on updating to version 3.

      There are no permission policy changes required to upgrade from version 3 to 4.
   3. List the permissions you've granted to principals on databases and tables. For more information, see Viewing database and table permissions in Lake Formation.
   4. Regrant existing Lake Formation cross- account permissions by opting in principals and resources.

      > **Note**
      >
      > Before updating a data location registration to hybrid access mode to grant cross-account permissions, you need to regrant at least one cross-account data share per account. This step is necessary to update the AWS RAM managed permissions attached to the AWS RAM resource share.
      > In July 2023, Lake Formation has updated the AWS RAM managed permissions used for sharing databases and tables:
      >
      > - `arn:aws:ram::aws:permission/AWSRAMLFEnabledGlueAllTablesReadWriteForDatabase` (database-level share policy)
      > - `arn:aws:ram::aws:permission/AWSRAMLFEnabledGlueTableReadWrite` (table-level share policy)

The cross-account permission grants made before July 2023 don't have these updated AWS RAM permissions.
If you've granted cross-account permissions directly to principals, you need to individually regrant those permissions to the principals. If you skip this step, the principals accessing the shared resource might get an illegal combination error.

5. Go to https://console.aws.amazon.com/ram/home.
6. The **Shared by me** tab in the AWS RAM console displays the database and table names that you've shared with an external account or principal.

   Ensure that the permissions attached to the shared resource has the correct ARN.
7. Verify the resources in the AWS RAM share are in `Associated` status. If the status shows as `Associating`, wait until they go into `Associated` state. If the status becomes `Failed`, stop and contact Lake Formation service team.
8. Choose **Hybrid access mode** under **Permissions** from the left navigation bar, and choose **Add**.
9. The **Add principals and resources** page shows the databases, and/or tables and the principals that have access. You can make the required updates by adding or removing principals and resources.
10. Choose the principals with Lake Formation permissions for the database and tables that you want to change to hybrid access mode. Choose the databases and tables.
11. Choose **Add** to opt in the principals to enforce Lake Formation permissions in hybrid access mode.
12. Grant `Super` permission to the virtual group `IAMAllowedPrincipals` on your database and selected tables.
13. Edit the Amazon S3 location Lake Formation registration to hybrid access mode.
14. Grant permissions for the AWS Glue users in the external (consumer) account using IAM permission policies for Amazon S3 AWS Glue actions.

2. **Consumer account set up**

   1. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/ as a data lake administrator.
   2. Go to https://console.aws.amazon.com/ram/home and accept the resource share invitation. The **Resources shared with me** tab in the AWS RAM page displays the database and table names that are shared with your account.

For the AWS RAM share, ensure that the attached permission has the correct ARN of the shared AWS RAM invite. Check if the resources in the AWS RAM share are in `Associated` status. If the status shows as `Associating`, wait until they go into `Associated` state. If the status becomes `Failed`, stop and contact Lake Formation service team.

3. Create a resource link to the shared database and/or table in Lake Formation.
4. Grant `Describe` permission on resource link and `Grant on target` permission (on the original shared resource) to the IAM principals in your (consumer) account.
5. Next, set up Lake Formation permissions for principals in your account on the shared database or table.

   On the left navigation bar, under **Permissions**, choose **Hybrid access mode**.
6. Choose **Add** in the lower section of the **Hybrid access mode** page to opt in the principals and the database or table shared with you from the producer account.
7. Grant permissions for the AWS Glue users in your account using IAM permission policies for Amazon S3 AWS Glue actions.
8. Test users' Lake Formation permissions and AWS Glue permissions by running separate sample queries on the table using Athena

   (Optional) Clean up IAM permission policies for Amazon S3 for the principals that are in the hybrid access mode.

Removing principals and resources from hybrid access mode

Follow these steps to remove databases, tables, and principals from hybrid access mode.

Console

1. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. Under **Permissions**, choose **Hybrid access mode**.
3. On the **Hybrid access mode** page, select the checkbox next to the database or table name and choose `Remove`.
4. A warning message prompts you to confirm the action. Choose **Remove**.

Lake Formation no longer enforces permissions for those resources, and access to this resource will be controlled using IAM and AWS Glue permissions. This may cause the user to no longer have access to this resource if they don't have the appropriate IAM permissions.

AWS CLI

The following example shows how to remove resources from hybrid access mode.

```
aws lakeformation delete-lake-formation-opt-in --cli-input-json file://file path

json:
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::<123456789012>:role/role name"
    },
    "Resource": {
        "Table": {
            "CatalogId": "<123456789012>",
            "DatabaseName": "<database name>",
            "Name": "<table name>"
          }
    }
}
```

Viewing principals and resources in hybrid access mode

Follow these steps to view databases, tables, and principals in hybrid access mode.

Console

1. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. Under **Permissions**, choose **Hybrid access mode**.
3. The **Hybrid access mode** page shows the resources and principals that are currently in hybrid access mode..

AWS CLI

The following example shows how to list all opt in principals and resources that are in hybrid access mode.

```
aws lakeformation list-lake-formation-opt-ins
```

The following example shows how to list opt in for a specific principal-resource pair.

```
aws lakeformation list-lake-formation-opt-ins --cli-input-json file://file path

json:
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::<account-id>:role/<role name>"
    },
    "Resource": {
        "Table": {
            "CatalogId": "<account-id>",
            "DatabaseName": "<database name>",
            "Name": "<table name>"
          }
    }
}
```

Additional resources

In the following blog post, we walk you through the instructions to onboard Lake Formation permissions in hybrid access mode for selected users while the database is already accessible to other users through IAM and Amazon S3 permissions. We will review the instructions to set up hybrid access mode within an AWS account and between two accounts.

- Introducing hybrid access mode for AWS Glue Data Catalog to secure access using Lake Formation and IAM and Amazon S3 policies.

Creating objects in the AWS Glue Data Catalog

AWS Lake Formation uses the AWS Glue Data Catalog (Data Catalog) to store metadata about data lakes, data sources, transforms, and targets. Metadata is data about the underlying data in your dataset. Each AWS account has one Data Catalog per AWS Region.

Metadata in the Data Catalog is organized in a three-level data hierarchy comprising catalogs, databases, and tables. It organizes data from various sources into logical containers called catalogs. Each catalog represents data from sources like Amazon Redshift data warehouses, Amazon DynamoDB databases, and third-party data sources such as Snowflake, MySQL, and over 30 external data sources, which are integrated through federated connectors. You can also create new catalogs in the Data Catalog to store data in S3 Table Buckets or Redshift Managed Storage (RMS).

Tables store information about the underlying data, including schema information, partition information, and data location. Databases are collections of tables. The Data Catalog also contains resource links, which are links to shared catalogs, databases and tables in external accounts, and are used for cross-account access to data in the data lake.

The Data Catalog is a nested catalog object that contains catalogs, databases and tables. It is referenced by the AWS account ID, and is the default catalog in an account and an AWS Region. The Data Catalog uses a three-level hierarchy (catalog.database.table) to organize tables.

- Catalog – The top-most level of Data Catalog's three level metadata hierarchy. You can add multiple catalogs in a Data Catalog through federation.
- Database – The second level of the metadata hierarchy comprising of tables and views. A database is also referred to as a schema in many data systems like Amazon Redshift and Trino.
- Table and view – The third-level of the Data Catalog's 3-level data hierarchy.

All Iceberg tables in Amazon S3 are stored in the default Data Catalog having Catalog ID = AWS account ID. You can create federated catalogs in AWS Glue Data Catalog that store definitions of tables in Amazon Redshift, Amazon S3 Table storage, or other third-party data sources through federation.

Topics

- Creating a catalog
- Creating a database
- Creating tables

- Building AWS Glue Data Catalog views

Creating a catalog

Catalogs represent the highest or top-most level in the three-level metadata hierarchy of the AWS Glue Data Catalog. You can use multiple methods to bring data into the Data Catalog and create multi-level catalogs.

For more information on creating catalogs from external data sources, see Bringing your data into the AWS Glue Data Catalog.

To create a catalog using the Lake Formation console, you must be signed in as a data lake administrator or a *catalog creator*. A catalog creator is a principal who has been granted the Lake Formation `CREATE_CATALOG` permission. You can see a list of catalog creators on the **Administrative roles and tasks** page of the Lake Formation console. To view this list, you must have the `lakeformation:ListPermissions` IAM permission and be signed in as a data lake administrator or as a catalog creator with the grant option on the `CREATE_CATALOG` permission.

Creating a database

Metadata tables in the Data Catalog are stored within databases. You can create as many databases as you need, and you can grant different Lake Formation permissions on each database.

Databases can have an optional location property. This location is typically within an Amazon Simple Storage Service (Amazon S3) location that is registered with Lake Formation. When you specify a location, principals do not need data location permissions to create Data Catalog tables that point to locations within the database location. For more information, see Underlying data access control.

To create a database using the Lake Formation console, you must be signed in as a data lake administrator or *database creator*. A database creator is a principal who has been granted the Lake Formation `CREATE_DATABASE` permission. You can see a list of database creators on the **Administrative roles and tasks** page of the Lake Formation console. To view this list, you must have the `lakeformation:ListPermissions` IAM permission and be signed in as a data lake administrator or as a database creator with the grant option on the `CREATE_DATABASE` permission.

To create a database

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator or database creator.
2. In the navigation pane, under **Data catalog**, choose **Databases**.
3. Choose **Create database**.
4. In the **Create database** dialog box, enter a database name, optional location, and optional description.
5. Optionally select **Use only IAM access control for new tables in this database**.

   For information about this option, see the section called "Changing the default settings for your data lake".
6. Choose **Create database**.

Creating tables

AWS Lake Formation metadata tables contain information about data in the data lake, including schema information, partition information, and data location. These tables are stored in the AWS Glue Data Catalog. You use them to access underlying data in the data lake and manage that data with Lake Formation permissions. Tables are stored within databases in the Data Catalog.

There are several ways to create Data Catalog tables:

- Run a crawler in AWS Glue. See Defining crawlers in the *AWS Glue Developer Guide*.
- Create and run a workflow. See the section called "Importing data using workflows".
- Create a table manually using the Lake Formation console, AWS Glue API, or AWS Command Line Interface (AWS CLI).
- Create a table using Amazon Athena.
- Create a resource link to a table in an external account. See the section called "Creating resource links".

Creating Apache Iceberg tables

AWS Lake Formation supports creating Apache Iceberg tables that use the Apache Parquet data format in the AWS Glue Data Catalog with data residing in Amazon S3. A table in the Data Catalog

is the metadata definition that represents the data in a data store. By default, Lake Formation creates Iceberg v2 tables. For the difference between v1 and v2 tables, see Format version changes in the Apache Iceberg documentation.

Apache Iceberg is an open table format for very large analytic datasets. Iceberg allows for easy changes to your schema, also known as schema evolution, meaning that users can add, rename, or remove columns from a data table without disrupting the underlying data. Iceberg also provides support for data versioning, which allows users to track changes to data overtime. This enables the time travel feature, which allows users to access and query historical versions of data and analyze changes to the data between updates and deletes.

You can use Lake Formation console or the `CreateTable` operation in the AWS Glue API to create an Iceberg table in the Data Catalog. For more information, see CreateTable action (Python: create_table).

When you create an Iceberg table in the Data Catalog, you must specify the table format and metadata file path in Amazon S3 to be able to perform reads and writes.

You can use Lake Formation to secure your Iceberg table using fine-grained access control permissions when you register the Amazon S3 data location with AWS Lake Formation. For source data in Amazon S3 and metadata that is not registered with Lake Formation, access is determined by IAM permissions policies for Amazon S3 and AWS Glue actions. For more information, see Managing Lake Formation permissions.

> **Note**
>
> Data Catalog doesn't support creating partitions and adding Iceberg table properties.

Topics

- Prerequisites
- Creating an Iceberg table

Prerequisites

To create Iceberg tables in the Data Catalog, and set up Lake Formation data access permissions, you need to complete the following requirements:

1. **Permissions required to create Iceberg tables without the data registered with Lake Formation.**

   In addition to the permissions required to create a table in the Data Catalog, the table creator requires the following permissions:

   - `s3:PutObject` on resource arn:aws:s3:::{bucketName}
   - `s3:GetObject` on resource arn:aws:s3:::{bucketName}
   - `s3:DeleteObjecton` resource arn:aws:s3:::{bucketName}

2. **Permissions required to create Iceberg tables with data registered with Lake Formation:**

   To use Lake Formation to manage and secure the data in your data lake, register your Amazon S3 location that has the data for tables with Lake Formation. This is so that Lake Formation can vend credentials to AWS analytical services such as Athena, Redshift Spectrum, and Amazon EMR to access data. For more information on registering an Amazon S3 location, see Adding an Amazon S3 location to your data lake.

   A principal who reads and writes the underlying data that is registered with Lake Formation requires the following permissions:

   - `lakeformation:GetDataAccess`
   - `DATA_LOCATION_ACCESS`

     A principal who has data location permissions on a location also has location permissions on all child locations.

     For more information on data location permissions, see Underlying data access control.

To enable compaction, the service needs to assume an IAM role that has permissions to update tables in the Data Catalog. For details, see Table optimization prerequisites.

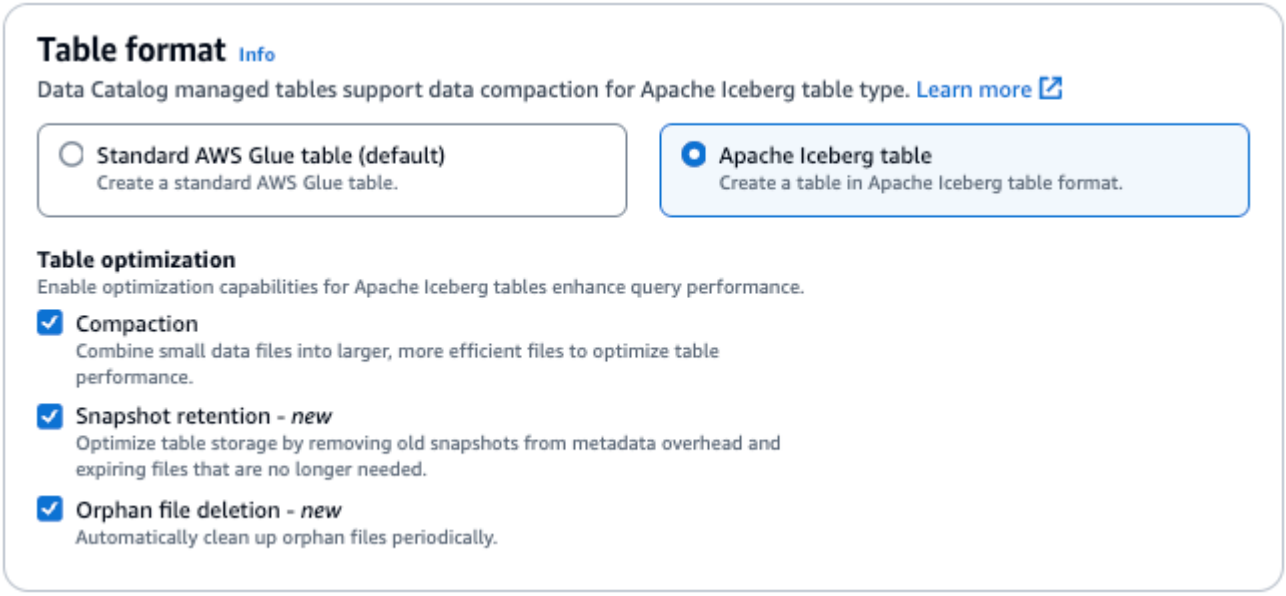
Creating an Iceberg table

You can create Iceberg v1 and v2 tables using Lake Formation console or AWS Command Line Interface as documented on this page. You can also create Iceberg tables using AWS Glue console or AWS Glue crawler. For more information, see Data Catalog and Crawlers in the AWS Glue Developer Guide.

To create an Iceberg table

Console

1. Sign in to the AWS Management Console, and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. Under Data Catalog, choose **Tables**, and use the **Create table** button to specify the following attributes:
   - **Table name**: Enter a name for the table. If you're using Athena to access tables, use these naming tips in the Amazon Athena User Guide.
   - **Database**: Choose an existing database or create a new one.
   - **Description**:The description of the table. You can write a description to help you understand the contents of the table.
   - **Table format**: For **Table format**, choose Apache Iceberg.



   - **Table optimization**
     - **Compaction** – Data files are merged and rewritten remove obsolete data and consolidate fragmented data into larger, more efficient files.
     - **Snapshot retention** – Snapshots are timestamped versions of an Iceberg table. Snapshot retention configurations allow customers to enforce how long to retain snapshots and how many snapshots to retain. Configuring a snapshot retention optimizer can help manage storage overhead by removing older, unnecessary snapshots and their associated underlying files.
     - **Orphan file deletion** – Orphan files are files that are no longer referenced by the Iceberg table metadata. These files can accumulate over time, especially after operations like

table deletions or failed ETL jobs. Enabling orphan file deletion allows AWS Glue to periodically identify and remove these unnecessary files, freeing up storage.

For more information, see Optimizing Iceberg tables.

- **IAM role**: To run compaction, the service assumes an IAM role on your behalf. You can choose an IAM role using the drop-down. Ensure that the role has the permissions required to enable compaction.

  To learn more about the required permissions, see Table optimization prerequisites.

- **Location**: Specify the path to the folder in Amazon S3 that stores the metadata table. Iceberg needs a metadata file and location in the Data Catalog to be able to perform reads and writes.

- **Schema**: Choose **Add columns** to add columns and data types of the columns. You have the option to create an empty table and update the schema later. Data Catalog supports Hive data types. For more information, see Hive data types.

  Iceberg allows you to evolve schema and partition after you create the table. You can use Athena queries to update the table schema and Spark queries for updating partitions.

AWS CLI

```
aws glue create-table \
    --database-name iceberg-db \
    --region us-west-2 \
    --open-table-format-input '{
      "IcebergInput": {
           "MetadataOperation": "CREATE",
           "Version": "2"
         }
      }' \
    --table-input '{"Name":"test-iceberg-input-demo",
            "TableType": "EXTERNAL_TABLE",
            "StorageDescriptor":{
               "Columns":[
                   {"Name":"col1", "Type":"int"},
                   {"Name":"col2", "Type":"int"},
                   {"Name":"col3", "Type":"string"}
                ],
               "Location":"s3://DOC_EXAMPLE_BUCKET_ICEBERG/"
            }
```

```
        }'
```

Optimizing Iceberg tables

Lake Formation supports multiple table optimization options to enhance the management and performance of Apache Iceberg tables used by the AWS analytical engines and ETL jobs. These optimizers provide efficient storage utilization, improved query performance, and effective data management. There are three types of table optimizers available in Lake Formation:

- **Compaction** – Data compaction compacts small data files to reduce storage usage and improve read performance. Data files are merged and rewritten to remove obsolete data and consolidate fragmented data into larger, more efficient files. Compaction can be configured to run automatically or manually triggered as needed.
- **Snapshot retention** – Snapshots are timestamped versions of an Iceberg table. Snapshot retention configurations allow customers to enforce how long to retain snapshots and how many snapshots to retain. Configuring a snapshot retention optimizer can help manage storage overhead by removing older, unnecessary snapshots and their associated underlying files.
- **Orphan file deletion** – Orphan files are files that are no longer referenced by the Iceberg table metadata. These files can accumulate over time, especially after operations like table deletions or failed ETL jobs. Enabling orphan file deletion allows AWS Glue to periodically identify and remove these unnecessary files, freeing up storage.

You can enable or disable compaction, snapshot retention, and orphan file deletion optimizers for individual Iceberg tables in the Data Catalog using the AWS Glue console, AWS CLI, or AWS Glue API operations.

For more information, see Optimizing Iceberg tables in the AWS Glue Developer Guide.

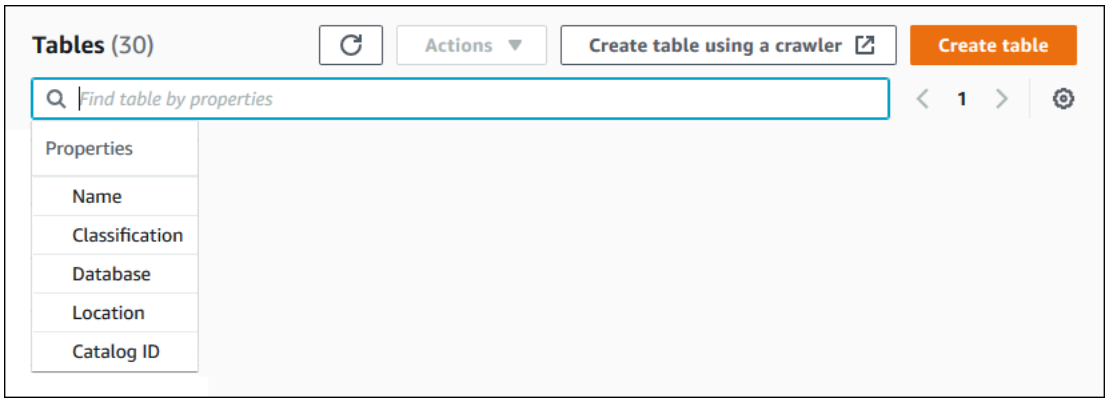
Searching for tables

You can use the AWS Lake Formation console to search for Data Catalog tables by name, location, containing database, and more. The search results show only the tables that you have Lake Formation permissions on.

To search for tables (console)

1. Sign in to the AWS Management Console and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, choose **Tables**.
3. Position the cursor in the search field. The field has the placeholder text *Find table by properties*.

   The **Properties** menu appears, showing the various table properties to search by.



4. Do one of the following:

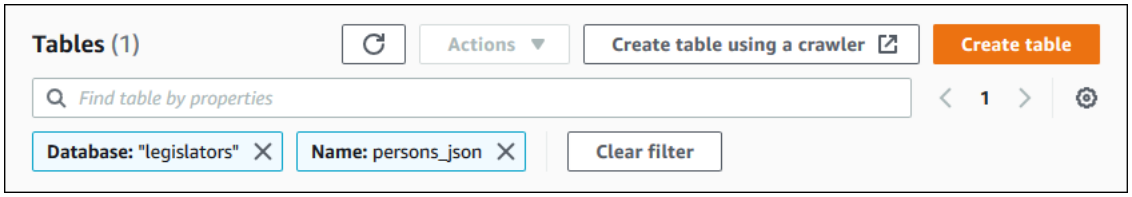
   - Search by containing database.

     1. Choose **Database** from the **Properties** menu, and then either choose a database from the **Databases** menu that appears or type a database name and press **Enter**.

        The tables that you have permissions on in the database are listed.

     2. (Optional) To narrow down the list to a single table in the database, position the cursor in the search field again, choose **Name** from the **Properties** menu, and either choose a table name from the **Tables** menu that appears or type a table name and press **Enter**.

        The single table is listed, and both the database name and table name appear as tiles under the search field.



        To adjust the filter, close either of the tiles or choose **Clear filter**.

- Search by other properties.
  1. Choose a search property from the **Properties** menu.

     To search by AWS account ID, choose **Catalog ID** from the **Properties** menu, enter a valid AWS account ID (for example, 111122223333), and press **Enter**.

     To search by location, choose **Location** from the **Properties** menu, and select a location from the **Locations** menu that appears. All tables in the root location of the selected location (for example, Amazon S3) are returned.

Searching tables using AWS CLI

- The following example shows how to run a partial serach. The `--search-text` parameter allows you to search for tables that contain the specified text in their metadata. In this case, it returns all tables that have "customer" in their name, description, or other metadata fields.

```
aws glue search-tables
      --search-text "customer"
      --region AWS Region
      --max-results 10
      --sort-criteria "FieldName=Name,Sort=ASC"
```

Sharing Data Catalog tables and databases across AWS Accounts

You can share Data Catalog resources (databases and tables) with external AWS accounts by granting Lake Formation permissions on the resources to the external accounts. Users can then run queries and jobs that join and query tables across multiple accounts. With some restrictions, when you share a Data Catalog resource with another account, principals in that account can operate on that resource as if the resource were in their Data Catalog.

You don't share resources with specific principals in external AWS accounts—you share the resources with an AWS account or organization. When you share a resource with an AWS organization, you're sharing the resource with all accounts at all levels in that organization. The data lake administrator in each external account must then grant permissions on the shared resources to principals in their account.

For more information, see Cross-account data sharing in Lake Formation and Granting permissions on Data Catalog resources.

See Also:

- Accessing and viewing shared Data Catalog tables and databases
- Prerequisites

Building AWS Glue Data Catalog views

In the AWS Glue Data Catalog, a *view* is a virtual table in which the contents are defined by a SQL query that references one or more tables. You can create a Data Catalog view that references up to 10 tables using SQL editors for Amazon Athena, Amazon Redshift, or Apache Spark using EMR Serverless or AWS Glue version 5.0. Underlying reference tables for a view can belong to the same database or different databases within the same AWS account's Data Catalog.

You can reference standard AWS Glue tables and tables in open table formats (OTF) such as Apache Hudi, Linux Foundation Delta Lake, and Apache Iceberg, with underlying data stored in Amazon S3 locations registered with AWS Lake Formation. Additionally, you can create views from federated tables from Amazon Redshift datashares that are shared with Lake Formation.

Differentiating Data Catalog views from other view types

Data Catalog views differ from Apache Hive, Apache Spark and Amazon Athena views. The Data Catalog view is a native feature of the AWS Glue Data Catalog, and is a multi-dialect definer-created view. You can create a Data Catalog view using one of the supported analytics services, such as Athena or Amazon Redshift Spectrum, and access the same view using other supported analytics services. On the other hand, the Apache Hive, Apache Spark, and Athena views are created independently in each analytics service, such as Athena and Amazon Redshift, and are visible and accessible only within that service.

What is a definer view?

A definer view is a SQL view that operates based on the permissions of the principal that created it. The definer role has the necessary permissions to access the referenced tables, and it runs the SQL statement that defines the view. The definer creates the view and shares it with other users through AWS Lake Formation's fine-grained access control.

When a user queries the definer view, the query engine uses the definer role's permissions to access the underlying reference tables. This approach enables users to interact with the view

without requiring direct access to the source tables, enhancing security and simplifying data access management.

To set up a definer view, the definer IAM role can be within the same AWS account as the base tables, or in a different account using cross-account definer roles. For more information about the permissions required for the definer role, see Prerequisites for creating views.

A framework for multi-dialect views

The Data Catalog supports creating views using multiple structured query language (SQL) dialects. SQL is a language used for storing and processing information in a relational database and each AWS analytical engine uses its own variation of SQL, or SQL dialect.

You create a Data Catalog view in one SQL dialect using one of the supported analytics query engine. Subsequently, you can update the view using the `ALTER VIEW` statement in a different SQL dialect within any other supported analytics engine. However, each dialect must reference the same set of tables, columns, and data types.

You can access the multiple dialects available for the view using the `GetTable` API, AWS CLI and AWS console. Thus, the Data Catalog view is visible and available to query across different supported analytics engines.

By defining a common view schema and metadata object that you can query from multiple engines, Data Catalog views enable you to use uniform views across your data lake.

For more details on how the schema is resolved for each dialect, see, link to the API reference. For more details on the matching rules for different types, see, link to the relevant section in the API doc.

Integrating with Lake Formation permissions

You can use AWS Lake Formation to centralize permissions management on AWS Glue Data Catalog views for users. You can grant fine-grained permissions on the Data Catalog views using the named resource method or LF-Tags, and share them across AWS accounts, AWS organizations, and organizational units. You can also share and access the Data Catalog views across AWS Regions using resource links. This allows users to provide data access without duplicating the data source, and sharing the underlying tables.

The `CREATE VIEW` DDL statement of a Data Catalog view can reference the standard AWS Glue tables and tables in open table formats (OTF) such as Hudi, Delta Lake, and Iceberg with underlying

data stored in Amazon S3 locations registered with Lake Formation as well as the federated tables from Amazon Redshift datashare that are shared with Lake Formation. The tables can be of any file format as long as the engine used to query the view supports that format. You can also reference built in functions of the engine on which it is run but other engine-specific resources may not be allowed. For more details, see Data Catalog views considerations and limitations

Use cases

Following are the important use cases for Data Catalog views:

- Create and manage permissions on a single view schema. This helps you avoid the risk of inconsistent permissions on duplicate views created in multiple engines.
- Grant permissions to users on a view that references multiple tables without granting permissions directly on the underlying reference tables.
- Achieve row level filtering on tables using LF-Tags (where LF-Tags cascade only up to column level) by applying LF-Tags on views and granting LF-Tags based permissions to users.

Supported AWS analytics services for views

The following AWS analytics services support creating Data Catalog views:

- Amazon Redshift
- Amazon Athena version 3
- Apache Spark on EMR Serverless
- Apache Spark on AWS Glue version 5.0

Additional resources

You can learn more about the Data Catalog in this guide, as well as using the following resources:

The following video demonstrates how to create views and query them from Athena and Amazon Redshift.

Topics

- Prerequisites for creating views
- Creating Data Catalog views using DDL statements

- Creating Data Catalog views using AWS Glue APIs
- Granting permissions on Data Catalog views
- Materialized views

Prerequisites for creating views

- To create views in Data Catalog, you must register the underlying Amazon S3 data locations of the reference tables with Lake Formation. For details on registering data with Lake Formation, see Adding an Amazon S3 location to your data lake.
- Only IAM roles can create Data Catalog views. Other IAM identities can't create Data Catalog views.
- The IAM role that defines the view must have the following permissions:
  - Lake Formation `SELECT` permission with the `Grantable` option on all reference tables, all columns included.
  - Lake Formation `CREATE_TABLE` permission on the target database where views are being created.
  - A trust policy for the Lake Formation and AWS Glue services to assume the role.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DataCatalogViewDefinerAssumeRole1",
            "Effect": "Allow",
            "Principal": {
               "Service": [
                    "glue.amazonaws.com",
                    "lakeformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

  - The iam:PassRole permission for AWS Glue and Lake Formation.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DataCatalogViewDefinerPassRole1",
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com",
                        "lakeformation.amazonaws.com"
                      ]
                }
            }
        }
    ]
}
```

- AWS Glue and Lake Formation permissions.

  JSON

```
{
    "Version":"2012-10-17",
                 "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "Glue:GetDatabase",
                "Glue:GetDatabases",
                "Glue:CreateTable",
                "Glue:GetTable",
                "Glue:GetTables",
                "Glue:BatchGetPartition",
                "Glue:GetPartitions",
                "Glue:GetPartition",
                "Glue:GetTableVersion",
```

```
                "Glue:GetTableVersions",
    "Glue:PassConnection",
                "lakeFormation:GetDataAccess"
            ],
            "Resource": "*"
        }
    ]
}
```

- You can't create views in a database that has `Super` or `ALL` permission granted to the `IAMAllowedPrincipals` group. You can either revoke the `Super` permission for the `IAMAllowedPrincipals` group on a database, see Step 4: Switch your data stores to the Lake Formation permissions model, or create a new database with the **Use only IAM access control for new tables in this database** box unchecked under **Default permissions for newly created tables**.

Creating Data Catalog views using DDL statements

You can create AWS Glue Data Catalog views using SQL editors for Athena, Amazon Redshift, and using the AWS Glue APIs/AWS CLI.

To create a Data Catalog view using SQL editors, choose Athena or Redshift Spectrum, and create the view using a `CREATE VIEW` Data Definition Language (DDL) statement. After creating a view in the dialect of the first engine, you can use an `ALTER VIEW` DDL statement from the second engine to add the additional dialects.

When defining views, it is important to consider the following:

- **Defining multi-dialect views** – When you define a view with multiple dialects, the schemas of the different dialects must match. Each SQL dialect will have a slightly different syntax specification. The query syntax defining the Data Catalog view should resolve to the exact same column list, including both types and names, across all the dialects. This information is stored in the `StorageDescriptor` of the view. The dialects must also reference the same underlying table objects from the Data Catalog.

  To add another dialect to a view using DDL, you can use the `ALTER VIEW` statement. If an `ALTER VIEW` statement tries to update the view definition, such as modifying the storage descriptor or underlying tables of the view, the statement errors out saying "Input and existing storage descriptor mismatch". You can use SQL cast operations to ensure that the view column types match.

- **Updating a view** – To update the view, you can use the `UpdateTable` API. If you update the view without matching storage descriptors or the reference tables, you can provide the `FORCE` flag (see engine SQL documentation for syntax). After a force update, the view will take on the forced `StorageDescriptor` and reference tables. Any further `ALTER VIEW` DDL should match the modified values. A view that has been updated to have incompatible dialects will be in a "Stale" status. The view status is visible in the Lake Formation console and using the `GetTable` operation.
- **Referencing a varchar column type as a string** – It is not possible to cast a varchar column type of Redshift Spectrum to a string. If a view is created in Redshift Spectrum with a varchar column type and a subsequent dialect tries to reference that field as a string, the Data Catalog will treat it as string without the need for the `FORCE` flag.
- **Treatment of complex type fields** – Amazon Redshift treats all complex types as SUPER types while Athena specifies the complex type. If a view has a SUPER type field, and another engine references that column as a particular complex type, such as struct (`<street_address:struct<street_number:int, street_name:string, street_type:string>>`), the Data Catalog assumes that the field to be the specific complex type, and uses that in the storage descriptor, without requiring the `Force` flag.

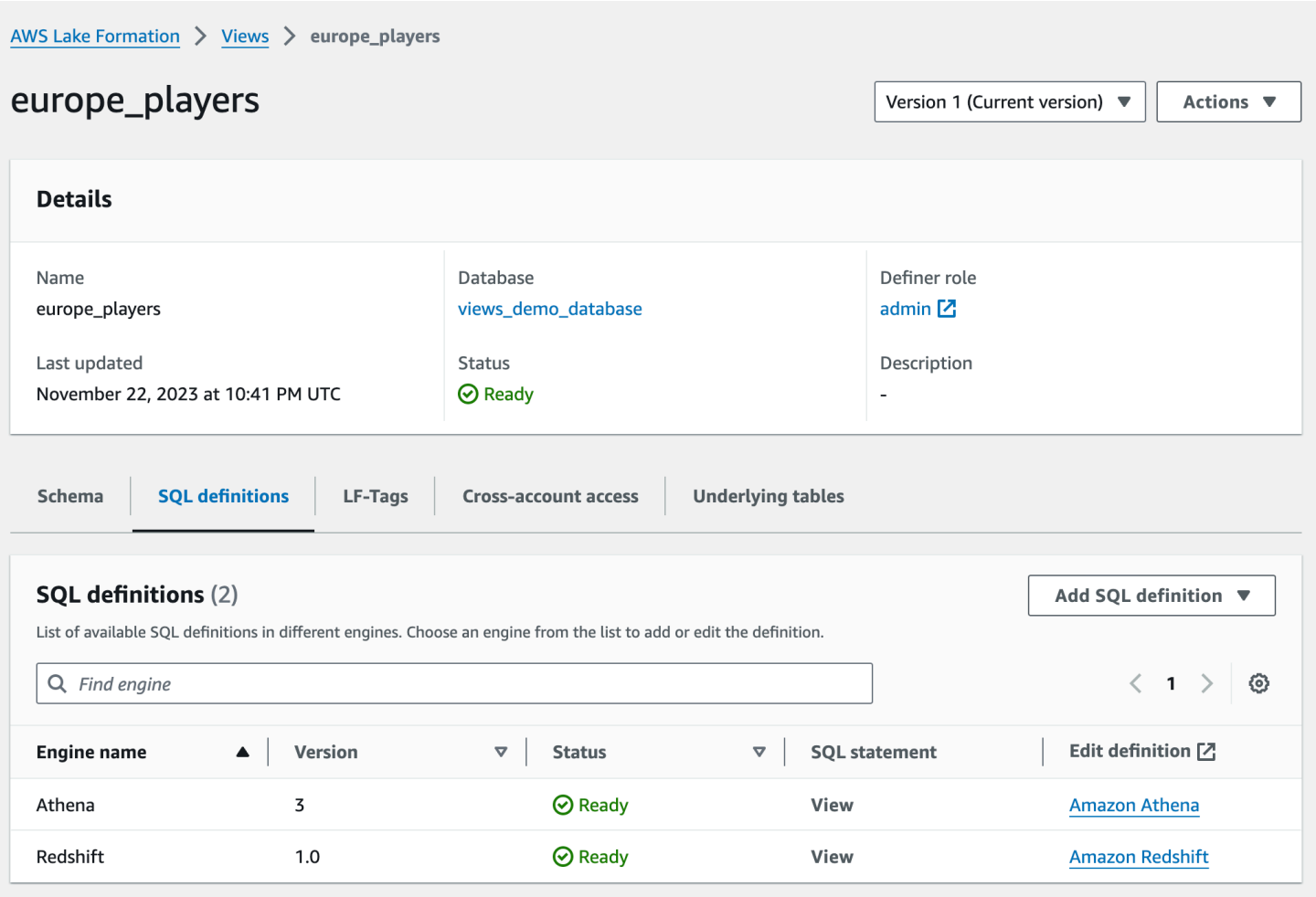
For more information about the syntax for creating and managing Data Catalog views, see:

- Using AWS Glue Data Catalog views in the Amazon Athena User Guide.
- Glue Data Catalog view query syntax in the Amazon Athena User Guide.
- Creating views in the AWS Glue Data Catalog in the Amazon Redshift Database Developer Guide.

  For more information about the SQL commands related to views in the Data Catalog, see CREATE EXTERNAL VIEW, ALTER EXTERNAL VIEW, and DROP EXTERNAL VIEW.

After you create a Data Catalog view, the details of the view is available in the Lake Formation console.

1. Choose **Views** under Data Catalog in the Lake Formation console.
2. A list of available views appears on the views page.
3. Choose a view from the list and the details page shows the attributes of the view.



Schema

Choose a `Column` row, and select **Edit LF-Tags** to update tag values or assign new LF-Tags.

SQL definitions

You can see a list of available SQL definitions. Select **Add SQL definition**, and choose a query engine to add a SQL definition. Choose a query engine (Athena or Amazon Redshift) under the `Edit definition` column to update a SQL definition.

LF-Tags

Choose **Edit LF-Tags** to edit values for a tag or assign new tags. You can use LF-Tags to grant permissions on views.

Cross-account access

You can see a list of AWS accounts, organizations and organizational units (OUs) which with you've shared the Data Catalog view.

Underlying tables

The underlying tables referenced in the SQL definition used to create the view are shown under this tab.

Creating Data Catalog views using AWS Glue APIs

You can use AWS Glue CreateTable, and UpdateTable APIs to create and update views in the Data Catalog. The `CreateTable` and `UpdateTable` operations have a new `TableInput` structure for `ViewDefinition`, while `SearchTables`, `GetTable`, `GetTables`, `GetTableVersion`, `GetTableVersions` operations provide the `ViewDefinition` in their output syntax for views. Additionally, there is a new `Status` field in the `GetTable` API output.

Two new AWS Glue connections are available for validating the SQL dialect for each supported query engine, Amazon Athena and Amazon Redshift.

The `CreateTable` and `UpdateTable` APIs are asynchronous when used with views. When these APIs are called with multiple SQL dialects, the call are validated with each engine to determine whether the dialect can be run on that engine, and if the resulting schema of the view from each dialect matches. The AWS Glue service uses these connections to make internal calls to the analytical engines. These calls simulates what the engine does to validate if a `CREATE VIEW` or `ALTER VIEW` SQL DDL were executed on the engine.

If the SQL provided is valid, and the schemas match between view dialects, the AWS Glue API atomically commits the result. Atomicity allows views with multiple dialects to be created or altered without any downtime.

Topics

- Creating AWS Glue connections to validate status
- Validating the view generation status
- Asynchronous states and operations
- View creation failure scenarios during asynchronous operations

Creating AWS Glue connections to validate status

To create or update a AWS Glue Data Catalog view using the `CreateTable` or `UpdateTable` operations, you must create a new type of AWS Glue connection for validation, and provide it to the supported analytics engine. These connections are required to use Data Catalog views with

Athena or Amazon Redshift. You can create these connections only using the AWS CLI, AWS SDKs, or AWS Glue APIs. You can't use the AWS Management Console to create the AWS Glue connection.

> **Note**
>
> If the view definer role and the role calling `CreateTable` or `UpdateTable` are different, then both of them require `glue:PassConnection` permission in their IAM policy statement.

For more information, see the create-connection AWS CLI documentation.

AWS CLI command for creating a connection

The following is an AWS CLI command for creating a connection:

```
aws glue create-connection --region us-east-1
--endpoint-url https://glue.us-east-1.amazonaws.com
--cli-input-json file:///root/path/to/create-connection.json
```

AWS CLI input JSON

For Amazon Redshift:

```
{
    "CatalogId": "123456789012",
    "ConnectionInput": {
        "ConnectionType": "VIEW_VALIDATION_REDSHIFT",
        "Name": "views-preview-cluster-connection-2",
        "Description": "My first Amazon Redshift validation connection",
        "ConnectionProperties": {
            "DATABASE": "dev",
            "CLUSTER_IDENTIFIER": "glue-data-catalog-views-preview-cluster"
        }
    }
}
```

For Amazon Athena:

```
{
    "CatalogId": "123456789012",
    "ConnectionInput": {
```

```
        "ConnectionType": "VIEW_VALIDATION_ATHENA",
        "Name": "views-preview-cluster-connection-3",
        "Description": "My first Amazon Athena validation connection",
        "ConnectionProperties": {
            "WORKGROUP_NAME": "workgroup-name"
        }
    }
}
```

Validating the view generation status

When you run the `CreateTable` or `UpdateTable` operations, the `Status` field for the `GetTable` API output shows the details of the view creation status. For `create` requests where the table does not already exist, AWS Glue creates an empty table for the duration of the asynchronous process. When calling `GetTable`, you can pass an optional boolean flag `IncludeStatusDetails`, which shows diagnostic information about the request. In the case of a failure, this flag shows an error message with individual statuses of each dialect.

Errors during view create, read, update, and delete (CRUD) operations can occur either during processing in the AWS Glue/Lake Formation service or during view SQL validation in Amazon Redshift or Athena. When an error occurs during validation in an engine, the AWS Glue service provides the error message that the engine returns.

Status fields

The following are the status fields:

- Status: a generic status, which is agnostic to different types of jobs:
  - QUEUED
  - IN_PROGRESS
  - SUCCESS
  - FAILED
- Action – Indicates which action was called on the table, currently only `CREATE` or `UPDATE` operations are available.

  Distinguishing between `UPDATE` and `CREATE` operations is important when working with views. The operation type determines how you should proceed with querying the tables.

  An `UPDATE` operation signifies that the table already exists in the Data Catalog. In this case, you can continue querying the previously created table without any issues. On the other hand, a

`CREATE` operation indicates that the table has never been successfully created before. If a table is marked for `CREATE`, attempting to query it will fail because the table does not yet exist in the system. Therefore, it is essential to identify the operation type (UPDATE or CREATE) before attempting to query a table.

- RequestedBy – The ARN of the user who requested the asynchronous change.
- UpdatedBy – The ARN of the user who last manually alter the asynchronous change process, such as requesting a cancellation or modification.
- Error – This field only appears when the state is **FAILED**. This is a parent level exception message. There may be different errors for each dialect.
  - ErrorCode – The type of exception.
  - ErrorMessage – a brief description of the exception.
- RequestTime – an ISO 8601-formatted date string indicating the time that the change was initiated.
- UpdateTime – an ISO 8601-formatted date string indicating the time that the state was last updated.

Asynchronous states and operations

When you run a `glue:CreateTable` request, the asynchronous creation of the Data Catalog view begins. In the following sections, this document describes the `Status` of a AWS Glue view that is available in a `glue:GetTable` response. For brevity, this section omits the full response.

```
{
    "Table": {
        ...
        "Status": {
            ...
            "Action": "CREATE",
            "State": "QUEUED",
        }
    }
}
```

Both of the above attributes represent important diagnostic information which indicates the state of the asynchronous operation, as well as the actions that can be performed on this view. Below are the possible values that these attributes can take on.

1. `Status.Action`
   a. CREATE
   b. UPDATE
2. `Status.State`
   a. QUEUED
   b. IN_PROGRESS
   c. SUCCESS
   d. FAILED

It is also important to note that some updates on a Data Catalog view don't require an asynchronous operation. For example, one may wish to update the `Description` attribute of the table. Since this does not require any asynchronous operations, the resulting table metadata will not have any `Status`, and the attribute will be `NULL`.

```
{
    "Table": {
        ...,
        "Description": "I changed this attribute!"
    }
}
```

Next, this topic explores how the above status information can impact operations that can be performed on an AWS Glue view.

glue:CreateTable

There are no changes for this API when compared to how `glue:CreateTable` functions for any Glue table. `CreateTable` may be called for any table name that does not already exist.

glue:UpdateTable

This operation cannot be performed on an AWS Glue view which has the following status information:

1. Action == CREATE and State == QUEUED
2. Action == CREATE and State == IN_PROGRESS
3. Action == CREATE and state == FAILED

4. Action == UPDATE and state == QUEUED
5. Action == UPDATE and state == IN_PROGRESS

To summarize, you can update a Data Catalog view only when it meets the following requirements.

1. It has been successfully created for the first time.
   a. Action == CREATE and State == SUCCESS
2. It has reached a terminal state after an asynchronous update operation.
   a. Action == UPDATE and State == SUCCESS
   b. Action == UPDATE and State == FAILED
3. It has a `NULL` state attribute as a result of a synchronous update.

glue:DeleteTable

There are no changes for this operation when compared to how `glue:DeleteTable` functions for any AWS Glue table. You can delete a Data Catalog view regardless of its state.

glue:GetTable

There are no changes for this operation when compared to how `glue:GetTable` functions for any AWS Glue table. However, you can't query a Data Catalog view from the analytical engines until it has been successfully created for the first time. `Action == CREATE and State == SUCCESS`. After you create a Data Catalog view successfully for the first time, you can query the view regardless of its status.

> **Note**
>
> All of the information in this section applies to all table read APIs such as `GetTable`, `GetTables`, and `SearchTables`.

View creation failure scenarios during asynchronous operations

The following examples are representative of the types of errors that may result from `CreateTable` or `UpdateTable` view API calls. They are not exhaustive as the error surface of SQL query failures is quite large.

Scenario 1: Amazon Redshift query failure

The query provided for Amazon Redshift includes a misspelled table name can't be found in the Data Catalog during the validation. The resulting error is shown in the `Status` field in the `GetTable` response for the view.

`GetTable` request:

```
{
    "CatalogId": "123456789012",
    "DatabaseName": "async-view-test-db",
    "TableInput": {
        "Name": "view-athena-redshift-72",
        "Description": "This is an atomic operation",
        "StorageDescriptor": {
            "Columns": [
                { "Name": "col1", "Type": "int" },
                { "Name": "col2", "Type": "string" },
                { "Name": "col3", "Type": "double" }
            ]
        },
        "ViewDefinition": {
            "Definer": "arn:aws:iam::123456789012:role/GDCViewDefiner",
            "SubObjects": [ "arn:aws:glue:us-east-1:123456789012:table/gdc-view-
playground-db/table_1" ],
            "Representations": [
                {
                    "Dialect": "ATHENA",
                    "DialectVersion": "3",
                    "ViewOriginalText": "SELECT * FROM \"gdc-view-playground-db\".
\"table_1\"",
                    "ValidationConnection": "athena-connection"
                },
                {
                    "Dialect": "REDSHIFT",
                    "DialectVersion": "1.0",
                    "ViewOriginalText": "SELECT * FROM \"gdc-view-playground-external-
schema\".\"table__1\";",
                    "ValidationConnection": "redshift-connection"
                }
            ]
        }
    }
```

```
}
```

`GetTable` response:

```
IncludeStatusDetails = FALSE
{
    "Table": {
        "Name": "view-athena-redshift-72",
        "DatabaseName": "async-view-test-db",
        "Description": "",
        "CreateTime": "2024-07-11T11:39:19-07:00",
        "UpdateTime": "2024-07-11T11:39:19-07:00",
        "Retention": 0,
        "ViewOriginalText": "",
        "ViewExpandedText": "",
        "TableType": "",
        "CreatedBy": "arn:aws:iam::123456789012:user/zcaisse",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "123456789012",
        "IsRowFilteringEnabled": false,
        "VersionId": "-1",
        "DatabaseId": "<databaseID>",
        "IsMultiDialectView": false,
        "Status": {
            "RequestedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "UpdatedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "RequestTime": "2024-07-11T11:39:19-07:00",
            "UpdateTime": "2024-07-11T11:40:06-07:00",
            "Action": "CREATE",
            "State": "FAILED"
        }
    }
}

IncludeStatusDetails = TRUE
{
    "Table": {
        "Name": "view-athena-redshift-72",
        "DatabaseName": "async-view-test-db",
        "Description": "",
        "CreateTime": "2024-07-11T11:39:19-07:00",
        "UpdateTime": "2024-07-11T11:39:19-07:00",
        "Retention": 0,
```

```
        "ViewOriginalText": "",
        "ViewExpandedText": "",
        "TableType": "",
        "CreatedBy": "arn:aws:iam::123456789012:user/zcaisse",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "123456789012",
        "IsRowFilteringEnabled": false,
        "VersionId": "-1",
        "DatabaseId": "<databaseID>",
        "IsMultiDialectView": false,
        "Status": {
            "RequestedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "UpdatedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "RequestTime": "2024-07-11T11:39:19-07:00",
            "UpdateTime": "2024-07-11T11:40:06-07:00",
            "Action": "CREATE",
            "State": "FAILED",
            "Error": {
                "ErrorCode": "QueryExecutionException",
                "ErrorMessage": "Error received during view SQL validation
using a connection: [Connection Name: redshift-connection | Query Execution
Id: ddb711d3-2415-4aa9-b251-6a76ab4f41b1 | Timestamp: Thu Jul 11 18:39:37 UTC
2024]: Redshift returned error for the statement: ERROR: AwsClientException:
EntityNotFoundException from glue - Entity Not Found"
            },
            "Details": {
                "RequestedChange": {
                    "Name": "view-athena-redshift-72",
                    "DatabaseName": "async-view-test-db",
                    "Description": "This is an atomic operation",
                    "Retention": 0,
                    "StorageDescriptor": {
                        "Columns": [
                            {
                                "Name": "col1",
                                "Type": "int"
                            },
                            {
                                "Name": "col2",
                                "Type": "string"
                            },
                            {
                                "Name": "col3",
                                "Type": "double"
```

```
                            }
                        ],
                        "Compressed": false,
                        "NumberOfBuckets": 0,
                        "SortColumns": [],
                        "StoredAsSubDirectories": false
                    },
                    "TableType": "VIRTUAL_VIEW",
                    "IsRegisteredWithLakeFormation": false,
                    "CatalogId": "123456789012",
                    "IsRowFilteringEnabled": false,
                    "VersionId": "-1",
                    "DatabaseId": "<databaseID>",
                    "ViewDefinition": {
                        "IsProtected": true,
                        "Definer": "arn:aws:iam::123456789012:role/GDCViewDefiner",
                        "SubObjects": [
                            "arn:aws:glue:us-east-1:123456789012:table/gdc-view-
playground-db/table_1"
                        ],
                        "Representations": [
                            {
                                "Dialect": "ATHENA",
                                "DialectVersion": "3",
                                "ViewOriginalText": "SELECT * FROM \"gdc-view-
playground-db\".\"table_1\"",
                                "IsStale": false
                            },
                            {
                                "Dialect": "REDSHIFT",
                                "DialectVersion": "1.0",
                                "ViewOriginalText": "SELECT * FROM \"gdc-view-
playground-external-schema\".\"table__1\";",
                                "IsStale": false
                            }
                        ]
                    },
                    "IsMultiDialectView": true
                },
                "ViewValidations": [
                    {
                        "Dialect": "ATHENA",
                        "DialectVersion": "3",
```

```
                        "ViewValidationText": "SELECT * FROM \"gdc-view-playground-db
\".\"table_1\"",
                        "UpdateTime": "2024-07-11T11:40:06-07:00",
                        "State": "SUCCESS"
                    },
                    {
                        "Dialect": "REDSHIFT",
                        "DialectVersion": "1.0",
                        "ViewValidationText": "SELECT * FROM \"gdc-view-playground-
external-schema\".\"table__1\";",
                        "UpdateTime": "2024-07-11T11:39:37-07:00",
                        "State": "FAILED",
                        "Error": {
                            "ErrorCode": "QueryExecutionException",
                            "ErrorMessage": "Error received during view SQL validation
 using a connection: [Connection Name: redshift-connection | Query Execution Id:
 ddb711d3-2415-4aa9-b251-6a76ab4f41b1 | Timestamp: Thu
 Jul 11 18:39:37 UTC 2024]: Redshift returned error for the statement: ERROR:
 AwsClientException: EntityNotFoundException from glue - Entity Not Found"
                        }
                    }
                ]
            }
        }
    }
}
```

Scenario 2: Invalid Amazon Redshift connection

The Amazon Redshift connection in the following example is malformed because it refers to a Amazon Redshift database that doesn't exist in the provided cluster/serverless endpoint. Amazon Redshift is not able to validate the view and the `Status` field in the `GetTable` response shows the error (`"State": "FAILED"` from Amazon Redshift.

`GetTable` request:

```
{
    "CatalogId": "123456789012",
    "DatabaseName": "async-view-test-db",
    "TableInput": {
        "Name": "view-athena-redshift-73",
        "Description": "This is an atomic operation",
        "StorageDescriptor": {
```

```
            "Columns": [
                { "Name": "col1", "Type": "int" },
                { "Name": "col2", "Type": "string" },
                { "Name": "col3", "Type": "double" }
            ]
        },
        "ViewDefinition": {
            "Definer": "arn:aws:iam::123456789012:role/GDCViewDdefiner",
            "SubObjects": [ "arn:aws:glue:us-east-1:123456789012:table/gdc-view-
playground-db/table_1" ],
            "Representations": [
                {
                    "Dialect": "ATHENA",
                    "DialectVersion": "3",
                    "ViewOriginalText": "SELECT * FROM \"gdc-view-playground-db\".
\"table_1\"",
                    "ValidationConnection": "athena-connection"
                },
                {
                    "Dialect": "REDSHIFT",
                    "DialectVersion": "1.0",
                    "ViewOriginalText": "SELECT * FROM \"gdc-view-playground-external-
schema\".\"table_1\";",
                    "ValidationConnection": "redshift-connection-malformed"
                }
            ]
        }
    }
}
```

`GetTable` response:

```
IncludeStatusDetails = FALSE
{
    "Table": {
        "Name": "view-athena-redshift-73",
        "DatabaseName": "async-view-test-db",
        "Description": "",
        "CreateTime": "2024-07-11T11:43:27-07:00",
        "UpdateTime": "2024-07-11T11:43:27-07:00",
        "Retention": 0,
        "ViewOriginalText": "",
        "ViewExpandedText": "",
```

```
        "TableType": "",
        "CreatedBy": "arn:aws:iam::123456789012:user/zcaisse",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "123456789012",
        "IsRowFilteringEnabled": false,
        "VersionId": "-1",
        "DatabaseId": "<databaseID>",
        "IsMultiDialectView": false,
        "Status": {
            "RequestedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "UpdatedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "RequestTime": "2024-07-11T11:43:27-07:00",
            "UpdateTime": "2024-07-11T11:43:40-07:00",
            "Action": "CREATE",
            "State": "FAILED"
        }
    }
}

IncludeStatusDetails = TRUE
{
    "Table": {
        "Name": "view-athena-redshift-73",
        "DatabaseName": "async-view-test-db",
        "Description": "",
        "CreateTime": "2024-07-11T11:43:27-07:00",
        "UpdateTime": "2024-07-11T11:43:27-07:00",
        "Retention": 0,
        "ViewOriginalText": "",
        "ViewExpandedText": "",
        "TableType": "",
        "CreatedBy": "arn:aws:iam::123456789012:user/zcaisse",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "123456789012",
        "IsRowFilteringEnabled": false,
        "VersionId": "-1",
        "DatabaseId": "<databaseID>",
        "IsMultiDialectView": false,
        "Status": {
            "RequestedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "UpdatedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "RequestTime": "2024-07-11T11:43:27-07:00",
            "UpdateTime": "2024-07-11T11:43:40-07:00",
            "Action": "CREATE",
```

```
            "State": "FAILED",
            "Error": {
                "ErrorCode": "QueryExecutionException",
                "ErrorMessage": "Error received during view SQL validation using a
connection: [Connection Name: redshift-connection-malformed | Query Execution Id:
69bfafd4-3d51-4cb0-9320-7ce5404b1809 | Timestamp: Thu Jul 11 18:43:38 UTC 2024]:
Redshift returned error for the statement: FATAL: database \"devooo\" does not exist"
            },
            "Details": {
                "RequestedChange": {
                    "Name": "view-athena-redshift-73",
                    "DatabaseName": "async-view-test-db",
                    "Description": "This is an atomic operation",
                    "Retention": 0,
                    "StorageDescriptor": {
                        "Columns": [
                            {
                                "Name": "col1",
                                "Type": "int"
                            },
                            {
                                "Name": "col2",
                                "Type": "string"
                            },
                            {
                                "Name": "col3",
                                "Type": "double"
                            }
                        ],
                        "Compressed": false,
                        "NumberOfBuckets": 0,
                        "SortColumns": [],
                        "StoredAsSubDirectories": false
                    },
                    "TableType": "VIRTUAL_VIEW",
                    "IsRegisteredWithLakeFormation": false,
                    "CatalogId": "123456789012",
                    "IsRowFilteringEnabled": false,
                    "VersionId": "-1",
                    "DatabaseId": "<databaseID>",
                    "ViewDefinition": {
                        "IsProtected": true,
                        "Definer": "arn:aws:iam::123456789012:role/GDCViewDefiner",
                        "SubObjects": [
```

```
                            "arn:aws:glue:us-east-1:123456789012:table/gdc-view-
playground-db/table_1"
                        ],
                        "Representations": [
                            {
                                "Dialect": "ATHENA",
                                "DialectVersion": "3",
                                "ViewOriginalText": "SELECT * FROM \"gdc-view-
playground-db\".\"table_1\"",
                                "IsStale": false
                            },
                            {
                                "Dialect": "REDSHIFT",
                                "DialectVersion": "1.0",
                                "ViewOriginalText": "SELECT * FROM \"gdc-view-
playground-external-schema\".\"table_1\";",
                                "IsStale": false
                            }
                        ]
                    },
                    "IsMultiDialectView": true
                },
                "ViewValidations": [
                    {
                        "Dialect": "ATHENA",
                        "DialectVersion": "3",
                        "ViewValidationText": "SELECT * FROM \"gdc-view-playground-db
\".\"table_1\"",
                        "UpdateTime": "2024-07-11T11:43:40-07:00",
                        "State": "SUCCESS"
                    },
                    {
                        "Dialect": "REDSHIFT",
                        "DialectVersion": "1.0",
                        "ViewValidationText": "SELECT * FROM \"gdc-view-playground-
external-schema\".\"table_1\";",
                        "UpdateTime": "2024-07-11T11:43:38-07:00",
                        "State": "FAILED",
                        "Error": {
                            "ErrorCode": "QueryExecutionException",
                            "ErrorMessage": "Error received during view SQL validation
 using a connection: [Connection Name: redshift-connection-malformed | Query Execution
 Id: 69bfafd4-3d51-4cb0-9320-7ce5404b1809 | Time
```

```
stamp: Thu Jul 11 18:43:38 UTC 2024]: Redshift returned error for the statement: FATAL:
 database \"devooo\" does not exist"
                        }
                    }
                ]
            }
        }
    }
}
```

Scenario 3: Athena query failure

The SQL for Athena here is invalid because the query misspells the database name. The Athena query validation catches this and the resulting error is surfaced through the `Status` object in a `GetTable` call.

`GetTable` request:

```
{
    "CatalogId": "123456789012",
    "DatabaseName": "async-view-test-db",
    "TableInput": {
        "Name": "view-athena-redshift-70",
        "Description": "This is an atomic operation",
        "StorageDescriptor": {
            "Columns": [
                { "Name": "col1", "Type": "int" },
                { "Name": "col2", "Type": "string" },
                { "Name": "col3", "Type": "double" }
            ]
        },
        "ViewDefinition": {
            "Definer": "arn:aws:iam::123456789012:role/GDCViewDefiner",
            "SubObjects": [ "arn:aws:glue:us-east-1:123456789012:table/gdc-view-
playground-db/table_1" ],
            "Representations": [
                {
                    "Dialect": "ATHENA",
                    "DialectVersion": "3",
                    "ViewOriginalText": "SELECT * FROM \"gdc--view-playground-db\".
\"table_1\"",
```

```
                    "ValidationConnection": "athena-connection"
                },
                {
                    "Dialect": "REDSHIFT",
                    "DialectVersion": "1.0",
                    "ViewOriginalText": "SELECT * FROM \"gdc-view-playground-external-
schema\".\"table_1\";",
                    "ValidationConnection": "redshift-connection"
                }
            ]
        }
    }
}
```

GetTable response:

```
IncludeStatusDetails = FALSE
{
    "Table": {
        "Name": "view-athena-redshift-70",
        "DatabaseName": "async-view-test-db",
        "Description": "",
        "CreateTime": "2024-07-11T11:09:53-07:00",
        "UpdateTime": "2024-07-11T11:09:53-07:00",
        "Retention": 0,
        "ViewOriginalText": "",
        "ViewExpandedText": "",
        "TableType": "",
        "CreatedBy": "arn:aws:iam::123456789012:user/",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "123456789012",
        "IsRowFilteringEnabled": false,
        "VersionId": "-1",
        "DatabaseId": "<databaseID>",
        "IsMultiDialectView": false,
        "Status": {
            "RequestedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "UpdatedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "RequestTime": "2024-07-11T11:09:54-07:00",
            "UpdateTime": "2024-07-11T11:10:41-07:00",
            "Action": "CREATE",
            "State": "FAILED",
        }
```

```
    }
}

IncludeStatusDetails = TRUE
{
    "Table": {
        "Name": "view-athena-redshift-70",
        "DatabaseName": "async-view-test-db",
        "Description": "",
        "CreateTime": "2024-07-11T11:09:53-07:00",
        "UpdateTime": "2024-07-11T11:09:53-07:00",
        "Retention": 0,
        "ViewOriginalText": "",
        "ViewExpandedText": "",
        "TableType": "",
        "CreatedBy": "arn:aws:iam::123456789012:user/zcaisse",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "123456789012",
        "IsRowFilteringEnabled": false,
        "VersionId": "-1",
        "DatabaseId": "<databaseID>",
        "IsMultiDialectView": false,
        "Status": {
            "RequestedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "UpdatedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "RequestTime": "2024-07-11T11:09:54-07:00",
            "UpdateTime": "2024-07-11T11:10:41-07:00",
            "Action": "CREATE",
            "State": "FAILED",
            "Error": {
                "ErrorCode": "QueryExecutionException",
                "ErrorMessage": "Error received during view SQL validation using
 a connection: [Connection Name: athena-connection | Query Execution Id: d9bb1e6d-
ce26-4b35-8276-8a199af966aa | Timestamp: Thu Jul 11 18:10:
41 UTC 2024]: Athena validation FAILED: {ErrorCategory: 2,ErrorType: 1301,Retryable:
 false,ErrorMessage: line 1:118: Schema 'gdc--view-playground-db' does not exist}"
            },
            "Details": {
                "RequestedChange": {
                    "Name": "view-athena-redshift-70",
                    "DatabaseName": "async-view-test-db",
                    "Description": "This is an atomic operation",
                    "Retention": 0,
                    "StorageDescriptor": {
```

```
                        "Columns": [
                            {
                                "Name": "col1",
                                "Type": "int"
                            },
                            {
                                "Name": "col2",
                                "Type": "string"
                            },
                            {
                                "Name": "col3",
                                "Type": "double"
                            }
                        ],
                        "Compressed": false,
                        "NumberOfBuckets": 0,
                        "SortColumns": [],
                        "StoredAsSubDirectories": false
                    },
                    "TableType": "VIRTUAL_VIEW",
                    "IsRegisteredWithLakeFormation": false,
                    "CatalogId": "123456789012",
                    "IsRowFilteringEnabled": false,
                    "VersionId": "-1",
                    "DatabaseId": "<databaseID>",
                    "ViewDefinition": {
                        "IsProtected": true,
                        "Definer": "arn:aws:iam::123456789012:role/GDCViewDefiner",
                        "SubObjects": [
                            "arn:aws:glue:us-east-1:123456789012:table/gdc-view-
playground-db/table_1"
                        ],
                        "Representations": [
                            {
                                "Dialect": "ATHENA",
                                "DialectVersion": "3",
                                "ViewOriginalText": "SELECT * FROM \"gdc--view-
playground-db\".\"table_1\"",
                                "IsStale": false
                            },
                            {
                                "Dialect": "REDSHIFT",
                                "DialectVersion": "1.0",
```

```
                                "ViewOriginalText": "SELECT * FROM \"gdc-view-
playground-external-schema\".\"table_1\";",
                                "IsStale": false
                            }
                        ]
                    },
                    "IsMultiDialectView": true
                },
                "ViewValidations": [
                    {
                        "Dialect": "ATHENA",
                        "DialectVersion": "3",
                        "ViewValidationText": "SELECT * FROM \"gdc--view-playground-db
\".\"table_1\"",
                        "UpdateTime": "2024-07-11T11:10:41-07:00",
                        "State": "FAILED",
                        "Error": {
                            "ErrorCode": "QueryExecutionException",
                            "ErrorMessage": "Error received during view SQL validation
 using a connection: [Connection Name: athena-connection | Query Execution Id:
 d9bb1e6d-ce26-4b35-8276-8a199af966aa | Timestamp: Thu J
ul 11 18:10:41 UTC 2024]: Athena validation FAILED: {ErrorCategory: 2,ErrorType:
 1301,Retryable: false,ErrorMessage: line 1:118: Schema 'gdc--view-playground-db' does
 not exist}"
                        }
                    },
                    {
                        "Dialect": "REDSHIFT",
                        "DialectVersion": "1.0",
                        "ViewValidationText": "SELECT * FROM \"gdc-view-playground-
external-schema\".\"table_1\";",
                        "UpdateTime": "2024-07-11T11:10:41-07:00",
                        "State": "SUCCESS"
                    }
                ]
            }
        }
    }
}
```

Scenario 4: Mismatch storage descriptors

The SQL provided for the Athena dialect selects `col1` and `col2` while the SQL for Redshift selects only `col1`. This leads to a storage descriptor mismatch error.

`GetTable` request:

```
{
    "CatalogId": "123456789012",
    "DatabaseName": "async-view-test-db",
    "TableInput": {
        "Name": "view-athena-redshift-71",
        "Description": "This is an atomic operation",
        "StorageDescriptor": {
            "Columns": [
                { "Name": "col1", "Type": "int" },
                { "Name": "col2", "Type": "string" },
                { "Name": "col3", "Type": "double" }
            ]
        },
        "ViewDefinition": {
            "Definer": "arn:aws:iam::123456789012:role/GDCViewDdefiner",
            "SubObjects": [ "arn:aws:glue:us-east-1:123456789012:table/gdc-view-
playground-db/table_1" ],
            "Representations": [
                {
                    "Dialect": "ATHENA",
                    "DialectVersion": "3",
                    "ViewOriginalText": "SELECT col1, col2 FROM \"gdc-view-playground-
db\".\"table_1\"",
                    "ValidationConnection": "athena-connection"
                },
                {
                    "Dialect": "REDSHIFT",
                    "DialectVersion": "1.0",
                    "ViewOriginalText": "SELECT col1 FROM \"gdc-view-playground-
external-schema\".\"table_1\";",
                    "ValidationConnection": "redshift-connection"
                }
            ]
        }
    }
}
```

`GetTable` response:

```
IncludeStatusDetails = FALSE

{
    "Table": {
        "Name": "view-athena-redshift-71",
        "DatabaseName": "async-view-test-db",
        "Description": "",
        "CreateTime": "2024-07-11T11:22:02-07:00",
        "UpdateTime": "2024-07-11T11:22:02-07:00",
        "Retention": 0,
        "ViewOriginalText": "",
        "ViewExpandedText": "",
        "TableType": "",
        "CreatedBy": "arn:aws:iam::123456789012:user/zcaisse",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "123456789012",
        "IsRowFilteringEnabled": false,
        "VersionId": "-1",
        "DatabaseId": "<databaseID>",
        "IsMultiDialectView": false,
        "Status": {
            "RequestedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "UpdatedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "RequestTime": "2024-07-11T11:22:02-07:00",
            "UpdateTime": "2024-07-11T11:23:19-07:00",
            "Action": "CREATE",
            "State": "FAILED"
        }
    }
}

IncludeStatusDetails = TRUE

{
    "Table": {
        "Name": "view-athena-redshift-71",
        "DatabaseName": "async-view-test-db",
        "Description": "",
        "CreateTime": "2024-07-11T11:22:02-07:00",
        "UpdateTime": "2024-07-11T11:22:02-07:00",
        "Retention": 0,
        "ViewOriginalText": "",
```

```
        "ViewExpandedText": "",
        "TableType": "",
        "CreatedBy": "arn:aws:iam::123456789012:user/zcaisse",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "123456789012",
        "IsRowFilteringEnabled": false,
        "VersionId": "-1",
        "DatabaseId": "<databaseID>",
        "IsMultiDialectView": false,
        "Status": {
            "RequestedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "UpdatedBy": "arn:aws:iam::123456789012:user/zcaisse",
            "RequestTime": "2024-07-11T11:22:02-07:00",
            "UpdateTime": "2024-07-11T11:23:19-07:00",
            "Action": "CREATE",
            "State": "FAILED",
            "Error": {
                "ErrorCode": "InvalidInputException",
                "ErrorMessage": "Engine and existing storage descriptor mismatch"
            },
            "Details": {
                "RequestedChange": {
                    "Name": "view-athena-redshift-71",
                    "DatabaseName": "async-view-test-db",
                    "Description": "This is an atomic operation",
                    "Retention": 0,
                    "StorageDescriptor": {
                        "Columns": [
                            {
                                "Name": "col1",
                                "Type": "int"
                            },
                            {
                                "Name": "col2",
                                "Type": "string"
                            },
                            {
                                "Name": "col3",
                                "Type": "double"
                            }
                        ],
                        "Compressed": false,
                        "NumberOfBuckets": 0,
                        "SortColumns": [],
```

```
                        "StoredAsSubDirectories": false
                    },
                    "TableType": "VIRTUAL_VIEW",
                    "IsRegisteredWithLakeFormation": false,
                    "CatalogId": "123456789012",
                    "IsRowFilteringEnabled": false,
                    "VersionId": "-1",
                    "DatabaseId": "<databaseID>",
                    "ViewDefinition": {
                        "IsProtected": true,
                        "Definer": "arn:aws:iam::123456789012:role/GDCViewDefiner",
                        "SubObjects": [
                            "arn:aws:glue:us-east-1:123456789012:table/gdc-view-
playground-db/table_1"
                        ],
                        "Representations": [
                            {
                                "Dialect": "ATHENA",
                                "DialectVersion": "3",
                                "ViewOriginalText": "SELECT col1, col2 FROM \"gdc-view-
playground-db\".\"table_1\"",
                                "IsStale": false
                            },
                            {
                                "Dialect": "REDSHIFT",
                                "DialectVersion": "1.0",
                                "ViewOriginalText": "SELECT col1 FROM \"gdc-view-
playground-external-schema\".\"table_1\";",
                                "IsStale": false
                            }
                        ]
                    },
                    "IsMultiDialectView": true
                },
                "ViewValidations": [
                    {
                        "Dialect": "ATHENA",
                        "DialectVersion": "3",
                        "ViewValidationText": "SELECT col1, col2 FROM \"gdc-view-
playground-db\".\"table_1\"",
                        "UpdateTime": "2024-07-11T11:23:19-07:00",
                        "State": "FAILED",
                        "Error": {
                            "ErrorCode": "InvalidInputException",
```

```
                            "ErrorMessage": "Engine and existing storage descriptor
 mismatch"
                        }
                    },
                    {
                        "Dialect": "REDSHIFT",
                        "DialectVersion": "1.0",
                        "ViewValidationText": "SELECT col1 FROM \"gdc-view-playground-
external-schema\".\"table_1\";",
                        "UpdateTime": "2024-07-11T11:22:49-07:00",
                        "State": "FAILED",
                        "Error": {
                            "ErrorCode": "InvalidInputException",
                            "ErrorMessage": "Engine and existing storage descriptor
 mismatch"
                        }
                    }
                ]
            }
        }
    }
}
```

Granting permissions on Data Catalog views

After creating views in the AWS Glue Data Catalog, you can grant data lake permissions on views to principals across AWS accounts, organizations and organizational units. You can grant permissions using LF-Tags or the named resource method. For more information on tagging resources, see Lake Formation tag-based access control. For more information on granting permissions on views directly, see Granting permissions on views using the named resource method.

Materialized views

Topics

- Differentiating materialized views from other view types
- Use cases
- Key concepts
- Permissions for materialized views
- Creating and managing materialized views
- Storage and data access

- Integrating with AWS Lake Formation permissions
- Monitoring and debugging
- Managing refresh jobs
- Monitoring and troubleshooting
- Considerations and limitations

In the AWS Glue Data Catalog, a materialized view is a managed table that stores the precomputed result of a SQL query in Apache Iceberg format. Unlike standard Data Catalog views that execute the query each time they are accessed, materialized views physically store the query results and update them as the underlying source tables change. You can create materialized views using Apache Spark version 3.5.6+ in Amazon Athena, Amazon EMR, or AWS Glue.

Materialized views reference Apache Iceberg tables registered in the AWS Glue Data Catalog, with precomputed data stored as Apache Iceberg tables in Amazon S3 Tables buckets or Amazon S3 general purpose buckets, making them accessible from multiple query engines including Amazon Athena, Amazon Redshift, and third-party Iceberg-compatible engines.

Differentiating materialized views from other view types

Materialized views differ from AWS Glue Data Catalog views, Apache Spark views, and Amazon Athena views in fundamental ways. While Data Catalog views are virtual tables that execute the SQL query definition each time they are accessed, materialized views physically store precomputed query results. This eliminates redundant computation and significantly improves query performance for frequently accessed complex transformations.

Materialized views also differ from traditional data transformation pipelines built with AWS Glue ETL or custom Spark jobs. Instead of writing custom code to handle change detection, incremental updates, and workflow orchestration, you define materialized views using standard SQL syntax. The AWS Glue Data Catalog automatically monitors source tables, detects changes, and refreshes materialized views using fully managed compute infrastructure.

Use cases

Following are important use cases for materialized views:

- **Accelerate complex analytical queries** – Create materialized views that precompute expensive joins, aggregations, and window functions. Spark engines automatically rewrite subsequent queries to use the precomputed results, reducing query latency and compute costs.

- **Simplify data transformation pipelines** – Replace complex ETL jobs that handle change detection, incremental updates, and workflow orchestration with simple SQL-based materialized view definitions. The AWS Glue Data Catalog manages all operational complexity automatically.
- **Enable self-service analytics with governed data access** – Create curated materialized views that transform raw data into business-ready datasets. Grant users access to materialized views without exposing underlying source tables, simplifying security management while empowering self-service analytics.
- **Optimize feature engineering for machine learning** – Define materialized views that implement feature transformations for ML models. The automatic refresh capability ensures feature stores remain current as source data evolves, while incremental refresh minimizes compute costs.
- **Implement efficient data sharing** – Create materialized views that filter and transform data for specific consumers. Share materialized views across accounts and regions using AWS Lake Formation, eliminating the need for data duplication while maintaining centralized governance.

Key concepts

Automatic refresh

Automatic refresh is a capability that continuously monitors your source tables and updates materialized views according to a schedule you define. When you create a materialized view, you can specify a refresh frequency using time-based scheduling with intervals as frequent as one hour. The AWS Glue Data Catalog uses managed Spark compute infrastructure to execute refresh operations in the background, transparently handling all aspects of change detection and incremental updates.

When source data changes between refresh intervals, the materialized view becomes temporarily stale. Queries directly accessing the materialized view may return outdated results until the next scheduled refresh completes. For scenarios requiring immediate access to the most current data, you can execute a manual refresh using the `REFRESH MATERIALIZED VIEW` SQL command.

Incremental refresh

Incremental refresh is an optimization technique that processes only the data that has changed in source tables since the last refresh, rather than recomputing the entire materialized view. The AWS Glue Data Catalog leverages Apache Iceberg's metadata layer to efficiently track changes in source tables and determine which portions of the materialized view require updates.

This approach significantly reduces compute costs and refresh duration compared to full refresh operations, particularly for large datasets where only a small percentage of data changes between refresh cycles. The incremental refresh mechanism operates automatically; you don't need to write custom logic to detect or process changed data.

Automatic query rewrite

Automatic query rewrite is a query optimization capability available in Spark engines across Amazon Athena, Amazon EMR, and AWS Glue. When you execute a query against base tables, the Spark optimizer analyzes your query plan and automatically determines whether available materialized views can satisfy the query more efficiently. If a suitable materialized view exists, the optimizer transparently rewrites the query to use the precomputed results instead of processing the base tables.

This optimization occurs without requiring any changes to your application code or query statements. The Spark optimizer ensures that automatic query rewrite only applies when the materialized view is current and can produce accurate results. If a materialized view is stale or doesn't fully match the query requirements, the optimizer executes the original query plan against base tables, prioritizing correctness over performance.

View definer role

A materialized view operates based on the permissions of the IAM role that created it, known as the view definer role. The definer role must have read access to all base tables referenced in the materialized view definition and create table permissions on the target database. When the AWS Glue Data Catalog refreshes a materialized view, it assumes the definer role to access source tables and write updated results.

This security model enables you to grant users access to materialized views without granting them direct permissions on the underlying source tables. If the view definer role loses access to any base table, subsequent refresh operations will fail until permissions are restored.

Permissions for materialized views

To create and manage materialized views, you must configure AWS Lake Formation permissions. The IAM role creating the materialized view (the definer role) requires specific permissions on source tables and target databases.

Required permissions for the definer role

The definer role must have the following Lake Formation permissions:

- On source tables – SELECT or ALL permissions without row, column, or cell filters
- On the target database – CREATE_TABLE permission
- On the AWS Glue Data Catalog – GetTable and CreateTable API permissions

When you create a materialized view, the definer role's ARN is stored in the view definition. The AWS Glue Data Catalog assumes this role when executing automatic refresh operations. If the definer role loses access to source tables, refresh operations will fail until permissions are restored.

IAM permissions for AWS Glue jobs

Your AWS Glue job's IAM role requires the following permissions:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetCatalog",
                "glue:GetCatalogs",
                "glue:GetTable",
                "glue:GetTables",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*:/aws-glue/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": "*"
        }
    ]
}
```

The role you use for Materialized View auto-refresh must have the iam:PassRole permission on the role.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
```

```
        "arn:aws:iam::111122223333:role/materialized-view-role-name"
      ]
    }
  ]
}
```

To let Glue automatically refresh the materialized view for you, the role must also have the following trust policy that enables the service to assume the role.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "glue.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

If the Materialized View is stored in S3 Tables Buckets, you also need to add the following permission to the role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3tables:PutTableMaintenanceConfiguration"
      ],
      "Resource": "arn:aws:s3tables:*:123456789012:*"
    }
  ]
}
```

Granting access to materialized views

To grant other users access to query a materialized view, use AWS Lake Formation to grant SELECT permission on the materialized view table. Users can query the materialized view without requiring direct access to the underlying source tables.

For detailed information about configuring Lake Formation permissions, see Granting and revoking permissions on Data Catalog resources in the AWS Lake Formation Developer Guide.

Creating and managing materialized views

You create materialized views using the `CREATE MATERIALIZED VIEW` SQL statement in Spark engines. The view definition specifies the SQL query that defines the transformation logic, the target database and table name, and optional refresh configuration. You can define complex transformations including aggregations, joins across multiple tables, filters, and window functions.

```
CREATE MATERIALIZED VIEW sales_summary
AS
SELECT
    region,
    product_category,
    SUM(sales_amount) as total_sales,
    COUNT(DISTINCT customer_id) as unique_customers
FROM sales_transactions
WHERE transaction_date >= current_date - interval '90' day
GROUP BY region, product_category;
```

To configure automatic refresh, include the refresh schedule in your view definition:

```
CREATE MATERIALIZED VIEW sales_summary
SCHEDULE REFRESH EVERY 1 HOUR
AS
SELECT region, product_category, SUM(sales_amount) as total_sales
FROM sales_transactions
GROUP BY region, product_category;
```

You can manually refresh a materialized view at any time using the `REFRESH MATERIALIZED VIEW` command:

```
REFRESH MATERIALIZED VIEW sales_summary;
```

To modify an existing materialized view's refresh schedule, use the `ALTER MATERIALIZED VIEW` statement:

```
ALTER MATERIALIZED VIEW sales_summary
ADD SCHEDULE REFRESH EVERY 2 HOURS;
```

Nested materialized views

You can create materialized views that reference other materialized views as base tables, enabling multi-stage data transformations. When you create nested materialized views, the AWS Glue Data Catalog tracks dependencies and automatically propagates updates through the materialized view hierarchy. When a base materialized view refreshes, all downstream materialized views that depend on it are updated accordingly.

This capability allows you to decompose complex transformations into logical stages, improving maintainability and enabling selective refresh of transformation layers based on your data freshness requirements.

Storage and data access

Materialized views store precomputed results as Apache Iceberg tables in S3 Tables buckets or general purpose S3 buckets within your AWS account. The AWS Glue Data Catalog manages all aspects of Iceberg table maintenance, including compaction and snapshot retention, through S3 Tables' automated optimization capabilities.

Because materialized views are stored as Iceberg tables, you can read them directly from any Iceberg-compatible engine, including Amazon Athena, Amazon Redshift, and third-party analytics platforms. This multi-engine accessibility ensures your precomputed data remains accessible across your entire analytics ecosystem without data duplication or format conversion.

Integrating with AWS Lake Formation permissions

You can use AWS Lake Formation to manage fine-grained permissions on materialized views. The view creator automatically becomes the owner of the materialized view and can grant other users or roles permissions using AWS Lake Formation's named resource method or LF-Tags.

When you grant a user `SELECT` permission on a materialized view, they can query the precomputed results without requiring access to the underlying source tables. This security model simplifies data

access management and enables you to implement the principle of least privilege, providing users with access to only the specific data transformations they need.

You can share materialized views across AWS accounts, AWS organizations, and organizational units using AWS Lake Formation's cross-account sharing capabilities. You can also access materialized views across AWS Regions using resource links, enabling centralized data governance with distributed data access.

Monitoring and debugging

The AWS Glue Data Catalog publishes all materialized view refresh operations and associated metrics to Amazon CloudWatch. You can monitor refresh start time, end time, duration, data volume processed, and refresh status through CloudWatch metrics. When refresh operations fail, error messages and diagnostic information are captured in CloudWatch Logs.

You can set up CloudWatch alarms to receive notifications when refresh jobs exceed expected duration or fail repeatedly. The AWS Glue Data Catalog also publishes change events to for both successful and failed refresh runs, enabling you to integrate materialized view operations into broader workflow automation.

To check the current status of a materialized view, use the `DESCRIBE MATERIALIZED VIEW` SQL command, which returns metadata including staleness status, last refresh timestamp, and refresh schedule configuration.

Managing refresh jobs

Starting a manual refresh

Trigger an immediate refresh outside the scheduled interval.

Required Permission: The AWS credentials used to make the API call must have `glue:GetTable` permission for the materialized view.

For S3 Tables Catalog:

```
aws glue start-materialized-view-refresh-task-run \
    --catalog-id <ACCOUNT_ID>:s3tablescatalog/<CATALOG_NAME> \
    --database-name <DATABASE_NAME> \
    --table-name <MV_TABLE_NAME>
```

For Root Catalog:

```
aws glue start-materialized-view-refresh-task-run \
    --catalog-id <ACCOUNT_ID> \
    --database-name <DATABASE_NAME> \
    --table-name <MV_TABLE_NAME>
```

Checking refresh status

Get the status of a specific refresh job:

```
aws glue get-materialized-view-refresh-task-run \
    --catalog-id <CATALOG_ID> \
    --materialized-view-refresh-task-run-id <TASK_RUN_ID>
```

Listing refresh history

View all refresh jobs for a materialized view:

```
aws glue list-materialized-view-refresh-task-runs \
    --catalog-id <CATALOG_ID> \
    --database-name <DATABASE_NAME> \
    --table-name <MV_TABLE_NAME>
```

> **Note**
>
> Use <ACCOUNT_ID>:s3tablescatalog/<CATALOG_NAME> for S3 Tables or <ACCOUNT_ID> for root catalog.

Stopping a running refresh

Cancel an in-progress refresh job:

```
aws glue stop-materialized-view-refresh-task-run \
    --catalog-id <CATALOG_ID> \
    --database-name <DATABASE_NAME> \
    --table-name <MV_TABLE_NAME>
```

Monitoring and troubleshooting

There are three ways to monitor materialized view refresh jobs:

CloudWatch Metrics

View aggregated metrics for all your materialized view refresh jobs in CloudWatch:

Available Metrics:

- AWS/Glue namespace with dimensions:
  - CatalogId: Your catalog identifier
  - DatabaseName: Database containing the materialized view
  - TableName: Materialized view name
  - TaskType: Set to "MaterializedViewRefresh"

Viewing in Console:

1. Navigate to CloudWatch Console → Metrics
2. Select AWS/Glue namespace
3. Filter by dimensions: CatalogId, DatabaseName, TableName, TaskType
4. View metrics for job success, failure, and duration

Example CloudWatch Metrics Query:

```
{AWS/Glue,CatalogId,DatabaseName,TableName,TaskType} MaterializedViewRefresh
```

Using AWS CLI:

```
aws cloudwatch get-metric-statistics \
    --namespace AWS/Glue \
    --metric-name <MetricName> \
    --dimensions Name=CatalogId,Value=<CATALOG_ID> \
                 Name=DatabaseName,Value=<DATABASE_NAME> \
                 Name=TableName,Value=<TABLE_NAME> \
                 Name=TaskType,Value=MaterializedViewRefresh \
    --start-time <START_TIME> \
    --end-time <END_TIME> \
    --period 3600 \
    --statistics Sum \
    --region <REGION>
```

CloudWatch Logs

View detailed execution logs for individual refresh task runs:

Log Group: `/aws-glue/materialized-views/<task_run_id>`

Where `<task_run_id>` is a UUID (e.g., abc12345-def6-7890-ghij-klmnopqrstuv).

Viewing Logs:

```
# List log streams for a task run
aws logs describe-log-streams \
    --log-group-name /aws-glue/materialized-views/<TASK_RUN_ID> \
    --region <REGION>

# Get log events
aws logs get-log-events \
    --log-group-name /aws-glue/materialized-views/<TASK_RUN_ID> \
    --log-stream-name <LOG_STREAM_NAME> \
    --region <REGION>
```

In CloudWatch Console:

1. Navigate to CloudWatch → Log groups
2. Search for /aws-glue/materialized-views/
3. Select the log group with your task run ID
4. View detailed execution logs, errors, and Spark job output

Notifications

Subscribe to events for real-time notifications about refresh job state changes:

Available Event Types:

- Glue Materialized View Refresh Task Started
- Glue Materialized View Refresh Task Succeeded
- Glue Materialized View Refresh Task Failed
- Glue Materialized View Auto-Refresh Invocation Failure

Creating an Rule:

```
aws events put-rule \
    --name materialized-view-refresh-notifications \
    --event-pattern '{
        "source": ["aws.glue"],
        "detail-type": [
            "Glue Materialized View Refresh Task Started",
            "Glue Materialized View Refresh Task Succeeded",
            "Glue Materialized View Refresh Task Failed",
            "Glue Materialized View Auto-Refresh Invocation Failure"
        ]
    }' \
    --region <REGION>
```

Adding a Target (e.g., SNS Topic):

```
aws events put-targets \
    --rule materialized-view-refresh-notifications \
    --targets "Id"="1","Arn"="arn:aws:sns:<REGION>:<ACCOUNT_ID>:<TOPIC_NAME>" \
    --region <REGION>
```

Viewing refresh status

Check the status of your materialized view refresh jobs using the AWS Glue API:

```
aws glue get-materialized-view-refresh-task-run \
    --catalog-id <CATALOG_ID> \
    --materialized-view-refresh-task-run-id <TASK_RUN_ID> \
    --region <REGION>
```

Or list all recent refresh runs:

```
aws glue list-materialized-view-refresh-task-runs \
    --catalog-id <CATALOG_ID> \
    --database-name <DATABASE_NAME> \
    --table-name <MV_TABLE_NAME> \
    --region <REGION>
```

This shows:

- Last refresh time

- Refresh status (SUCCEEDED, FAILED, RUNNING, STOPPED)
- Task run ID
- Error messages (if failed)

Common Refresh States:

- RUNNING: Refresh job is currently executing
- SUCCEEDED: Refresh completed successfully
- FAILED: Refresh encountered an error
- STOPPED: Refresh was manually cancelled

Troubleshooting Failed Refreshes:

If a refresh fails, check:

1. IAM Permissions: Ensure the definer role has access to all base tables and the materialized view location
2. Base Table Availability: Verify all referenced tables exist and are accessible
3. Query Validity: Confirm the SQL query is valid for Spark SQL dialect
4. Resource Limits: Check if you've reached concurrent refresh limits for your account

Use the GetMaterializedViewRefreshTaskRun API to retrieve detailed error messages.

Considerations and limitations

- Materialized views can only reference Apache Iceberg tables registered in the AWS Glue Data Catalog as base tables.
- View creation and automatic query rewrite are available only from Spark engines in Apache Spark version 3.5.6 and above across Amazon Athena, Amazon EMR, and AWS Glue (Version 5.1).
- Materialized views are eventually consistent with base tables. During the refresh window, queries directly accessing the materialized view may return outdated data. For immediate access to current data, execute a manual refresh.
- The minimum automatic refresh interval is one hour. For use cases requiring more frequent updates, execute manual refreshes programmatically using the `REFRESH MATERIALIZED VIEW` command.

- Query rewrite prioritizes correctness over performance. If a materialized view is stale or cannot satisfy query requirements accurately, Spark engines execute the original query against base tables.

Importing data using workflows in Lake Formation

With AWS Lake Formation, you can import your data using *workflows*. A workflow defines the data source and schedule to import data into your data lake. It is a container for AWS Glue crawlers, jobs, and triggers that are used to orchestrate the processes to load and update the data lake.

Topics

- Blueprints and workflows in Lake Formation
- Creating a workflow
- Running a workflow

Blueprints and workflows in Lake Formation

A workflow encapsulates a complex multi-job extract, transform, and load (ETL) activity. Workflows generate AWS Glue crawlers, jobs, and triggers to orchestrate the loading and update of data. Lake Formation executes and tracks a workflow as a single entity. You can configure a workflow to run on demand or on a schedule.

> **Note**
>
> Spark parquet writer doesn't support special characters in column names. This is a technical limitation of the writer itself, not a configuration issue.

Workflows that you create in Lake Formation are visible in the AWS Glue console as a directed acyclic graph (DAG). Each DAG node is a job, crawler, or trigger. To monitor progress and troubleshoot, you can track the status of each node in the workflow.

When a Lake Formation workflow has completed, the user who ran the workflow is granted the Lake Formation SELECT permission on the Data Catalog tables that the workflow creates.

You can also create workflows in AWS Glue. However, because Lake Formation enables you to create a workflow from a blueprint, creating workflows is much simpler and more automated in Lake Formation. Lake Formation provides the following types of blueprints:

- **Database snapshot** – Loads or reloads data from all tables into the data lake from a JDBC source. You can exclude some data from the source based on an exclude pattern.
- **Incremental database** – Loads only new data into the data lake from a JDBC source, based on previously set bookmarks. You specify the individual tables in the JDBC source database to include. For each table, you choose the bookmark columns and bookmark sort order to keep track of data that has previously been loaded. The first time that you run an incremental database blueprint against a set of tables, the workflow loads all data from the tables and sets bookmarks for the next incremental database blueprint run. You can therefore use an incremental database blueprint instead of the database snapshot blueprint to load all data, provided that you specify each table in the data source as a parameter.
- **Log file** – bulk loads data from log file sources, including AWS CloudTrail, Elastic Load Balancing logs, and Application Load Balancer logs.

Use the following table to help decide whether to use a database snapshot or incremental database blueprint.

Use database snapshot when...	Use incremental database when...
• Schema evolution is flexible. (Columns are re-named, previous columns are deleted, and new columns are added in their place.) • Complete consistency is needed between the source and the destination.	• Schema evolution is incremental. (There is only successive addition of columns.) • Only new rows are added; previous rows are not updated.

> **Note**
>
> Users cannot edit blue prints and workflows created by Lake Formation.

Creating a workflow

Before you start, ensure that you have granted the required data permissions and data location permissions to the role `LakeFormationWorkflowRole`. This is so the workflow can create metadata tables in the Data Catalog and write data to target locations in Amazon S3. For more information, see (Optional) Create an IAM role for workflows and Overview of Lake Formation permissions.

> **Note**
>
> Lake Formation uses `GetTemplateInstance`, `GetTemplateInstances`, and `InstantiateTemplate` operations to create workflows from blueprints. These operations are not publicly available, and are used only internally for creating resources on your behalf. You receive CloudTrail events for creating workflows.

To create a workflow from a blueprint

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator or as a user who has data engineer permissions. For more information, see Lake Formation personas and IAM permissions reference.
2. In the navigation pane, choose **Blueprints**, and then choose **Use blueprint**.
3. On the **Use a blueprint** page, choose a tile to select the blueprint type.
4. Under **Import source**, specify the data source.

   If you are importing from a JDBC source, specify the following:

   - **Database connection**–Choose a connection from the list. Create additional connections using the AWS Glue console. The JDBC user name and password in the connection determine the database objects that the workflow has access to.
   - **Source data path**–Enter *<database>*/*<schema>*/*<table>* or *<database>*/*<table>*, depending on the database product. Oracle Database and MySQL don't support schema in the path. You can substitute the percent (%) character for *<schema>* or *<table>*. For example, for an Oracle database with a system identifier (SID) of `orcl`, enter `orcl/%` to import all tables that the user named in the connection has access to.

> **Important**
>
> This field is case sensitive. The workflow will fail if there is a case mismatch for any of the components.

If you specify a MySQL database, AWS Glue ETL uses the Mysql5 JDBC driver by default, so MySQL8 is not natively supported. You can edit the ETL job script to use a `customJdbcDriverS3Path` parameter as described in JDBC connectionType Values in the *AWS Glue Developer Guide* to use a different JDBC driver that supports MySQL8.

If you are importing from a log file, ensure that the role that you specify for the workflow (the "workflow role") has the required IAM permissions to access the data source. For example, to import AWS CloudTrail logs, the user must have the `cloudtrail:DescribeTrails` and `cloudtrail:LookupEvents` permissions to see the list of CloudTrail logs while creating the workflow, and the workflow role must have permissions on the CloudTrail location in Amazon S3.

5. Do one of the following:

   - For the **Database snapshot** blueprint type, optionally identify a subset of data to import by specifying one or more exclude patterns. These exclude patterns are Unix-style `glob` patterns. They are stored as a property of the tables that are created by the workflow.

     For details on the available exclude patterns, see Include and Exclude Patterns in the *AWS Glue Developer Guide*.

   - For the **Incremental database** blueprint type, specify the following fields. Add a row for each table to import.

     Table name

     Table to import. Must be all lower case.

     Bookmark keys

     Comma-delimited list of column names that define the bookmark keys. If blank, the primary key is used to determine new data. Case for each column must match the case as defined in the data source.

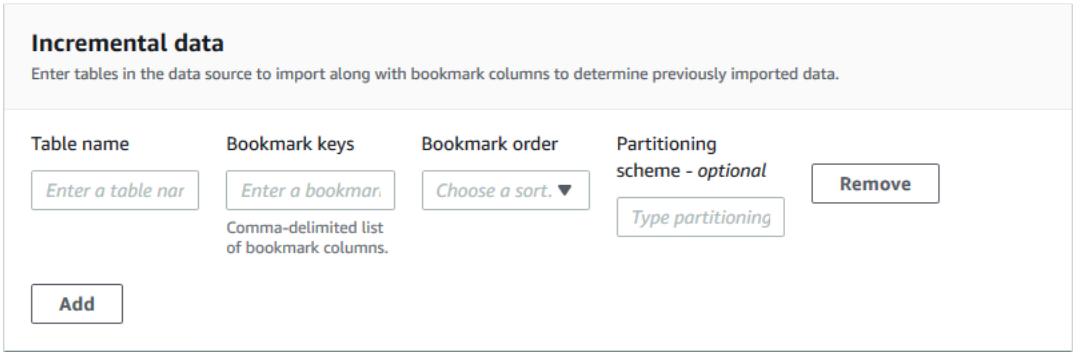
> **Note**
>
> The primary key qualifies as the default bookmark key only if it is sequentially increasing or decreasing (with no gaps). If you want to use the primary key as the bookmark key and it has gaps, you must name the primary key column as a bookmark key.

Bookmark order

When you choose **Ascending**, rows with values greater than bookmarked values are identified as new rows. When you choose **Descending**, rows with values less than bookmarked values are identified as new rows.

Partitioning scheme

(Optional) List of partitioning key columns, delimited by slashes (/). Example: `year/month/day`.



For more information, see Tracking Processed Data Using Job Bookmarks in the *AWS Glue Developer Guide*.

6. Under **Import target**, specify the target database, target Amazon S3 location, and data format.

   Ensure that the workflow role has the required Lake Formation permissions on the database and Amazon S3 target location.

> **Note**
>
> Currently, blueprints do not support encrypting data at the target.

7. Choose an import frequency.

   You can specify a `cron` expression with the **Custom** option.

8. Under **Import options**:

   a. Enter a workflow name.

   b. For role, choose the role `LakeFormationWorkflowRole`, which you created in [(Optional) Create an IAM role for workflows].

   c. Optionally specify a table prefix. The prefix is prepended to the names of Data Catalog tables that the workflow creates.

9. Choose **Create**, and wait for the console to report that the workflow was successfully created.

> **Tip**
>
> Did you get the following error message?
> `User: arn:aws:iam::<account-id>:user/<username> is not authorized to perform: iam:PassRole on resource:arn:aws:iam::<account-id>:role/<rolename>...`
> If so, check that you replaced *<account-id>* with a valid AWS account number in all policies.

> **See also:**
>
> - Blueprints and workflows in Lake Formation

Running a workflow

You can run a workflow using the Lake Formation console, the AWS Glue console, or the AWS Glue Command Line Interface (AWS CLI), or API.

To run a workflow (Lake Formation console)

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as the data lake administrator or as a user who has data engineer permissions. For more information, see Lake Formation personas and IAM permissions reference.

2. In the navigation pane, choose **Blueprints**.
3. On the **Blueprints** page, select the workflow. Then on the **Actions** menu, choose **Start**.
4. As the workflow runs, view its progress in the **Last run status** column. Choose the refresh button occasionally.

   The status goes from **RUNNING**, to **Discovering**, to **Importing**, to **COMPLETED**.

   When the workflow is complete:

   - The Data Catalog has new metadata tables.
   - Your data is ingested into the data lake.

   If the workflow fails, do the following:

   a. Select the workflow. Choose **Actions**, and then choose **View graph**.

      The workflow opens in the AWS Glue console.

   b. Ensure that the workflow is selected, and choose the **History** tab.

   c. Under **History**, select the most recent run and choose **View run details**.

   d. Select a failed job or crawler in the dynamic (runtime) graph, and review the error message. Failed nodes are either red or yellow.

See also:

- Blueprints and workflows in Lake Formation

Bringing your data into the AWS Glue Data Catalog

You can create federated catalogs in the AWS Glue Data Catalog (Data Catalog) and unify data across Amazon S3 data lakes and Amazon Redshift data warehouses. You can also integrate data from your operational databases such as Amazon DynamoDB, and third-party data sources such as PostgreSQL, Google BigQuery, MySQL, among others. The Data Catalog provides a centralized metadata repository that makes managing and discovering data across disparate systems easier.

The Data Catalog integrates with over 30 external data sources through federated connectors. With this integration, you can query data from these external sources without having to build data pipelines to ingest the data into AWS first.

After cataloging the external data, you can use AWS Lake Formation to centrally manage data access permissions in the Data Catalog. Data lake administrators can grant fine-grained access permissions to other IAM principals (users or roles) within the same account or across accounts using tag-based access control (LF-Tags) and named resources methods.

By using LF-Tags data administrators can logically organize resources based on attributes such as domain and sensitivity level, simplifying permission management while ensuring consistent access controls across analytics and machine learning services including Athena, Amazon EMR, AWS Glue or Redshift Spectrum.

The Data Catalog provides the following methods to manage data and permissions on external datasets and external metastores:

- **Integrate Amazon S3 Table buckets with Data Catalog** – You can publish and catalog Amazon S3 Tables as Data Catalog objects and register the catalog as a Lake Formation data location from Lake Formation console or using AWS Glue API operations.
- **Bring data in Amazon Redshift data warehouses into the AWS Glue Data Catalog** – Register an existing Amazon Redshift namespace or a cluster with the Data Catalog, and create a multi-level federated catalog in the Data Catalog.

  You can access your data using any query engine compatible with Apache Iceberg REST catalog OpenAPI specification, such as Amazon EMR Serverless, and Amazon Athena.
- **Federate remote Iceberg REST catalogs to Data Catalog** – Federate remote Iceberg REST catalogs to Data Catalog and securely access remote Iceberg tables stored in Amazon S3 using AWS analytics engines.

- **Federate into the Data Catalog from external data sources** – Connect the Data Catalog to external data sources using AWS Glue connections, and create federated catalogs to centrally manage access permissions on datasets using Lake Formation. No migration of metadata into the Data Catalog is necessary.
- **Create catalogs to manage Amazon Redshift tables in the Data Catalog** – You might not have an Amazon Redshift producer cluster or an Amazon Redshift datashare available today, but want to create and manage Amazon Redshift tables using Data Catalog. You can get started by creating an AWS Glue managed catalog using the `glue:CreateCatalog` API operation or the AWS Lake Formation console by setting the catalog type as `Managed` and `Catalog source` as **Redshift.**
- **Publish Amazon Redshift datashares with Data Catalog** – Publish Amazon Redshift datashares to Data Catalog, and use Lake Formation to centrally manage data access of datashares and restrict user access.

  You can query your data using Amazon Redshift Spectrum.
- **Connect Data Catalog to external Hive metastores** – Connect the Data Catalog to external metastores to manage access permissions on datasets in Amazon S3 using Lake Formation. No migration of metadata into the Data Catalog is necessary.
- **Integrate Lake Formation with AWS Data Exchange** – Lake Formation supports licensing access to your data through AWS Data Exchange. If you want to license your Lake Formation data, see What is AWS Data Exchange in the *AWS Data Exchange User Guide*.

Topics

- Catalog federation to remote Iceberg catalogs
- Bringing Amazon Redshift data into the AWS Glue Data Catalog
- Federating into external data sources in the AWS Glue Data Catalog
- Amazon S3 Tables integration with AWS Glue Data Catalog and AWS Lake Formation
- Changing access controls for S3 Tables integration
- Creating an Amazon Redshift managed catalog in the AWS Glue Data Catalog
- Managing permissions for data in an Amazon Redshift datashare
- Managing permissions on datasets that use external metastores

Catalog federation to remote Iceberg catalogs

Catalog federation in AWS Glue provides direct and secure access to Iceberg tables, stored in Amazon S3 and cataloged in remote catalogs, using AWS analytics engines. Catalog federation synchronizes metadata across Data Catalog and remote catalogs when you access remote tables. It is supported by a wide variety of analytics engines, including Amazon Redshift, Amazon EMR, Amazon Athena, AWS Glue, third-party engines like Apache Spark, and more.

Catalog federation uses AWS Glue Data Catalog to communicate with remote catalog systems to discover tables and Lake Formation to authorize access to table data in Amazon S3. When you query a federated table, Data Catalog discovers the latest table information in the remote catalog at query time, getting the table's Amazon S3 location, current schema, and partition information. Your analytics engine (Amazon Athena, Amazon Redshift, Amazon EMR) then uses this information to access Iceberg data files directly from Amazon S3. Lake Formation manages access to table(s) by vending scoped credentials to the table data stored in Amazon S3, allowing the engines to apply fine-grained permissions to federated table(s).

Features of Catalog Federation

Governed using Lake Formation

Federated Iceberg catalogs in Data Catalog are Lake Formation registered resources, allowing you to grant fine-grained row-, column-, cell-level permissions to Iceberg tables in federated Iceberg catalogs using Lake Formation grants. Federated Iceberg catalogs and associated objects can be securely shared across AWS accounts. Federated Iceberg catalogs also work with Lake Formation Tag-based access control allowing you to scale governance using tags.

Networking configurations

Catalog federation supports direct connections to remote catalog sources using standard HTTPS connectivity. It also supports connectivity through Amazon VPC when you want to maintain network isolation and connectivity using proxy support when you want secure communication through organization firewalls.

Limitations

- **Iceberg table metadata size limit** – For federated Iceberg catalogs, AWS Glue Data Catalog supports a maximum metadata size of 5 MB per REST API call. AWS Glue Data Catalog rejects requests to tables with metadata exceeding this limit.

Topics

- Federate to Snowflake Iceberg Catalog
- Federate to Databricks Unity Catalog

Federate to Snowflake Iceberg Catalog

AWS Glue Data Catalog federates to Snowflake using the OAuth2 credentials of a Snowflake service principal. You can use this federation to connect to Snowflake Horizon and Snowflake Polaris catalogs. This authentication mechanism allows Data Catalog to access the metadata of various objects (such as catalogs, databases, and tables) in your catalog, based on the privileges associated with the service principal. To ensure access to the right objects, it is essential to grant the service principal in Snowflake with the necessary permissions to read metadata of these objects.

Prerequisites

Before you create a federated catalog in Data Catalog that is governed by Lake Formation, ensure you have the following permissions:

Your IAM principal (user or role) must have the following permissions:

- **Lake Formation permissions** – `lakeformation:RegisterResource`, `lakeformation:DescribeResource`
- **AWS Glue permissions** – `glue:CreateConnection`, `glue:CreateCatalog`, `glue:GetConnection`, `glue:PassConnection`
- **Secrets Manager permissions** – `secretsmanager:CreateSecret`, `secretsmanager:GetSecretValue`
- **IAM permissions** – `iam:CreateRole`, `iam:AttachRolePolicy`, `iam:PassRole`

You must be a Lake Formation data lake administrator or have `CREATE_CATALOG` permission on the Data Catalog

Create Federated Catalog

Using Console

1. Sign in to the the console and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

2. Choose the preferred AWS Region in the Region selector.
3. In the left navigation pane, choose Catalogs.
4. Choose **Create Catalog** to open the **Create Catalog Workflow**.
5. In **Choose data source** step, select Snowflake from the available options.
6. In **Set catalog details** step, you provide three information - catalog details, connection details, and registration details.
7. In **catalog details** container, provide a unique name to your AWS Glue federated catalog and enter the name of the existing Snowflake catalog.
8. In **connections details** container, you can either choose from an existing connection that you have access or provide configuration to create a new connector.
9. New connection configurations include:

   - Connection Name – A unique name of the AWS Glue connection object.
   - Instance URL – The endpoint URL of your existing Snowflake account.
   - Authentication – Specify the authentication configuration that AWS Glue uses to connect to remote catalog server. AWS Glue supports both OAuth2 and Custom authentication.
   - Token URL – Specify the URL of remote catalog's identity provider.
   - OAuth2 Client ID – (Optional) Specify the Client ID of the OAuth2 credential associated with your remote catalog. If you don't provide a client ID, the token request authenticates using only the client secret stored in AWS Secrets Manager.
   - Secret – Store and use OAuth2 client secret using AWS Secrets Manager or enter the secret value in textbox. When you enter the secret manually in console, AWS Glue creates the secret on your behalf.
   - Token URL Scope – Specify the OAuth scope for authentication.
   - Catalog Casing Filter – Choose whether to bring lowercase or uppercase objects from your remote catalog to Data Catalog.

10. Create an IAM role that AWS Glue and Lake Formation service principals can use to access secret in AWS Secrets Manager and Amazon S3 locations of remote Iceberg tables respectively. Select the IAM role in the registration dropdown. Refer to step 2 and 3 in following CLI section for IAM policy details.
11. Select Test Connection to test whether your connection properties and IAM role access are configured correctly.
12. Select **Next** to review your settings.

13. Select **Create Catalog** in review page.

Using CLI

1. **Create an AWS Secrets Manager secret**

   The AWS Glue connector supports two authentication types - **OAuth2** and **Custom**. When using OAuth2 option, use AWS Secrets Manager to store client secret of the Snowflake service principal. You will later use this secret when creating the AWS Glue Connection. For Custom authentication, use AWS Secrets Manager to store and retrieve the access token.

   In the following example, replace *your-snowflake-secret*, *client_secret*, *region* with your own information.

   ```
   aws secretsmanager create-secret \
   --name your-snowflake-secret \
   --description "Snowflake secret" \
   --secret-string '{
   "USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET": "client_secret"
   }' \
   --region region
   ```

   > **Note**
   >
   > USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET is a reserved keyword that AWS Glue uses to refer to a client secret value in the secret. Use the same keyword when you are creating the secret in Lake Formation console too.

2. **Create an IAM role which gives AWS Glue connection object access to secret created in previous step**

   The AWS Glue connection object requires access to the AWS Secrets Manager secret when you use AWS Secrets Manager to store, retrieve, and refresh your OAuth secret token. The AWS Glue connection object also requires access to create, describe, and use the Amazon VPC network interfaces when you use a Amazon VPC endpoint to restrict connectivity to your Snowflake account.

   Create an IAM policy and attach it to an IAM role. Add AWS Glue service principal to the trust policy.

In the following example, replace *your-secrets-manager-ARN*, *your-vpc-id*, and *your-subnet-id1* with your own information.

Example IAM Policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:PutSecretValue"
            ],
            "Resource": [
                "your-secrets-manager-ARN"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "ec2:Vpc": "arn:aws:ec2:region:account-id:vpc/your-vpc-id",
                    "ec2:Subnet": ["arn:aws:ec2:region:account-id:subnet/your-
subnet-id1"]
                }
            }
        }
    ]
}
```

Example Trust Policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
```

```
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

3. **Create an IAM policy that gives Lake Formation read access to catalog's Amazon S3 location**

   As the catalog owner of a federated catalog in Data Catalog, you use Lake Formation to grant coarse-grained table access, fine-grained - column-level, row-level, and cell-level - access, and tag-based access to your data teams. Lake Formation uses an IAM role that gives it access to the underlying Amazon S3 locations of your remote Iceberg tables. This access allows Lake Formation to vend scoped access credentials to analytics engines querying remote tables.

   Create IAM policy and attach to an IAM role. Add Lake Formation service principal to the role trust policy.

   In the following example, replace *amzn-s3-demo-bucketN* and *your-kms-key* with your own information.

   Example IAM Policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket2/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt"
            ],
            "Resource": [
                "your-kms-key"
            ]
        }
    ]
}
```

Example Trust Policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
            "Service": "lakeformation.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

> **Note**
>
> When you use Lake Formation console to create a federated catalog, the console uses a single IAM role with both policies attached to complete setup.

4. **Create an AWS Glue connection object**

AWS Glue connector supports OAuth2 and Custom authentication methods. Data Catalog connection object supports `CATALOG_CASING_FILTER` configuration that allows you to bring either lowercase or uppercase objects in your remote catalog to Data Catalog.

The following example uses OAuth2 authentication configuration to create an AWS Glue connection. Replace *highlighted sections* with your information.

```
aws glue create-connection \
    --connection-input '{
"Name": "your-glue-connection-to-snowflake-account",
"ConnectionType": "SNOWFLAKEICEBERGRESTCATALOG",
"ConnectionProperties": {
    "INSTANCE_URL": "your-snowflake-account-URL",
    "ROLE_ARN": "your-IAM-role-for-secrets-and-VPC-access",
    "CATALOG_CASING_FILTER": "LOWERCASE_ONLY"
},
"AuthenticationConfiguration": {
    "AuthenticationType": "OAUTH2",
    "OAuth2Properties": {
        "OAuth2GrantType": "CLIENT_CREDENTIALS",
        "TokenUrl": "your-internal-or-external-token-server-url",
        "OAuth2ClientApplication": {
            "UserManagedClientApplicationClientId": "our-client-id"
        },
        "TokenUrlParametersMap": {
            "scope": "all-apis"
        }
    },
    "SecretArn": "arn:aws:secretsmanager:your-aws-region:your-aws-account-
id:secret:snowflake-secret"
}
}'
```

> **Note**
>
> The `OAuth2ClientApplication` field is optional. If you omit it, the token request authenticates using only the client secret stored in AWS Secrets Manager, without sending a client ID.

5. **Register AWS Glue connection as a Lake Formation resource**

Using the AWS Glue connection object (created in Step 4) and IAM role (created in Step 3), you can now register the AWS Glue connection object as a Lake Formation managed resource.

Replace *your-glue-connector-arn* and *your-IAM-role-ARN-having-LF-access* with your information.

```
aws lakeformation register-resource \
    --resource-arn your-glue-connector-arn \
    --role-arn your-IAM-role-ARN-having-LF-access \
    --with-federation \
    --with-privileged-access
```

6. **Create a federated catalog in Data Catalog**

   After creating an AWS Glue connection object and registering it with Lake Formation, you can create a federated catalog in the Data Catalog:

   Provide the federated catalog a unique name at *your-federated-catalog-name*, reference the catalog in Snowflake at *catalog-name-in-Snowflake*, and input connection name created earlier at *your-glue-connection-name*.

```
aws glue create-catalog \
    --name your-federated-catalog-name \
    --catalog-input '{
    "FederatedCatalog": {
        "Identifier": catalog-name-in-Snowflake",
        "ConnectionName": your-glue-connection-name"
    },
    "CreateTableDefaultPermissions": [],
    "CreateDatabaseDefaultPermissions": []
}'
```

Considerations when integrating with Snowflake

- When you drop resources (like databases and tables) in Snowflake, Lake Formation does not automatically revoke the permissions granted on that federated resource. To remove the access permissions, you need to explicitly revoke the permissions that were previously granted on the federated resource using Lake Formation.
- When you mount remote catalog with `CATALOG_CASING_FILTER='UPPERCASE_ONLY'` configuration, databases and tables with uppercase identifiers are federated but objects with lowercase identifiers are not.

- You can reuse the same AWS Glue connection to create multiple federated catalogs. Deleting a catalog will not delete the associated connection. To delete a connection, please use AWS CLI `aws glue delete-connection` command and ensure all associated catalogs are deleted first.
- Nested namespaces in Polaris catalog are not supported. That is, catalog federation can access remote Iceberg tables that follow a 3-part notation `catalog.database.table`.

Federate to Databricks Unity Catalog

AWS Glue Data Catalog federates to Databricks using the OAuth2 credentials of a Databricks service principal. This authentication mechanism allows AWS Glue Data Catalog to access the metadata of various objects (such as catalogs, databases, and tables) in Databricks Unity Catalog, based on the privileges associated with the service principal. To ensure access to the right objects, it is essential to grant the service principal with the necessary permissions in Databricks to read metadata of these objects.

Next, catalog federation enables discovery and query of Iceberg tables in your Databricks Unity Catalog. To read delta tables, please ensure Iceberg metadata is available for these tables using Uniform. Follow the Databricks tutorial and documentation to create the service principal and associated privileges in your Databricks workspace.

Prerequisites

Before you create a federated catalog in Data Catalog that is governed by Lake Formation, ensure you have the following permissions:

Your IAM principal (user or role) must have the following permissions:

- **Lake Formation permissions** – `lakeformation:RegisterResource`, `lakeformation:DescribeResource`
- **AWS Glue permissions** – `glue:CreateConnection`, `glue:CreateCatalog`, `glue:GetConnection`, `glue:PassConnection`
- **Secrets Manager permissions** – `secretsmanager:CreateSecret`, `secretsmanager:GetSecretValue`
- **IAM permissions** – `iam:CreateRole`, `iam:AttachRolePolicy`, `iam:PassRole`

You must be a Lake Formation data lake administrator or have `CREATE_CATALOG` permission on the Data Catalog

Create Federated Catalog

Using Console

1. Sign in to the the console and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. Choose the preferred AWS Region in the Region selector.
3. In the left navigation pane, choose Catalogs.
4. Choose **Create Catalog** to open the **Create Catalog Workflow**.
5. In **Choose data source** step, select Databricks from the available options.
6. In **Set catalog details** step, you provide three information - catalog details, connection details, and registration details.
7. In **catalog details** container, provide a unique name to your AWS Glue federated catalog and enter the name of the existing Databricks catalog.
8. In **connections details** container, you can either choose from an existing connection that you have access or provide configuration to create a new connector.
9. New connection configurations include:
   - Connection Name – A unique name of the AWS Glue connection object.
   - Workspace URL – The endpoint URL of your existing Databricksworkspace.
   - Authentication – Specify the authentication configuration that AWS Glue uses to connect to remote catalog server. AWS Glue supports both OAuth2 and Custom authentication.
   - Token URL – Specify the URL of remote catalog's identity provider.
   - OAuth2 Client ID – Specify the Client ID of the OAuth2 credential associated with your remote catalog.
   - Secret – Store and use OAuth2 client secret using AWS Secrets Manager or enter the secret value in textbox. When you enter the secret manually in console, AWS Glue creates the secret on your behalf.
   - Token URL Scope – Specify the OAuth scope for authentication.
10. Create an IAM role that AWS Glue and Lake Formation service principals can use to access secret in and Amazon S3 locations of remote Iceberg tables respectively. Select the IAM role in the registration dropdown. Refer to step 2 and 3 in following CLI section for IAM policy details.

11. Select Test Connection to test whether your connection properties and IAM role access are configured correctly. Test connection functionality is not available when connecting to Databricks using Amazon VPC.
12. Select **Next** to review your settings.
13. Select **Create Catalog** in review page.

Using CLI

1. **Create an AWS Secrets Manager secret**

   The AWS Glue connector supports two authentication types - **OAuth2** and **Custom**. When using OAuth2 option, use AWS Secrets Manager to store client secret of the Databricks service principal. You will later use this secret when creating the AWS Glue Connection. For Custom authentication, use AWS Secrets Manager to store and retrieve the access token.

   In the following example, replace *<databricks-secret>*, *<client_secret>*, *<region>* with your own information.

   ```
   aws secretsmanager create-secret \
   --name <databricks-secret> \
   --description "Databricks secret" \
   --secret-string '{
   "USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET": "<client_secret>"
   }' \
   --region <region>
   ```

   > **Note**
   >
   > USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET is a reserved keyword that AWS Glue uses to refer to a client secret value in the secret. Use the same keyword when you are creating the secret in Lake Formation console too.

2. **Create an IAM role which gives AWS Glue connection object access to secret created in previous step**

   The AWS Glue connection object requires access to the AWS Secrets Manager secret when you use AWS Secrets Manager to store, retrieve, and refresh your OAuth secret token. The AWS Glue connection object also requires access to create, describe, and use the Amazon

VPC network interfaces when you use a Amazon VPC endpoint to restrict connectivity to your Databricks workspace.

Create an IAM policy and attach it to an IAM role. Add AWS Glue service principal to the trust policy.

In the following example, replace *<your-secrets-manager-ARN>*, *<your-vpc-id>*, and *<your-subnet-id1>* with your own information.

Example IAM Policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:PutSecretValue"
            ],
            "Resource": [
                "<your-secrets-manager-ARN>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "ec2:Vpc": "arn:aws:ec2:region:account-id:vpc/<your-vpc-id>",
                    "ec2:Subnet": ["arn:aws:ec2:region:account-id:subnet/<your-
subnet-id1>"]
                }
            }
        }
    ]
}
```

Example Trust Policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

3. **Create an IAM policy that gives Lake Formation read access to catalog's Amazon S3 location**

   As the catalog owner of a federated catalog in Data Catalog, you use Lake Formation to grant coarse-grained table access, fine-grained - column-level, row-level, and cell-level - access, and tag-based access to your data teams. Lake Formation uses an IAM role that gives it access to the underlying Amazon S3 locations of your remote Iceberg tables. This access allows Lake Formation to vend scoped access credentials to analytics engines querying remote tables.

   Create IAM policy and attach to an IAM role. Add Lake Formation service principal to the IAM role trust policy.

   In the following example, replace *<your-s3-bucket-N>* and *<your-kms-key>* with your own information.

   Example IAM Policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::<your-s3-bucket-1>/*",
                "arn:aws:s3:::<your-s3-bucket-2>/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<your-s3-bucket-1>",
                "arn:aws:s3:::<your-s3-bucket-2>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt"
            ],
            "Resource": [
                "<your-kms-key>"
            ]
        }
    ]
}
```

Example Trust Policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
            "Service": "lakeformation.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

> **Note**
>
> When you use Lake Formation console to create a federated catalog, the console uses a single IAM role with both policies attached to complete setup.

4. **Create an AWS Glue connection object**

   Data Catalog supports connectionType: DATABRICKSICEBERGRESTCATALOG for connecting Data Catalog to Databricks. This AWS Glue connector supports OAuth2 and Custom authentication methods.

   The following example uses OAuth2 authentication configuration to create an AWS Glue connection. Replace *highlighed sections* with your information.

```
aws glue create-connection \
    --connection-input '{
"Name": "<your-glue-connection-to-databricks-unity-account>",
"ConnectionType": "DATABRICKSICEBERGRESTCATALOG",
"ConnectionProperties": {
    "INSTANCE_URL": "<your-databricks-workspace-catalog-URL>",
    "ROLE_ARN": "<your-IAM-role-for-secrets-and-VPC-access>"
},
"AuthenticationConfiguration": {
    "AuthenticationType": "OAUTH2",
    "OAuth2Properties": {
        "OAuth2GrantType": "CLIENT_CREDENTIALS",
        "TokenUrl": "<your-internal-or-external-token-server-url>",
        "OAuth2ClientApplication": {
            "UserManagedClientApplicationClientId": "<your-client-id>"
        },
        "TokenUrlParametersMap": {
            "scope": "all-apis"
        }
    },
    "SecretArn": "arn:aws:secretsmanager:<aws-region>:<your-aws-account-
id>:secret:<databricks-secret>"
}
}'
```

5. **Register AWS Glue connection as a Lake Formation resource**

   Using the AWS Glue connection object (created in Step 4) and IAM role (created in Step 3), you can now register the AWS Glue connection object as a Lake Formation managed resource.

   Replace *<your-glue-connector-arn>* and *<your-IAM-role-ARN-having-LF-access>* with your information.

```
aws lakeformation register-resource \
    --resource-arn <your-glue-connector-arn> \
    --role-arn <your-IAM-role-ARN-having-LF-access> \
    --with-federation \
    --with-privileged-access
```

6. **Create a federated catalog in Data Catalog**

   After creating an AWS Glue connection object and registering it with Lake Formation, you can create a federated catalog in the Data Catalog:

   Provide the federated catalog a unique name at *<your-federated-catalog-name>*, reference the catalog in Databricks at *"<catalog-name-in-Databricks>*, and input connection name created earlier at *"<your-glue-connection-name>*.

```
aws glue create-catalog \
    --name <your-federated-catalog-name> \
    --catalog-input '{
    "FederatedCatalog": {
        "Identifier": "<catalog-name-in-Databricks>",
        "ConnectionName": "<your-glue-connection-name>"
    },
    "CreateTableDefaultPermissions": [],
    "CreateDatabaseDefaultPermissions": []
}'
```

Considerations when integrating with Databricks

- When you drop resources (like databases and tables) in the Databricks, Lake Formation does not automatically revoke the permissions granted on that federated resource. To remove the access permissions, you need to explicitly revoke the permissions that were previously granted on the federated resource using Lake Formation.
- You can query Iceberg tables stored in Amazon S3 using this integration. When using any other table format or object storage, you can federate metadata in remote catalogs to AWS Glue and list its databases and tables but query operations like `SELECT ColumnFoo from TableBar` will fail during query with error 'Failed to read Apache Iceberg table. Object storage location is not supported.'

- You can reuse the same AWS Glue connection to create multiple federated catalogs. Deleting a catalog will not delete the associated connection object. To delete a connection object, please use AWS CLI `aws glue delete-connection` command and ensure all associated catalogs are deleted first.

Bringing Amazon Redshift data into the AWS Glue Data Catalog

You can manage analytic data in Amazon Redshift data warehouses in the AWS Glue Data Catalog (Data Catalog), and unify Amazon S3 data lakes and Amazon Redshift data warehouses. Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the AWS Cloud. An Amazon Redshift data warehouse is a collection of computing resources called *nodes*, which are organized into a group called a *cluster*. Each cluster runs an Amazon Redshift engine and contains one or more databases.

In Amazon Redshift, you can create Amazon Redshift provisioned clusters and serverless namespaces, and register them with the Data Catalog. By doing this, you can unify data in Amazon Redshift managed storage (RMS) and Amazon S3 buckets, and access data from Apache Iceberg compatible analytical engines.

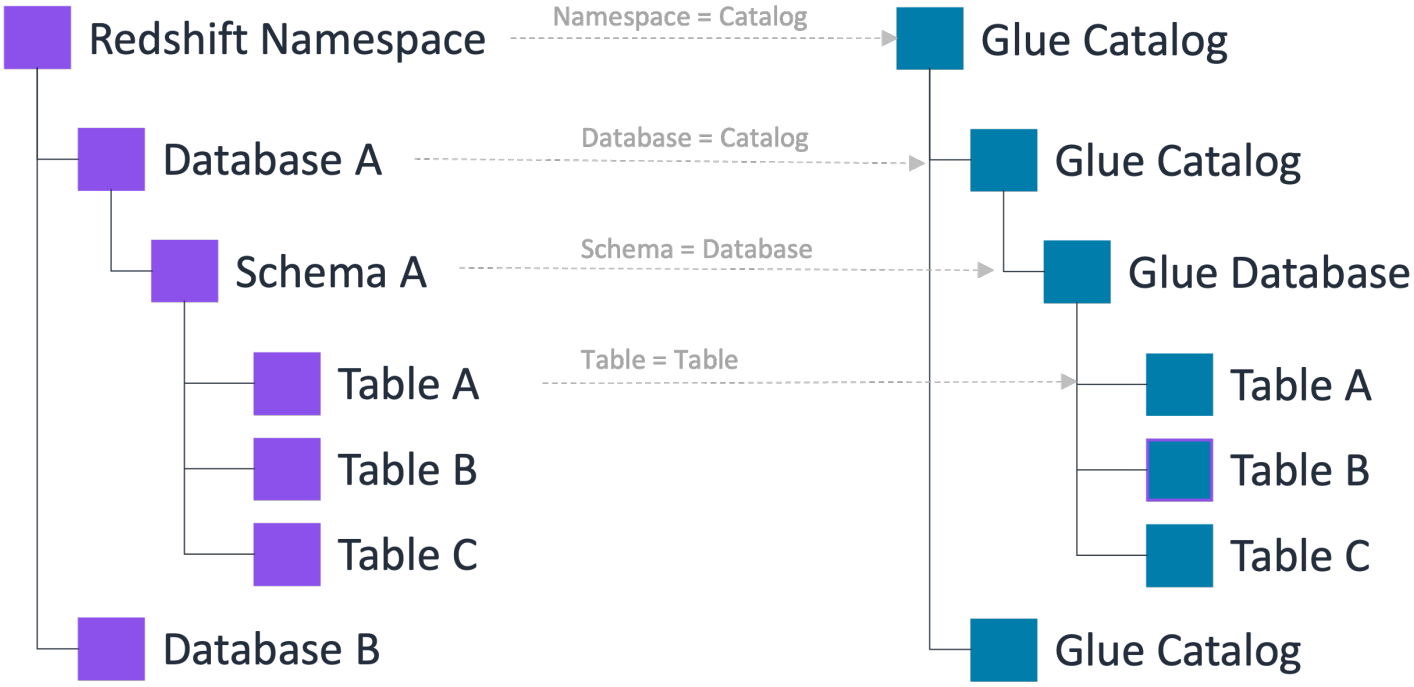
By registering namespaces and clusters, you can provide access to data without the need to copy it or move it. For more information about registering clusters and namespaces in Amazon Redshift, see Registering Amazon Redshift clusters and namespaces to the AWS Glue Data Catalog.

In Amazon Redshift, you can perform data sharing through datashares or by registering namespaces and clusters with Data Catalog. With datashares, which operate at the individual database object level, you have to enable sharing for each table or view. In contrast, namespace publishing functions at the cluster or namespace level. When you register a cluster or namespace with the Data Catalog, all databases and tables within it are automatically shared, without you having to configure sharing for individual objects.

In the Data Catalog, you can create a federated catalog for each namespace or cluster. A catalog is referred to as a *federated catalog* when it points to an entity outside of the Data Catalog. Tables and views in the Amazon Redshift namespace are listed as individual tables in the Data Catalog. You can share databases and tables in the federated catalog with selected IAM principals and SAML users within the same account, or in another account with Lake Formation. You can also include row and column filter expressions to restrict access to certain data. For more information, see Data filtering and cell-level security in Lake Formation.

The Data Catalog supports a three-level metadata hierarchy comprising catalogs, databases, and tables (and views). When you register a namespace with the Data Catalog, the Amazon Redshift data hierarchy is mapped to the Data Catalog's 3-level hierarchy as follows:

- The Amazon Redshift namespace becomes a multi-level catalog in the Data Catalog.
- The associated Amazon Redshift database is registered as a catalog in the Data Catalog.
- The Amazon Redshift schema becomes a database in the Data Catalog.
- The Amazon Redshift table becomes a table in the Data Catalog.



With this three-level metadata hierarchy, you can access Amazon Redshift tables by using the 3-part notation - "catalog1/catalog2.database.table" in the Data Catalog. Also, data teams can maintain the same organization that Amazon Redshift uses to organize tables within the Data Catalog account.

In Lake Formation, you can securely manage the data from Amazon Redshift using fine-grained access control for the Data Catalog resources. With this integration, you can manage, secure, and query analytical data from a single catalog with a common access control mechanism.

For limitations, see Limitations for bringing Amazon Redshift data warehouse data into the AWS Glue Data Catalog.

Topics

- Key benefits
- Roles and responsibilities
- Prerequisites for managing Amazon Redshift namespaces in the AWS Glue Data Catalog
- Creating Amazon Redshift federated catalogs
- Viewing catalog objects
- Updating a federated catalog
- Accessing a shared federated catalog
- Deleting a federated catalog
- Querying federated catalogs
- Additional resources

Key benefits

Registering Amazon Redshift clusters and namespaces with the AWS Glue Data Catalog and unifying data across Amazon S3 data lakes and Amazon Redshift data warehouses, offers the following benefits:

- **Uniform querying experience** – Query your Amazon Redshift managed data and data in the Amazon S3 buckets using any query engine compatible with Apache Iceberg, such as Amazon EMR Serverless and Amazon Athena without having to move or copy data.
- **Consistent data access across services** – You don't need to update database and table names in your data pipelines when accessing the same federated data sources from different AWS analytics services, as the data sources are registered in the Data Catalog.
- **Fine-grained access control** – You can apply Lake Formation permissions to manage access to the federated data sources using fine-grained access control permissions.

Roles and responsibilities

Role	Responsibility
Amazon Redshift producer cluster administrator	Registers the cluster or namespace with the Data Catalog.

Lake Formation data lake administrator	Accepts the cluster or namespace invitation, creates federated catalogs, and grants access on the federated catalogs to other principals.
Lake Formation read only administrator	Discovers the federated catalog, queries Amazon Redshift tables in the federated catalog.
Data transfer role	Amazon Redshift assumes on your behalf to transfer data to and from the Amazon S3 bucket.

The following are the high-level steps to provide users access to an Amazon Redshift namespace:

1. In Amazon Redshift, the producer cluster administrator registers a cluster or namespace with the Data Catalog.
2. The data lake administrator accepts the namespace invitation from the Amazon Redshift producer cluster administrator, and creates a federated catalog in the Data Catalog.

   After completing this step, you can manage the Amazon Redshift namespace catalog within the Data Catalog.
3. Grant permissions to users on catalogs, databases and tables. You can share the entire namespace catalog or a subset of tables with users in the same account or another account.

Prerequisites for managing Amazon Redshift namespaces in the AWS Glue Data Catalog

1. Create a data lake administrator - Create an IAM role that is authorized to accept the namespace invitation, and creates the AWS Glue Data Catalog objects (catalogs, databases, tables/views), and grant Lake Formation permissions to other users.

   For step-by-step instructions on creating a data lake administrator, see Create a data lake administrator.
2. Update data lake administrator permissions.

In addition to data lake administrator permissions, the data lake administrator requires the following permissions to accept an Amazon Redshift namespace invitation in Lake Formation, create or update the Data Catalog resources, and enable data lake access:

JSON

```
{
"Version":"2012-10-17",
"Id": "glue-enable-datalake-access",
"Statement": [{
        "Effect": "Allow",
        "Action": [
            "redshift:AssociateDataShareConsumer",
            "redshift:DescribeDataSharesForConsumer",
            "redshift:DescribeDataShares",
            "redshift-serverless:CreateNamespace",
            "redshift-serverless:CreateWorkgroup",
            "redshift-serverless:DeleteNamespace",
            "redshift-serverless:DeleteWorkgroup",
            "ec2:DescribeAccountAttributes",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeAvailabilityZones",
            "s3:createBucket",
            "s3:deleteBucket",
            "s3:putBucketPolicy",
            "s3:putEncryptionConfiguration",
            "s3:putLifecycleConfiguration",
            "s3:putBucketVersioning",
            "iam:CreateRole"
        ],
        "Resource": "*"
    },
    {
        "Action": [
            "iam:PassRole"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iam::*:role/data transfer role name",
        "Condition": {
            "StringLike": {
                "iam:PassedToService": [
```

```
                    "glue.amazonaws.com"
                ]
            }
        }
    }
  ]
}
```

3. If the IAM role used for creating federated catalogs is not a data lake administrator, you need to grant the role the `Create catalog` permission.

   To create catalog creators

   a. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   b. Choose **Administrative roles and tasks** under **Administration**.

   c. Choose **Grant.**

   d. On the **Grant permissions** screen, choose an IAM user or role.

   e. Select **Create catalog** permission.

   f. Optionally, you can also grant grantable **Create catalog** permission. Grantable permission allows the catalog creator to grant the `Create catalog` permission to other principals.

   g. Choose **Grant**.

   AWS CLI example for granting permissions to create a federated catalog.

```
aws lakeformation grant-permissions \
--cli-input-json \
'{
    "Principal": {
     "DataLakePrincipalIdentifier":"arn:aws:iam::123456789012:role/Admin"
    },
    "Resource": {
        "Catalog": {
        }
    },
    "Permissions": [
        "CREATE_CATALOG",
        "DESCRIBE"
    ]
}'
```

4. Create a read only administrator role to discover the Amazon Redshift federated catalogs in the Data Catalog from Amazon Redshift Query Editor v2.

   To query the Amazon Redshift tables in the federated catalog from Amazon Redshift Query Editor v2, ensure that the Read only administrator role policy contains the ARN for the Amazon Redshift service-linked role-`AWSServiceRoleForRedshift`.

```
 aws lakeformation put-data-lake-settings
       --region us-east-1 \
       --data-lake-settings \
 '{
   "DataLakeAdmins": [{"DataLakePrincipalIdentifier":"arn:aws:iam::123456789012:role/
Admin"}],
   "ReadOnlyAdmins":[{"DataLakePrincipalIdentifier":"arn:aws:iam::123456789012:role/
aws-service-role/redshift.amazonaws.com/AWSServiceRoleForRedshift"}],
   "CreateDatabaseDefaultPermissions":[],
   "CreateTableDefaultPermissions":[],
   "Parameters":{"CROSS_ACCOUNT_VERSION":"4","SET_CONTEXT":"TRUE"}
  }'
```

5. Create a data transfer role that Amazon Redshift can assume on your behalf to transfer data to and from the Amazon S3 bucket.

   When you enable data lake access for Apache Iceberg compatible query engines such as Athena, Amazon EMR on Amazon EC2 to access the Amazon Redshift resources in the Data Catalog, you need to create an IAM role with the required permissions to perform data transfer to and from the Amazon S3 bucket.

   JSON

```
{
    "Version":"2012-10-17",
    "Id": "glue-enable-datalake-access",
    "Statement": [{
      "Sid": "DataTransferRolePolicy",
        "Effect": "Allow",
        "Action": [ "glue:GetCatalog",
                    "glue:GetDatabase",
                    "kms:GenerateDataKey",
```

```
                    "kms:Decrypt"],
        "Resource": "*"
    }
    ]
}
```

6. Add the following trust policy to the data transfer role for AWS Glue and Amazon Redshift services to assume the role to transfer data to and from the Amazon S3 bucket.

   JSON

```
      {
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
        "Service": [
        "redshift.amazonaws.com",
        "glue.amazonaws.com"
        ]
    },
        "Action": "sts:AssumeRole"
    }]
}
```

7. Add the following key policy to the AWS KMS key if you're using a customer managed key to encrypt the data in the Amazon Redshift cluster/namespace. Replace the account number with a valid AWS account number, and specify data transfer role name. By default, the data in the Amazon Redshift cluster is encrypted using an KMS key. Lake Formation provides an option to create your custom KMS key for encryption. If you're using a customer managed key, you must add specific key policies to the key.

   For more information about managing the permissions of a customer managed key, see Customer managed keys.

   JSON

```
{
    "Version":"2012-10-17",
    "Id": "auto-redshift-3",
```

```
    "Statement": [{
            "Sid": "RedshiftAllowAccessPolicy",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:CallerAccount": "111122223333",
                    "kms:ViaService": "redshift.us-east-1.amazonaws.com"
                }
            }
        },
        {
            "Sid": "RedshiftServerlessAllowAccessPolicy",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:CallerAccount": "111122223333",
                    "kms:ViaService": "redshift-serverless.us-
east-1.amazonaws.com"
                }
            }
```

```
        },
        {
            "Sid": "DirectMetadataAccess",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "kms:Describe*",
                "kms:Get*",
                "kms:List*",
                "kms:RevokeGrant"
            ],
            "Resource": "*"
        },
        {
            "Sid": "GenerateDataKeyDecryptDataTransferRole",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/data-transfer-role-name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                "kms:ViaService": "s3.us-east-1.amazonaws.com"
                }
            }
        }
    ]
}
```

Creating Amazon Redshift federated catalogs

This topic describes the steps you need to follow to accept a cluster or namespace invitation, create a federated multi-level catalog, and grant permissions to other principals. You can complete these tasks using the Lake Formation console, the AWS Command Line Interface (AWS CLI), or APIs/SDKs.

The examples in this topic show the producer cluster/namespace, the Data Catalog, and the data consumer in the same account.

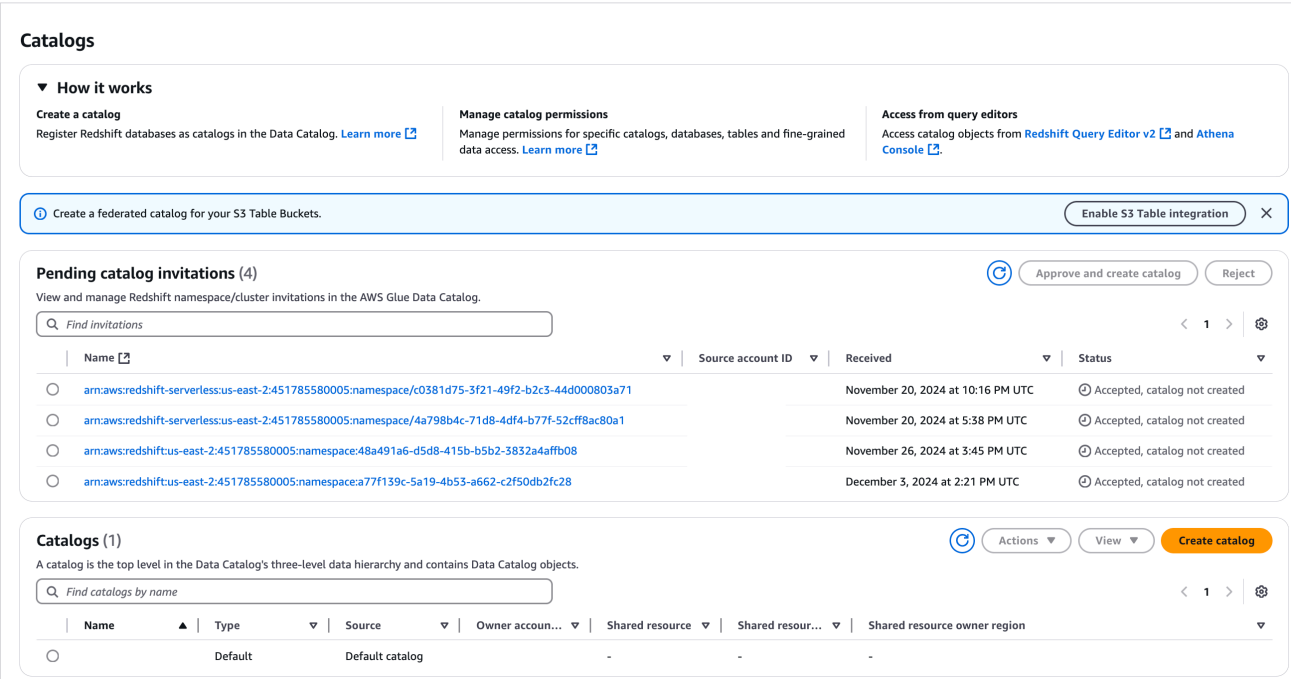
To learn more about Lake Formation cross-account capabilities, see Cross-account data sharing in Lake Formation.

To manage a Amazon Redshift namespace in the Data Catalog

1. Review a namespace invitation and accept it.

   Console

   1. Sign in to the Lake Formation console as a data lake administrator at https://console.aws.amazon.com/lakeformation/. Navigate to the **Catalogs** page under **Data Catalog**.
   2. Review the namespace invitation that you're authorized to access. The **Status** column indicates your current participation status for the namespace. The **Not accepted** status indicates that you have been added to the namespace, but you have not yet accepted it or have rejected the invitation.



   3. To respond to a namespace or cluster invitation, select the invitation name and choose **Review invitation**. In **Accept or reject invitation**, review the invitation details. Choose **Accept** to accept the invitation or **Reject** to decline the invitation. You don't get access to the namespace if you reject the invitation.

AWS CLI

The following examples show how to view, accept, and register the invitation. Replace the AWS account ID with a valid AWS account ID. Replace the `data-share-arn` with the actual Amazon Resource Name (ARN) that references the namespace.

1. View a pending invitation.

```
aws redshift describe-data-shares \
 --data-share-arn 'arn:aws:redshift:us-
east-1:123456789012:datashare:abcd1234-1234-ab12-cd34-1a2b3c4d5e6f/
ds_internal_namespace' \
```

2. Accept an invitation.

```
 aws redshift associate-data-share-consumer \
 --data-share-arn 'arn:aws:redshift:us-
east-1:123456789012:datashare:abcd1234-1234-ab12-cd34-1a2b3c4d5e6f/
ds_internal_namespace' \
 --consumer-arn 'arn:aws:glue:us-east-1:123456789012:catalog'
```

3. Register the cluster or namespace in the Lake Formation account. Use the RegisterResource API operation to register the datashare in Lake Formation. `DataShareArn` is the input parameter for `ResourceArn`.

> **Note**
>
> This is a mandatory step.

```
aws lakeformation register-resource \
 --resource-arn 'arn:aws:redshift:us-
east-1:123456789012:datashare:abcd1234-1234-ab12-cd34-1a2b3c4d5e6f/
ds_internal_namespace'
```

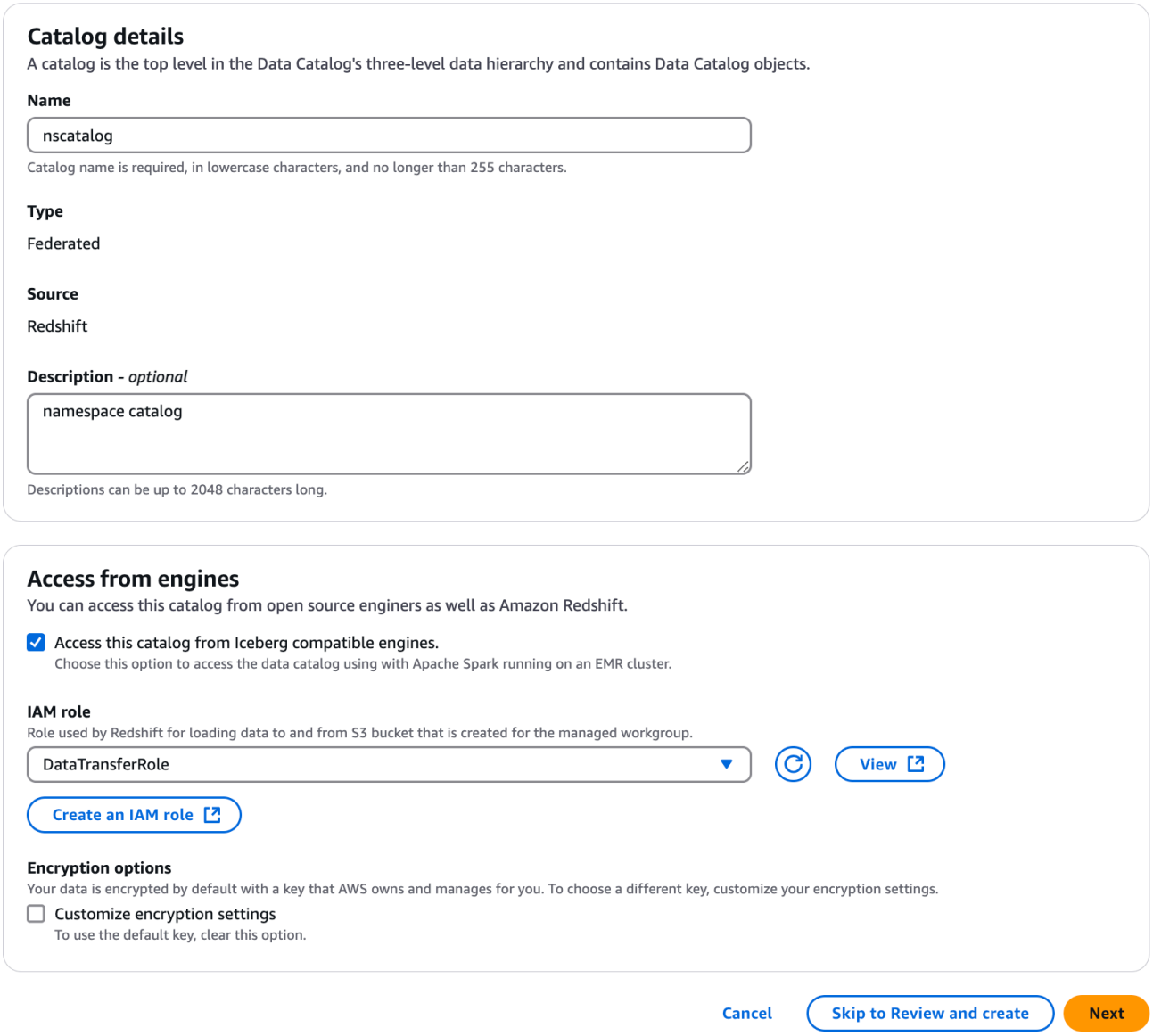
2. Create a federated catalog.

After you've accepted an invitation, you need to create a federated catalog in the Data Catalog that maps the objects in the Amazon Redshift namespace to the Data Catalog. You must be a data lake administrator or a user or role that has required permissions to create a catalog.

Console

1. After accepting the namespace **Invitation**, the **Set catalog details** page is displayed.
2. On the **Set catalog details** page, enter a unique name for the catalog. Use lower case for catalog names. Catalog names must be less than or equal to 255 characters long. You use this identifier for mapping the namespace internally in the metadata hierarchy (catalogid.dbName.schema.table).
3. Enter a description for the catalog. Description must be less than or equal to 2048 characters long.
4. Next, choose the **Access this catalog from Iceberg compatible engines** check box to enable accessing the Amazon Redshift resources using Apache Iceberg compatible analytical engines such as Athena and Apache Spark on Amazon EMR.

   You don't need to enable data lake access to access the federated catalogs using Amazon Redshift.



5. To enable these query engines to read and write to Amazon Redshift namespaces, AWS Glue creates a managed Amazon Redshift cluster with the compute and storage resources required to perform read and write operations without impacting Amazon Redshift data warehouse workloads.

   You also need to provide an IAM role with the permissions required to transfer data to and from the Amazon S3 bucket.

6. By default, the data in the Amazon Redshift cluster is encrypted using an AWS managed key. Lake Formation provides an option to create your custom KMS key for encryption. If you're using a customer managed key, you must add specific key policies to the key.

Choose the **Customize encryption settings** if you're using a customer managed key to encrypt the data in the Amazon Redshift cluster/namespace. To use a custom key, you must add additional custom managed key policy to your KMS key. For more information, see Prerequisites for managing Amazon Redshift namespaces in the AWS Glue Data Catalog.

AWS CLI

Use the following example code to create a catalog with the Amazon Redshift data published to the Data Catalog using the AWS CLI.

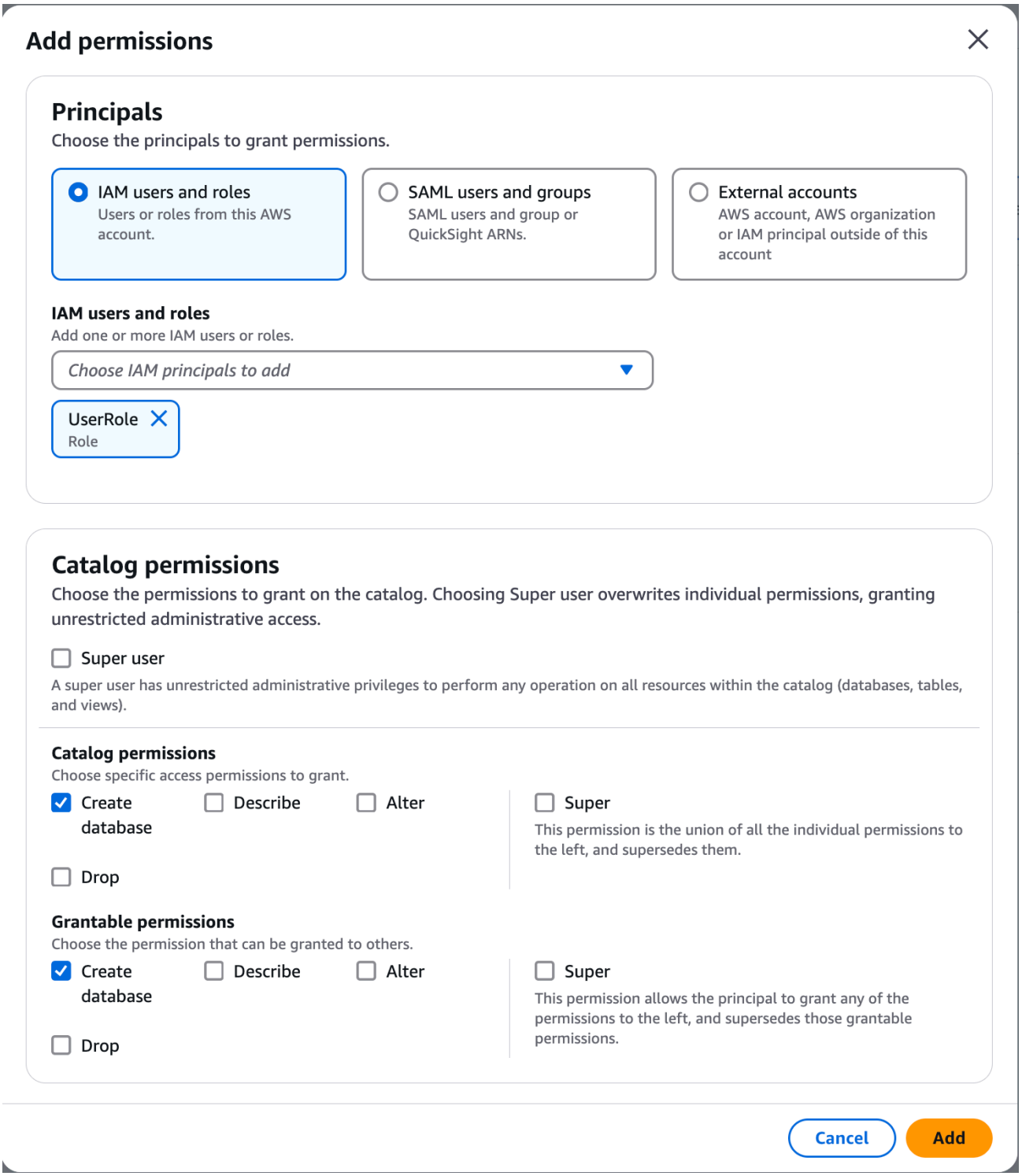
```
aws glue create-catalog
--cli-input-json \
'{
    "Name": "nscatalog",
    "CatalogInput": {
        "Description": "Redshift federated catalog",
        "CreateDatabaseDefaultPermissions" : [],
        "CreateTableDefaultPermissions": [],
        "FederatedCatalog": {
            "Identifier": "arn:aws:redshift:us-
east-1:123456789012:datashare:11524d7f-f56d-45fe-83f7-d7bb0a4d6d71/
ds_internal_namespace",
            "ConnectionName": "aws:redshift"
        },
        "CatalogProperties": {
          "DataLakeAccessProperties" : {
            "DataLakeAccess" : true,
            "DataTransferRole" :
 "arn:aws:iam::123456789012:role/DataTransferRole"
         }
       }
    }
}'
```

3. Grant permissions to users in your account or in external accounts.

AWS Management Console

1. Choose **Next** to grant permissions to other users on the shared catalogs, databases, and tables.

2. On the **Add permissions** screen, choose the principals and the types of permissions to grant.



a. In the **Principals** section, choose a principal type and then specify principals to grant permissions.

- **IAM users and roles** – Choose one or more users or roles from the IAM users and roles list.
- **SAML users and groups** – For SAML and Amazon Quick users and groups, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon Quick users or groups. Press **Enter** after each ARN.

  For information about how to construct the ARNs, see AWS CLI grant and revoke AWS CLI commands.
- **External accounts** – For AWS, AWS organization, or IAM Principal enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or ARN for the IAM user or role. Press Enter after each ID. An organization ID consists of "o-" followed by 10–32 lower-case letters or digits. An organizational unit ID starts with "ou-" followed by 4–32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

b. In the **Permissions** section, select permissions and grantable permissions.

Under **Catalog permissions**, select one or more permissions to grant. Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account. This option is not supported when you are granting permissions to an IAM principal from an external account.

Choose **Super user** to grant the user unrestricted permissions to the resources (databases, tables, views) within the catalog.

3. Choose **Add**.

AWS CLI

Use the following examples to grant catalog, database, and table permissions using the AWS CLI:

- The following example shows how to grant permissions on the federated catalog.

```
aws lakeformation grant-permissions
 --cli-input-cli-json \
   '{
        "Principal": {
```

```
              "DataLakePrincipalIdentifier": "arn:aws:iam::123456789012:role/
non-admin"
          },
          "Resource": {
               "Catalog": {
                     "Id": "123456789012:nscatalog"
                }
            },
            "Permissions": [
                   "DESCRIBE","CREATE_CATALOG"
             ],
            "PermissionsWithGrantOption": [
             ]
    }'
```

- Use the following example to grant permissions on a database.

```
aws lakeformation grant-permissions \
  --cli-input-json \
          '{
              "Principal": {

 "DataLakePrincipalIdentifier": "arn:aws:iam::123456789012:role/non-admin"
              },
              "Resource": {
                  "Database": {
                      "CatalogId": "123456789012:nscatalog/dev",
                      "Name": "public"
                  }
              },
              "Permissions": [
                  "ALL"
              ]
          }'
```

- The following example shows how to grant permissions on a table in the Amazon Redshift database.

```
aws lakeformation grant-permissions \
  --cli-input-json \
        '{
            "Principal": {
```

```
"DataLakePrincipalIdentifier": "arn:aws:iam::123456789012:role/non-admin"
            },
            "Resource": {
                "Table": {
                    "CatalogId": "123456789012:nscatalog2/dev",
                    "DatabaseName": "public",
                    "TableWildcard" : {}
                }
            },
            "Permissions": [
                "ALL"
            ]
        }'
```

4. Choose **Next** to review the catalog details and create a federated catalog. The newly created federated catalog and the catalog objects appear in the **Catalogs** page.

   An Amazon Redshift federated catalog is referenced with `catalogID = 123456789012:Redshift-federated catalog id`.

Viewing catalog objects

After you create the federated catalog, you can view the objects in the catalog using the Lake Formation console or AWS CLI.

AWS Management Console

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. Choose **Catalogs** under Data Catalog.
3. Choose a federated catalog from the list on the **Catalogs** page.
4. The catalog summary page shows the catalog objects (databases and tables) that you have permissions on. The **Permissions** tab shows the IAM principals who has been granted permissions on these objects.

AWS CLI

- The following AWS CLI example shows how to request the top-level catalog.

```
aws glue get-catalog \
--catalog-id 123456789012:nscatalog
```

Response

```
{
    "Catalog": {
        "CatalogId": "123456789012:nscatalog",
        "Name": "nscatalog",
        "ResourceArn": "arn:aws:glue:us-east-1:123456789012:catalog/nscatalog",
        "Description": "Redshift published Catalog",
        "CreateTime": "2024-09-05T14:49:16-07:00",
        "FederatedCatalog": {
            "Identifier": "arn:aws:redshift:us-
east-1:123456789012:datashare:b1234589-e823-4a14-ad8e-077085540a50/
ds_internal_namespace",
            "ConnectionName": "aws:redshift"
        },
        "CatalogProperties": {
            "DataLakeAccessProperties": {
                "DataLakeAccess": true,
                "DataTransferRole": "arn:aws:iam::123456789012:role/
DataTransferRole",
                "KmsKey": "AWS_OWNED_KMS_KEY",
                "ManagedWorkgroupName": "123456789012:nscatalog",
                "ManagedWorkgroupStatus": "AVAILABLE",
                "RedshiftDatabaseName": "dev"
            }
        },
        "CatalogIdentifier": "e2309c2c2fb048f1a3069dfdc1c7883e",
        "CreateTableDefaultPermissions": [],
        "CreateDatabaseDefaultPermissions": []
    }
}
```

- The following example shows how to request all catalogs in the account.

```
aws glue get-catalogs \
  --recursive
```

- The following example request shows how to get a Amazon Redshift database-level catalog.

```
aws glue get-catlog \
 --catalog-id 123456789012:namespace catalog name/redshift database name
```

- The following example request shows how to get the databases in the Amazon Redshift database-level catalog.

```
aws glue get-databases \
--catalog-id 123456789012:namespace catalog name/redshift database name
```

- The following example request shows how to get a Amazon Redshift table in the catalog.

```
aws glue get-table \
  --catalog-id 123456789012:parent catalog name/redshift database \
  --database-name redshift schema name \
  --name table name
```

- Following example shows how to get all tables from the Amazon Redshift database.

```
aws glue get-tables \
 --catalog-id 123456789012:namespace catalog name/redshift database name \
 --database-name RS schema name
```

Updating a federated catalog

You can update a Amazon Redshift federated catalog in the Data Catalog using the Lake Formation console, the AWS CLI or the UpdateCatalog API operation.

AWS Management Console

Follow these steps to update your federated catalog using Lake Formation console.

1. Sign in to the AWS Management Console, and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the left navigation pane, choose **Catalogs** under **Data Catalog**.
3. On the **Catalogs** page, choose the Amazon Redshift federated catalog you want to update.
4. Under **Actions**, choose **Edit**.

5. On the **Set Catalog details** screen, under the **Access from engines** section, choose **Access this catalog from Iceberg compatible engines**. Checking this option will enable data lake access for Apache Iceberg compatible query engines.
6. Next, create a new IAM role or choose an existing IAM role that has the policy that grants permissions to perform data transfer to and from the Amazon S3 bucket.

   For more information on permissions, see Prerequisites for managing Amazon Redshift namespaces in the AWS Glue Data Catalog.
7. By default, your data in the Amazon Redshift cluster is encrypted using an AWS managed key. If you choose to encrypt data using a customer managed key, either create a KMS key or choose an existing key that has the permissions defined in the Prerequisites for managing Amazon Redshift namespaces in the AWS Glue Data Catalog section.
8. Choose **Save**.

   Upon successful completion, the **Catalog details** page shows the managed workgroup name with the status as "Success".

AWS CLI

The following is an example of the `update-catalog` CLI input with the data lake access disabled by setting the `DataLakeAacess` parameter value as `false`.

```
aws glue update-catalog  --cli-input-json \
'{
    "Name": "nscatalog",
    "CatalogInput": {
        "Description": "Redshift published catalog",
        "CreateDatabaseDefaultPermissions" : [],
        "CreateTableDefaultPermissions": [],
        "FederatedCatalog": {
            "Identifier": "arn:aws:redshift:us-
east-1:123456789012:datashare:11524d7f-f56d-45fe-83f7-d7bb0a4d6d71/
ds_internal_namespace",
            "ConnectionName": "aws:redshift"
        },
        "CatalogProperties": {
          "DataLakeAccessProperties" : {
            "DataLakeAccess" : false
        }
       }
```

```
    }
}'
```

Accessing a shared federated catalog

AWS Lake Formation cross-account capabilities allow users to securely share distributed data lakes across multiple AWS accounts, AWS organizations, or directly with IAM principals in another account providing fine-grained access to the metadata and underlying data.

Lake Formation uses the AWS Resource Access Manager (AWS RAM) service to facilitate resource sharing. When you share a catalog resource with another account, AWS RAM sends an invitation to the grantee account to accept or reject the resource grant.

Integrated analytical services such as Amazon Athena and Redshift Spectrum require resource links to be able to include shared resources in queries. Principals need to create a resource link in their AWS Glue Data Catalog to a shared resource from another AWS account. For more information about resource links, see How resource links work in Lake Formation.

A *Catalog link container* is a Data Catalog object, which references a local or cross-account federated database-level catalog from other AWS accounts. You can also create database links and table links within a catalog link container. When you create a database link or a table link, you must specify a target resource that resides under the same target Amazon Redshift database-level catalog (Amazon Redshift database).

To create a catalog link container, you need the Lake Formation `CREATE_CATALOG` or the `glue:CreateCatalog` permission.

You must have **Cross account version settings** version 4 or higher for sharing databases or tables in the federated catalog across AWS accounts.

Creating a catalog link container to a cross-account federated catalog

You can create a catalog link container that points to a Redshift database-level federated catalog in any AWS Region by using the AWS Lake Formation console, AWS Glue `CreateCatalog` API, or AWS Command Line Interface (AWS CLI).

To create a catalog link container to a shared catalog (console)

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as a principal who has the Lake Formation `CREATE_CATALOG` permission.

2. In the navigation pane, choose **Catalogs**, and then choose **Create catalog**.
3. On the **Set catalog details** page, provide the following information:

Name

Enter a name that adheres to the same rules as a catalog name. The name can be the same as the target shared catalog.

Type

Choose **Catalog link container** as the type of catalog.

Source

Choose `Redshift`.

Target Redshift catalog

Select a Redshift database-level federated catalog or choose a local (owned) catalog from the list.

The list contains all the catalogs shared with your account. Note the catalog owner account ID is listed with each catalog. If you don't see a catalog that you know was shared with your account, check the following:

- If you aren't a data lake administrator, check that the data lake administrator granted you Lake Formation permissions on the catalog.
- If you are a data lake administrator, and your account is not in the same AWS organization as the granting account, ensure that you have accepted the AWS Resource Access Manager (AWS RAM) resource share invitation for the catalog. For more information, see Accepting a resource share invitation from AWS RAM.

> **Note**
>
> When creating a catalog link container through the console, the **Target Redshift Catalog** dropdown might display `No matches` when attempting to select a cross-account Redshift catalog. Despite this display, you can manually enter the target ARN of the Amazon Redshift federated database-level catalog (Amazon Redshift database) in the input field, and the form will still work correctly. For example: `arn:aws:glue:us-east-1:123456789012:catalog/federated-catalog-redshift/dev`.

This behavior occurs because the console can only search for potential candidates within the currently signed-in account. The dropdown is intended as an auto-complete feature, but you can still manually input ARNs for cross-account access.

4. To enable Apache Iceberg query engines to read and write to Amazon Redshift namespaces, AWS Glue creates a managed Amazon Redshift cluster with the compute and storage resources required to perform read and write operations without impacting Amazon Redshift data warehouse workloads. You need to provide an IAM role with the permissions required to transfer data to and from the Amazon S3 bucket.
5. Choose **Next**.
6. (Optional) Choose **Add permissions** to grant permissions to other principals.

   However, granting permissions on a catalog link container doesn't grant permissions on the target (linked) catalog. You must grant permissions on the target catalog separately for the catalog link to be visible in Athena.
7. Next, review the catalog link container details and choose **Create catalog**.

   You can then view the link container name under the **Catalogs** page.

   Now, you can create database links and table links in the catalog link container to enable access from query engines.

Create a catalog link container CLI example

- In the following example, the `TargetRedshiftCatalog` object specifies the arn of the Amazon Redshift federated database-level catalog (Amazon Redshift database). The `DataLakeAccess` must be enabled when you create the catalog link container.

```
aws glue create-catalog \
  --cli-input-json
    '{
        "Name": "linkcontainer",
        "CatalogInput": {
            "TargetRedshiftCatalog": {
               "CatalogArn": "arn:aws:us-east-1:123456789012:catalog/nscatalog/dev"
             },
            "CatalogProperties": {
              "DataLakeAccessProperties" : {
                "DataLakeAccess" : true,
```

```
                "DataTransferRole" : "arn:aws:iam::111122223333:role/
DataTransferRole"
              }
            }
          }
    }'
```

Creating resource links under the catalog link container

You can create resource links to databases and tables links under a catalog link container. When you create database resource links or table resource links, you must specify a target resource that resides under the same target Amazon Redshift database-level catalog (Amazon Redshift database) that the link container points to.

You can create a resource link to a shared Amazon Redshift database or a table by using the AWS Lake Formation console, API, or AWS Command Line Interface (AWS CLI).

- For detailed instructions, see Creating a resource link to a shared Data Catalog database.

  Following is a AWS CLI example to create a database resource link under a catalog link container.

```
aws glue create-database \
  --cli-input-json \
    '{
        "CatalogId": "111122223333:linkcontainer",
        "DatabaseInput": {
            "Name": "dblink",
             "TargetDatabase": {
               "CatalogId": "123456789012:nscatalog/dev",
                "DatabaseName": "schema1"
             }
        }
    }'
```

- To create a table resource link under a catalog link container, you need to first create a AWS Glue database in the local AWS Glue Data Catalog to contain the table resource link.

  For more information on creating resource links to shared tables, see Creating a resource link to a shared Data Catalog table.

- Create a database to contain the table resource link example

```
aws glue create-database \
  --cli-input-json \
      '{
          "CatalogId": "111122223333:linkcontainer",
          "DatabaseInput": {
              "Name": "db1",
              "Description": "creating parent database for table link"
          }
      }'
```

- Create table resource link example

```
aws glue create-table \
  --cli-input-json \
    '{
        "CatalogId": "111122223333:linkcontainer",
         "DatabaseName": "db1",
        "TableInput": {
            "Name": "tablelink",
            "TargetTable": {
                "CatalogId": "123456789012:nscatalog/dev",
               "DatabaseName": "schema1",
                "Name": "table1"
             }
        }
    }'
```

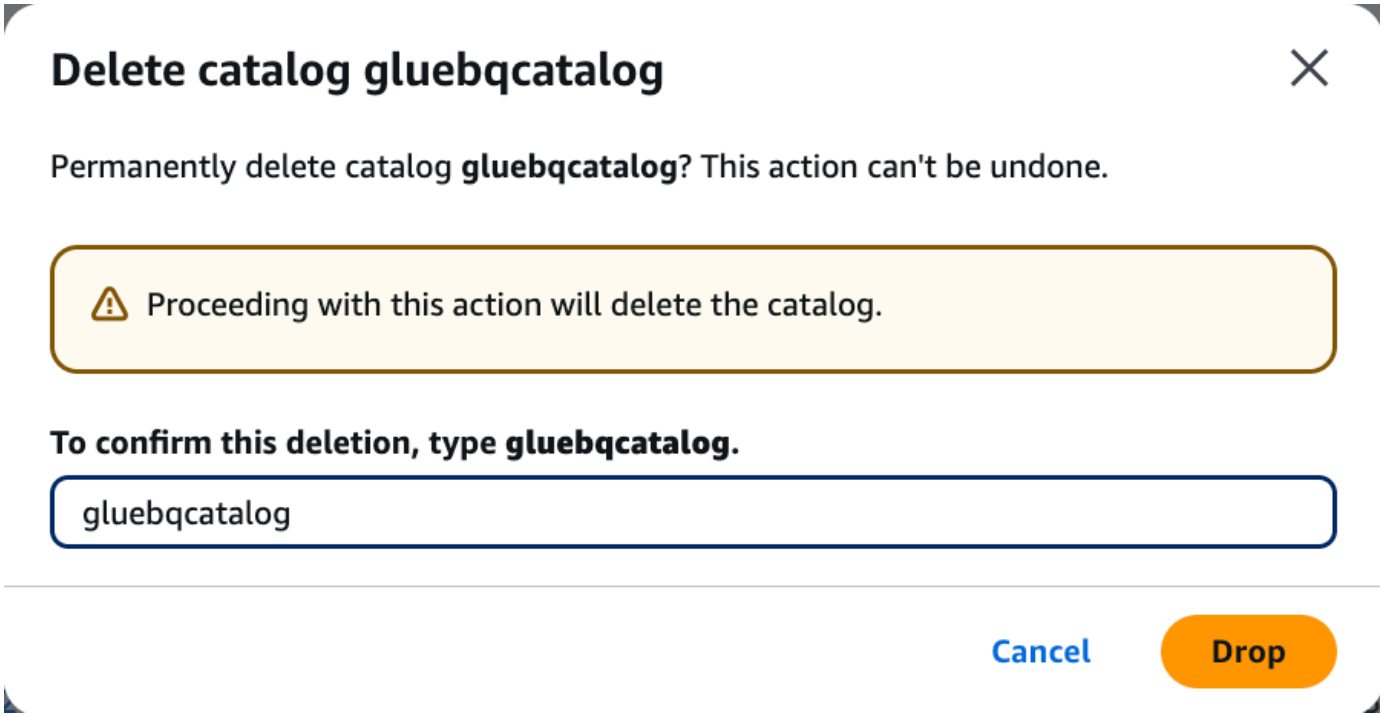
Deleting a federated catalog

You can delete the federated catalogs that you created in the AWS Glue Data Catalog using the `glue:DeleteCatalog` operation or the AWS Lake Formation console.

To delete a federated catalog (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, choose **Catalogs** under **Data Catalog**.

3. Choose the catalog that you want to delete from the catalogs list.
4. Choose **Delete** from **Actions**.
5. Choose **Drop** to confirm and the federated catalog will be deleted from the Data Catalog.



To delete a federated catalog (CLI)

- ```
  aws glue delete-catalog
   --catalog-id 123456789012:catalog name
  ```

Querying federated catalogs

After you grant permissions to other principals, they can sign in and start querying the tables in the federated catalogs by logging into the SQL tools using Amazon Redshift, Amazon EMR, Amazon Athena, and AWS Glue ETL.

For more information on connecting to the AWS Glue Data Catalog using Apache Iceberg Rest extension endpoint or standalone Spark application, see Accessing the AWS Glue Data Catalog section in the AWS Glue Developer Guide.
  ```

You can use the data definition language (DDL) queries to create and manage tables in the database using Apache Spark on Amazon EMR. To create and delete tables in the Amazon Redshift database, the principal must have Lake Formation `Create table`, `Drop` permissions.

For more information on granting Data Catalog permissions, see Granting permissions on Data Catalog resources.

For more information on querying the catalog resources from Amazon Athena, see Querying AWS Glue Data Catalog from Amazon Athena in Amazon Athena User Guide.

Additional resources

You can use Amazon SageMaker Lakehouse to achieve unified access to data in both data warehouses and data lakes. Through SageMaker Lakehouse, you can use preferred analytics, machine learning, and business intelligence engines through an open, Apache Iceberg REST API to help ensure secure access to data with consistent, fine-grained access controls.

- Amazon SageMaker workshop
- Simplify data access for your enterprise using Amazon SageMaker Lakehouse

Federating into external data sources in the AWS Glue Data Catalog

You can connect the AWS Glue Data Catalog (Data Catalog) to data warehouses such as Amazon Redshift, Snowflake, cloud databases such as Amazon RDS, Amazon DynamoDB, Oracle, and streaming services such as Amazon MSK, and on-premises systems such as Teradata using AWS Glue connections. These connections are stored in the AWS Glue Data Catalog and registered with AWS Lake Formation, allowing you to create a federated catalog for each available data source.

A *federated catalog* is a top level container that points to a database in an external data system. It enables you to query the data directly from the external data system without extract, transform, and load (ETL) process.

For more information about AWS Glue connections, see Connecting to data in the AWS Glue Developer Guide.

Data lake administrators can create federated catalogs using Amazon SageMaker Lakehouse or Amazon Athena.

Data lake administrators can then grant fine-grained permissions on the objects within the catalog using Lake Formation, controlling access at various levels such as catalog, database, table, column, row, or cell. Data analysts can discover and query the cataloged data sources using Athena, with Lake Formation enforcing the defined access policies. Analysts can join data across multiple sources in a single query without needing to connect to each source individually.

Topics

- Workflow
- Prerequisites for connecting the Data Catalog to external data sources
- Creating a federated catalog using an AWS Glue connection
- Viewing catalog objects
- Deleting a federated catalog
- Querying federated catalogs
- Additional resources

Workflow

A data lake administrator or a user with the required permissions completes the following the steps for connecting the AWS Glue Data Catalog to an external data source.

1. Creates an AWS Glue connection to the data source. When you register the connection, the IAM role used in registering the connection must have access to the Lambda function and the Amazon S3 spill bucket location.
2. Registers the connection with Lake Formation.
3. Creates a federated catalog in the Data Catalog using a AWS Glue connection to connect to the available data sources. The databases, tables, and views are automatically cataloged in the Data Catalog, and registered with Lake Formation.
4. Grants access to specific catalogs, databases, and tables to data analysts using Lake Formation permissions. Fine-grained access control policies can be defined across data lakes, warehouses, and OLTP sources using Lake Formation, enabling row-level and column-level security filters.

   Data analysts can then access all data through the Data Catalog using SQL queries in Athena, without needing separate connections or data source credentials. Analysts can run federated SQL queries that scan data from multiple sources, joining data in-place without complex data pipelines.

Prerequisites for connecting the Data Catalog to external data sources

To connect the AWS Glue Data Catalog to external data sources, register the connection with Lake Formation, and set up federated catalogs, you need to complete the following requirements:

> **Note**
>
> We recommend that a Lake Formation data lake administrator creates the AWS Glue connections to connect to external data sources, and create the federated catalogs.

1. **Create IAM roles.**

   - Create a role that has the necessary permissions to deploy resources (Lambda function, Amazon S3 spill bucket, IAM role, and the AWS Glue connection) required to create a connection to the external data source.
   - Create a role that has the necessary minimum permissions to access the AWS Glue connection properties (the Lambda function and the Amazon S3 spill bucket). This is the role that you'll include when you register the connection with Lake Formation.

     To use Lake Formation to manage and secure the data in your data lake, you must register the AWS Glue connection with Lake Formation. By doing so, Lake Formation can vend credentials to Amazon Athena for querying the federated data sources.

     The role must have `Select` or `Describe` permissions on the Amazon S3 bucket and the Lambda function.

     - s3:ListBucket
     - s3:GetObject
     - lambda:InvokeFunction

     JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1/object/*",
        "arn:aws:s3:::amzn-s3-demo-bucket1/object"
      ]
    },
    {
      "Sid": "lambdainvoke",
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:example-
lambda-function"
    },
    {
      "Sid": "gluepolicy",
      "Effect": "Allow",
      "Action": "glue:*",
      "Resource": "*"
    }
  ]
}
```

- Add the following trust policy to the IAM role that is used in registering the connection:

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "lakeformation.amazonaws.com"
              ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- The data lake administrator who registers the connection must have the `iam:PassRole` permission on the role.

  The following is an inline policy that grants this permission. Replace *<account-id>* with a valid AWS account number, and replace *<role-name>* with the name of the role.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PassRolePermissions",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/example-role-name>"
            ]
        }
    ]
}
```

- To create federated catalogs in Data Catalog, make sure the IAM role you're using is a Lake Formation data lake administrator by checking the data lake settings (`aws lakeformation get-data-lake-settings`).

  If you're not a data lake administrator, you need the Lake Formation CREATE_CATALOG permission to create a catalog. The following example shows how to grant the required permissions to create catalogs.

```
aws lakeformation grant-permissions \
--cli-input-json \
        '{
            "Principal": {
             "DataLakePrincipalIdentifier":"arn:aws:iam::123456789012:role/non-
admin"
            },
            "Resource": {
                "Catalog": {
                }
```

```
            },
            "Permissions": [
                "CREATE_CATALOG",
                "DESCRIBE"
            ]
        }'
```

In addition to the CREATE_CATALOG permission, you need DATA_LOCATION_ACCESS on the registered connection to create a federated catalog. A data lake administrator has implicit data location permissions and does not need this grant. For more information, see Implicit Lake Formation permissions.

The following example shows how to grant DATA_LOCATION_ACCESS on a registered connection.

```
aws lakeformation grant-permissions \
  --cli-input-json \
    '{
      "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::123456789012:role/non-admin-
role"
      },
      "Resource": {
        "DataLocation": {
          "CatalogId": "123456789012",
          "ResourceArn": "arn:aws:glue:us-east-1:123456789012:connection/
connection-name"
        }
      },
      "Permissions": ["DATA_LOCATION_ACCESS"]
    }'
```

2. Add the following key policy to the AWS KMS key if you're using a customer managed key to encrypt the data in the data source. Replace the account number with a valid AWS account number, and specify role name. By default, the data is encrypted using an KMS key. Lake Formation provides an option to create your custom KMS key for encryption. If you're using a customer managed key, you must add specific key policies to the key.

For more information about managing the permissions of a customer managed key, see Customer managed keys.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-1"
        }
    ]
}
```

Creating a federated catalog using an AWS Glue connection

To connect the AWS Glue Data Catalog to external data sources, you need to use AWS Glue connections that enable communication with the external data sources. You can create AWS Glue connections using the AWS Glue console, Create connection API, and Amazon SageMaker Lakehouse console.

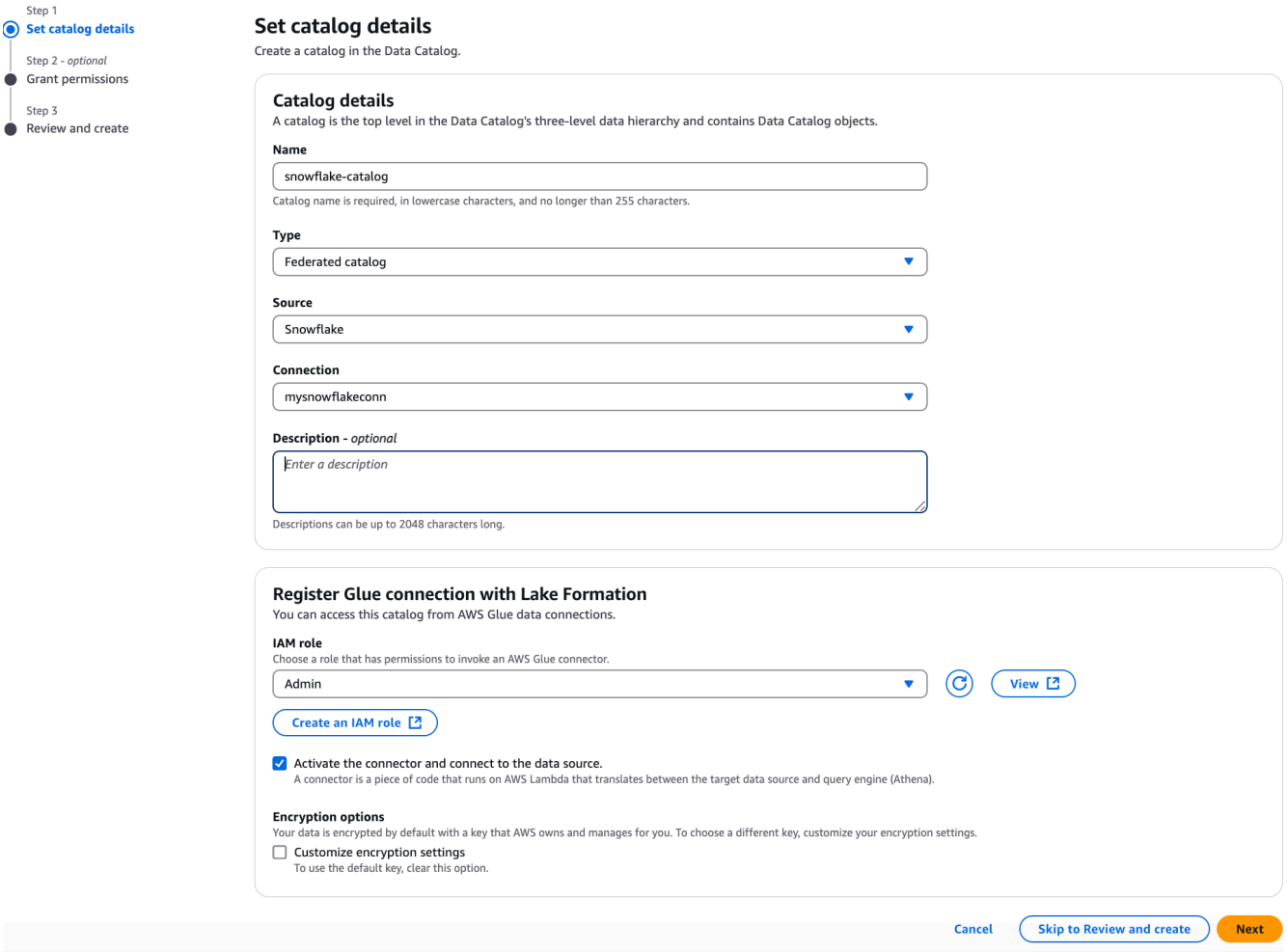
For step by step instructions for creating an AWS Glue connection, see Connecting to data in the AWS Glue Developer Guide or Creating connections in Amazon SageMaker Lakehouse.

When a user runs a query on federated tables, Lake Formation vends credentials that invoke an AWS Lambda function specified in the AWS Glue connection to retrieve metadata objects from the data source.

AWS Management Console

To create a federated catalog from an external data source and set up permissions (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, choose **Catalogs** under **Data Catalog**.
3. Select the option **Create catalog**.
4. On the **Set Catalog** details page, enter the following information:



- **Name** – A unique name for your federated catalog. The name can't be changed, and must be in lower case. The name can consist of a maximum of 255 characters maximum. account.
- **Type** – Choose federated catalog as the catalog type.
- **Source** – Choose a data source from the dropdown. The data sources for which you've created connections are displayed. For more information about creating an AWS Glue

connection to an external data source, see Creating connections for connectors in the AWS Glue Developer Guide or Creating connections in Amazon SageMaker Lakehouse.

- **Connection** – Choose an existing AWS Glue connection to the data source.
- **Description** – Enter a description for the catalog created from the data source.

5. Choose an **IAM role** for Lake Formation to assume to vend credentials for the querying engine to access data from the data source. This role must have the required permissions to access the AWS Glue connection and invoke the Lambda function to access data from the external data source.

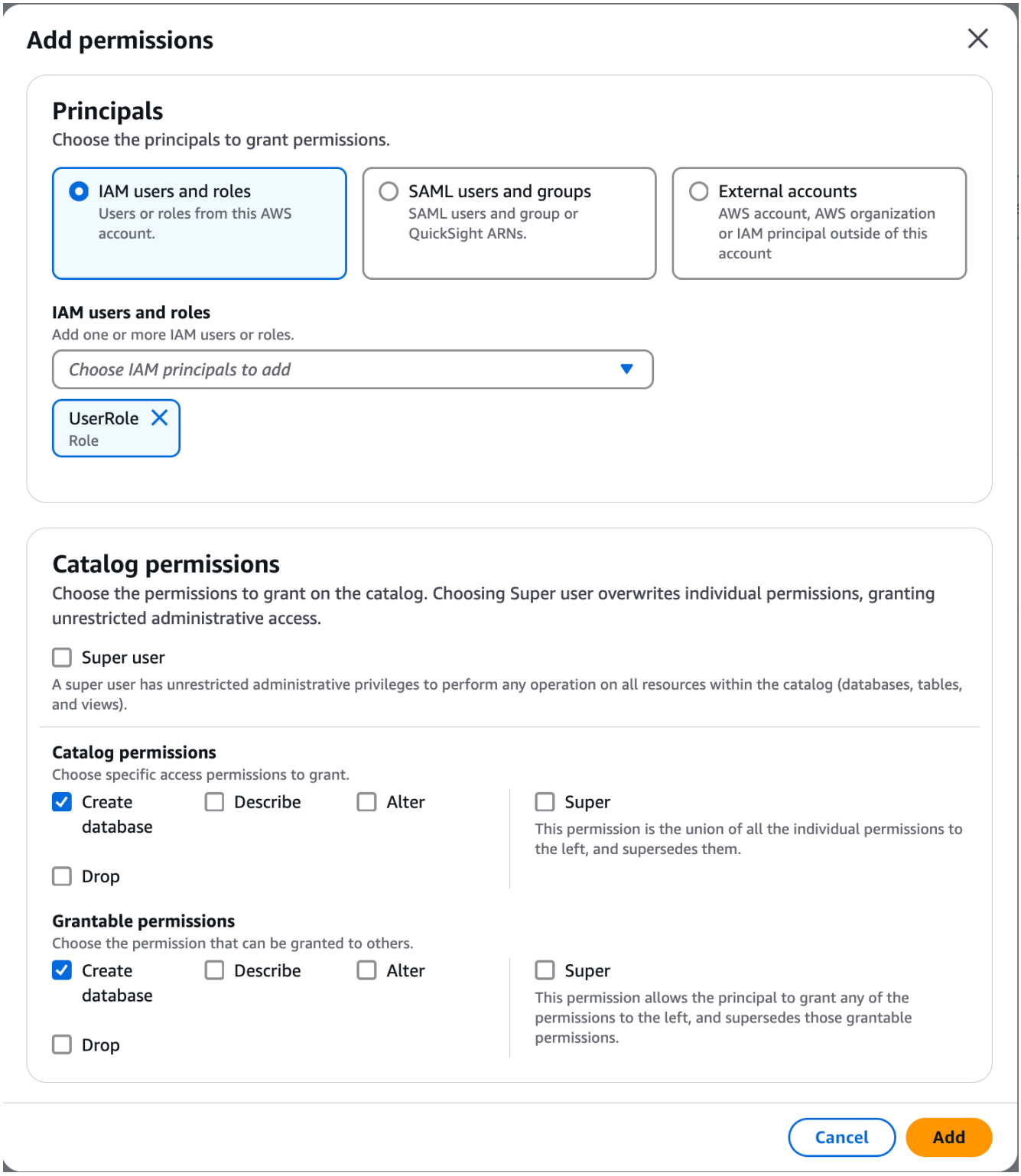
   You can also **Create a new role** in the IAM console.

   See the Prerequisites for connecting the Data Catalog to external data sources section for the required permissions.

6. Select the option **Activate the connector to connect to the data source** to enable Athena to run federated queries.

   For the supported list of connectors, see Register your connection in the Amazon Athena User Guide.

7. **Encryption options** – Choose **Customize encryption settings** option if you want to use a custom key to encrypt the catalog. To use a custom key, you must add additional custom managed key policy to your KMS key.
8. Choose **Next** to grant permissions to other principals.
9. On the **Grant permissions** page, choose **Add permissions**.
10. On the **Add permissions** screen, choose the principals and the types of permissions to grant.



- In the **Principals** section, choose a principal type and then specify principals to grant permissions.

- **IAM users and roles** – Choose one or more users or roles from the IAM users and roles list.
- **SAML users and groups** – For SAML and Amazon Quick users and groups, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon Quick users or groups. Press **Enter** after each ARN.

- In the **Permissions** section, select permissions and grantable permissions.

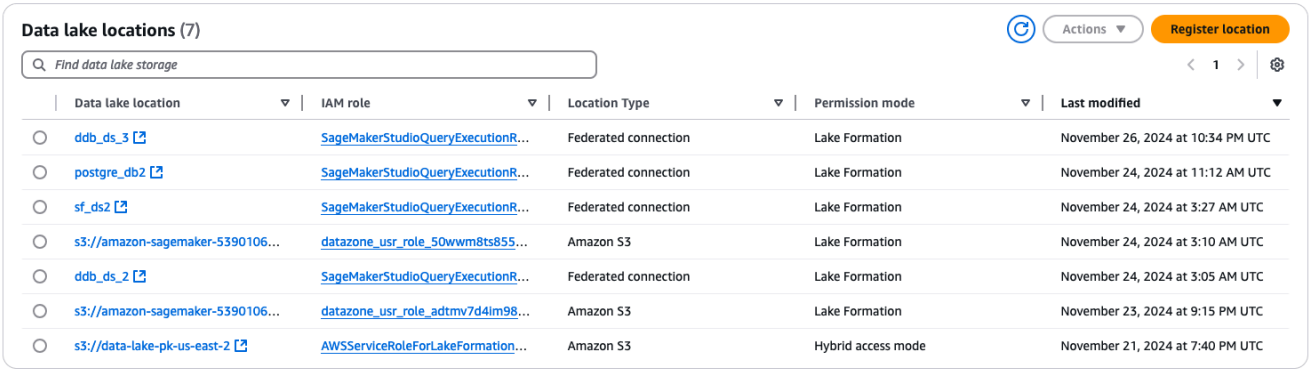
  Under **Catalog permissions**, select one or more permissions to grant.

  Choose **Super user** to grant unrestricted administrative permissions on all resources within the catalog.

  Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account. This option is not supported when you are granting permissions to an IAM principal from an external account.

11. Choose **Next** to review the information and create the catalog. The **Catalogs** list shows the new federated catalog.

    The **Data locations** list shows the newly registered federated connection.



AWS CLI

To create a federated catalog from an external data source and set up permissions

1. The following example shows how to create an AWS Glue connection.

```
aws glue create-connection
  --connection-input \
      '{
```

```
        "Name": "DynamoDB connection",
        "ConnectionType": "DYNAMODB",
        "Description": "A connection created for DynamoDB",
        "ConnectionProperties": {},
        "AthenaProperties": "spill_prefix": "your_spill_prefix",
        "lambda_function_arn": "Lambda_function_arn",
        "spill_bucket": "Your_Bucket_name",
        "AuthenticationConfiguration": {}
    }'
```

2. The following example shows how to register an AWS Glue connection with Lake Formation.

```
aws lakeformation register-resource
  --cli-input-json \
    {"ResourceArn":"arn:aws:glue:us-east-1:123456789012:connection/
dynamo","RoleArn":"arn:aws:iam::123456789012:role/
AdminTelemetry","WithFederation":true}
```

3. If you are not a data lake administrator, grant `DATA_LOCATION_ACCESS` on the registered connection to the principal that creates the federated catalog. A data lake administrator has implicit data location permissions and does not need this grant. For more information, see Implicit Lake Formation permissions.

   The following example shows how to grant `DATA_LOCATION_ACCESS` on a registered connection.

```
aws lakeformation grant-permissions \
  --cli-input-json \
    '{
      "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::123456789012:role/non-
admin-role"
      },
      "Resource": {
        "DataLocation": {
          "CatalogId": "123456789012",
          "ResourceArn": "arn:aws:glue:us-east-1:123456789012:connection/
connection-name"
        }
```

```
      },
      "Permissions": ["DATA_LOCATION_ACCESS"]
    }'
```

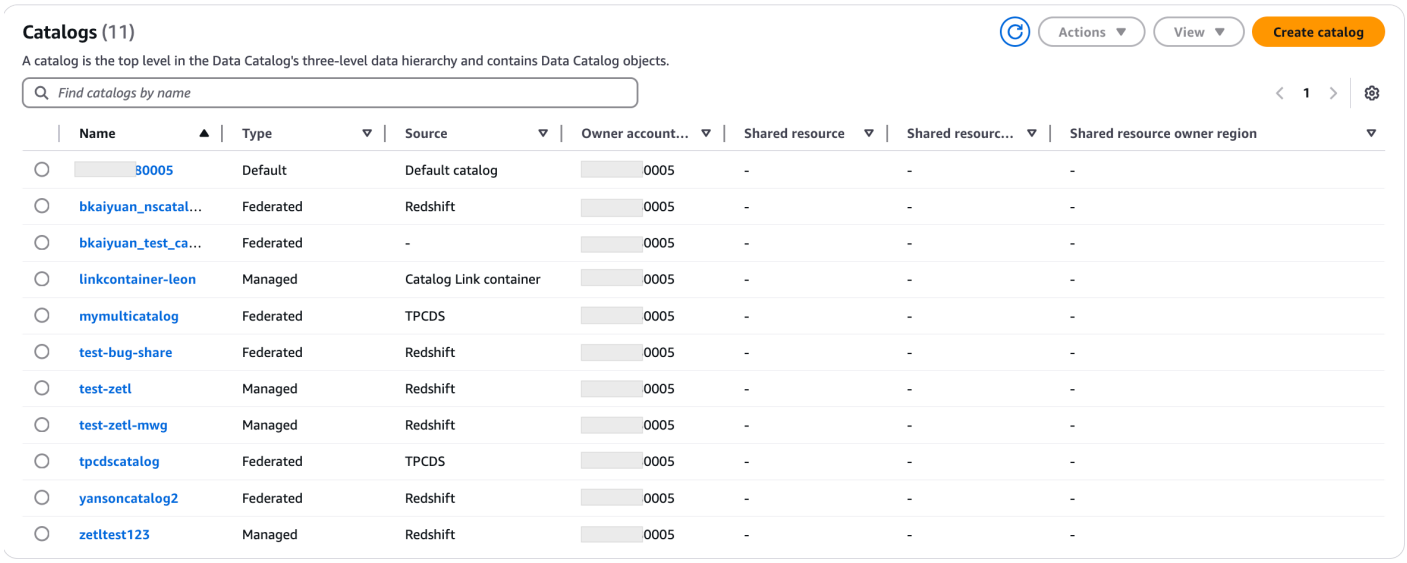
4. The following example shows how to create a federated catalog.

```
aws glue create-catalog
 --cli-input-json \
      '{
       "Name":"ddbcatalog",
       "CatalogInput":{"CatalogProperties":{"DataLakeAccessProperties":
{"DataTransferRole":"arn:aws:iam::123456789012:role/role name"}},
       "CreateDatabaseDefaultPermissions":[],
       "CreateTableDefaultPermissions":[],
       "FederatedCatalog":{"ConnectionName":"dynamo","Identifier":"dynamo"}
         }
       }'
```

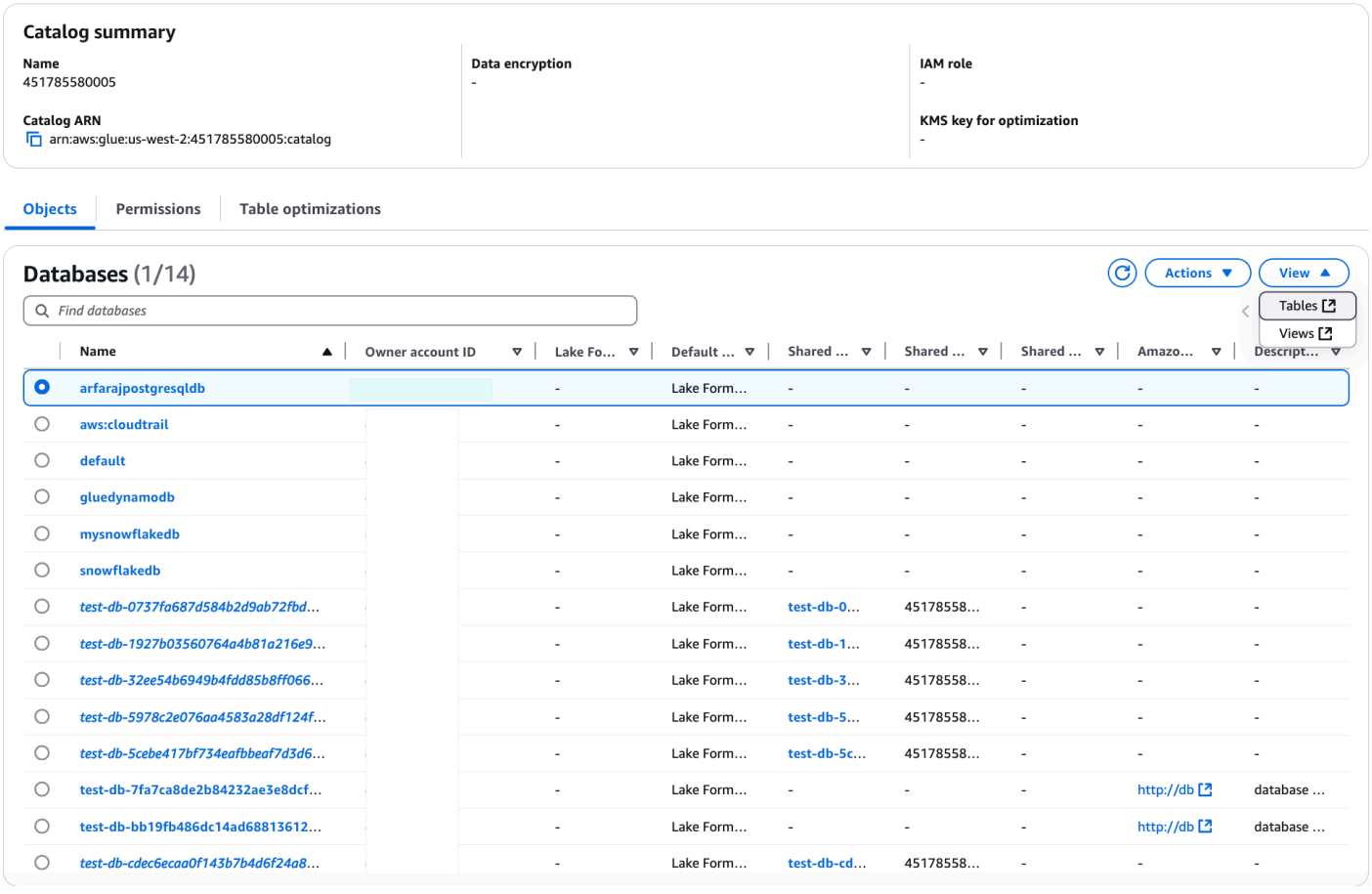
Viewing catalog objects

For each available data source, AWS Glue creates a corresponding catalog in the AWS Glue Data Catalog. After you create a catalog, you can view the databases and tables in the catalog using the Lake Formation console or AWS CLI. For

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. Choose **Catalogs** under Data Catalog. The catalogs page shows the catalogs that you've permissions on.



3. Choose a catalog from the list to view the databases and tables contained in the catalog. The list contains the databases in your account and resource links, which are links to shared databases and tables in external accounts, and are used for cross-account access to data in the data lake.



4. Choose **Tables** option under **View** to view and manage the tables in the database.

AWS CLI examples for viewing catalogs and databases

The following example shows how to view a catalog using AWS CLI

```
aws glue get-catalog \
--catalog-id 123456789012:dynamodbcatalog
```

The following example shows how to request all catalogs in the account.

```
aws glue get-catalogs \
 --recursive
```

The following example request shows how to get the databases in the catalog.

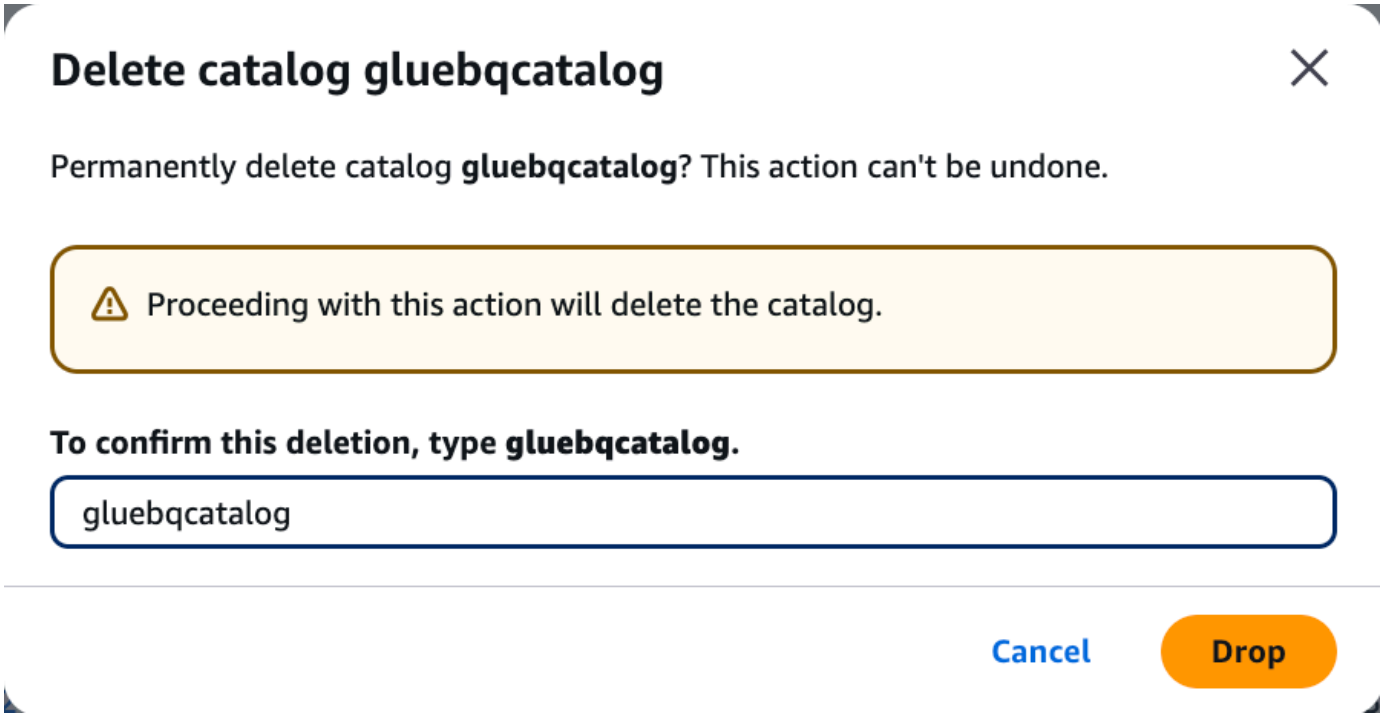
```
aws glue get-database \
--catalog-id 123456789012:dynamodbcatalog
--database-name database name
```

Deleting a federated catalog

You can delete the federated catalogs that you created in the AWS Glue Data Catalog using the `glue:DeleteCatalog` operation or the AWS Lake Formation console.

To delete a federated catalog (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, choose **Catalogs** under **Data Catalog**.
3. Choose the catalog that you want to delete from the catalogs list.
4. Choose **Delete** from **Actions**.
5. Choose **Drop** to confirm and the federated catalog will be deleted from the Data Catalog.



To delete a federated catalog (CLI)

- ```
  aws glue delete-catalog
    --catalog-id 123456789012:catalog name
  ```

Querying federated catalogs

After you grant permissions to other principals, they can sign in and start querying the tables in the federated catalogs using Athena.

To create and delete tables in the federated database, the principal must have Lake Formation `Create table`, `Drop` permissions.

For more information on granting Data Catalog permissions, see Granting permissions on Data Catalog resources.

For more information on querying the Data Catalog from Amazon Athena, see Querying AWS Glue Data Catalog from Amazon Athena in Amazon Athena User Guide.
  ```

Additional resources

In this blog post, we show how data analysts can now securely access and query data stored outside S3 data lakes, including Amazon Redshift data warehouses and Amazon DynamoDB databases, all through a single, unified experience. Administrators can now apply access controls at different levels of granularity to ensure sensitive data remains protected while expanding data access. This allows organizations to accelerate data initiatives while maintaining security and compliance, leading to faster, data-driven decision-making.

- Catalog and govern Amazon Athena federated queries with Amazon SageMaker Lakehouse

Amazon S3 Tables integration with AWS Glue Data Catalog and AWS Lake Formation

Amazon S3 Tables provide S3 storage that's specifically optimized for analytics workloads, improving query performance while reducing costs. The data in S3 Tables is stored in a new bucket type: a *table bucket*, which stores tables as subresources. S3 tables have built-in support for Apache Iceberg standard, which allows you to easily query tabular data in Amazon S3 table buckets using popular query engines like Apache Spark.

You can integrate Amazon S3 Tables with AWS Glue Data Catalog using either IAM access controls or with IAM and Lake Formation grants:

- **IAM access control**: Uses IAM policies to control access to S3 Tables and Data Catalog. In this access control approach, you need IAM permissions on both S3 Tables resources and Data Catalog objects to access resources.
- **Lake Formation access control**: Uses AWS Lake Formation grants in addition to AWS Glue IAM permissions to control access to S3 Tables through the Data Catalog. In this mode, principals require IAM permissions to interact with the Data Catalog, and Lake Formation grants determine which catalog resources (databases, tables, columns, rows) the principal can access. This mode supports both coarse-grained access control (database-level and table-level grants) and fine-grained access control (column-level and row-level security). When a registered role is configured and credential vending is enabled, S3 Tables IAM permissions are not required for the principal, as Lake Formation vends credentials on behalf of the principal using the registered role. Lake Formation access control also supports credential vending for third-party analytics engines.

This section provides guidance to configure the integration with AWS Lake Formation for the following scenarios:

- **Scenario A**: You integrated S3 Tables and Data Catalog using IAM access controls and now plan to use AWS Lake Formation. See Changing access controls for S3 Tables integration to learn more.
- **Scenario B**: You plan to integrate S3 Tables and Data Catalog using AWS Lake Formation and do not have them integrated in your account and Region today. Start with the Prerequisites for integrating Amazon S3 tables catalog with the Data Catalog and Lake Formation section and follow Enabling Amazon S3 Tables integration.
- **Scenario C**: You integrated S3 Tables and Data Catalog using AWS Lake Formation and now plan to use IAM. See Changing access controls for S3 Tables integration to learn more.

Make sure that you follow the steps in Integrating S3 Tables with AWS analytics services so that you have the appropriate permissions to access the AWS Glue Data Catalog and your table resources, and to work with AWS analytics services.

Topics

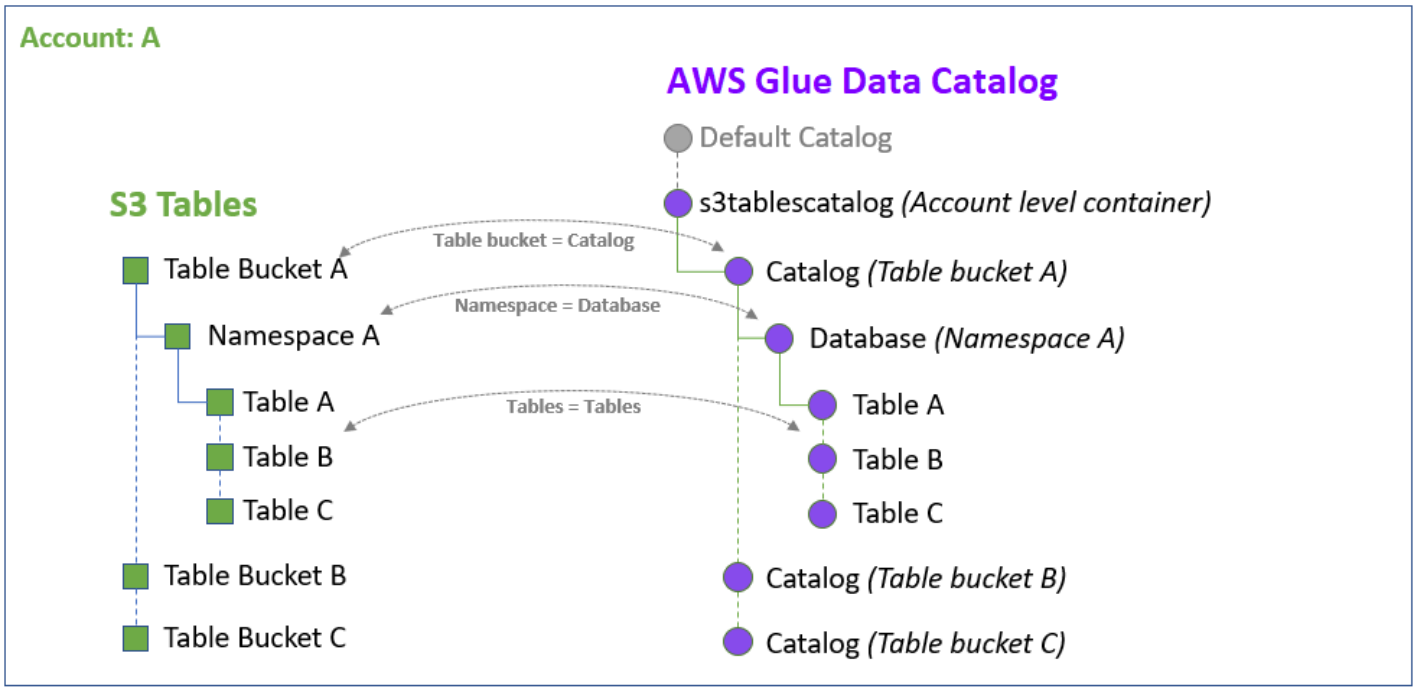
- How Data Catalog and Lake Formation integration works
- Prerequisites for integrating Amazon S3 tables catalog with the Data Catalog and Lake Formation
- Enabling Amazon S3 Tables integration
- Creating databases and tables in the S3 tables catalog
- Registering an Amazon S3 table bucket in another AWS account
- Granting permissions

How Data Catalog and Lake Formation integration works

When you integrate the S3 tables catalog with the Data Catalog and Lake Formation, the AWS Glue service creates a single federated catalog called `s3tablescatalog` in your account's default Data Catalog specific to your AWS Region. The integration maps all Amazon S3 table bucket resources in your account and AWS Region under the federated catalog in the following manner:

- Amazon S3 table buckets become a multi-level catalog in the Data Catalog.
- The associated Amazon S3 namespace is registered as a database in the Data Catalog.

- The Amazon S3 tables in the table bucket becomes tables in the Data Catalog.



After integrating with Lake Formation, you can create Apache Iceberg tables in the table buckets catalog, and access them via integrated AWS analytics engines such as Amazon Athena, Amazon EMR as well as third-party analytics engines.

When you also enable Lake Formation with integration, it enables fine-grained access control through AWS Lake Formation. This security approach means that, in addition to AWS Identity and Access Management (IAM) permissions, you must grant your IAM principal with Lake Formation permissions on your tables before you can work with them.

There are two main types of permissions in AWS Lake Formation:

- Metadata access permissions control the ability to create, read, update, and delete metadata databases and tables in the Data Catalog.
- Underlying data access permissions control the ability to read and write data to the underlying Amazon S3 locations that the Data Catalog resources point to.

Lake Formation uses a combination of its own permissions model and the IAM permissions model to control access to Data Catalog resources and underlying data:

- For a request to access Data Catalog resources or underlying data to succeed, the request must pass permission checks by both IAM and Lake Formation.
- IAM permissions control access to the Lake Formation and AWS Glue APIs and resources, whereas Lake Formation permissions control access to the Data Catalog resources, Amazon S3 locations, and the underlying data.

Lake Formation permissions apply only in the Region in which they were granted, and a principal must be authorized by a data lake administrator or another principal with the necessary permissions in order to be granted Lake Formation permissions.

Prerequisites for integrating Amazon S3 tables catalog with the Data Catalog and Lake Formation

Following are the prerequisites to enable Amazon S3 table integration with AWS Glue Data Catalog and AWS Lake Formation.

1. When you enable the Amazon S3 tables integration, Lake Formation automatically registers the S3 tables' location. To register the table bucket location with Lake Formation, you need an IAM role/user with `lakeformation:RegisterResource`, `lakeformation:RegisterResourceWithPrivilegedAccess`, and `lakeformation:CreateCatalog` permissions. When a non-administrator user with these permissions registers a catalog location, Lake Formation automatically grants them the `DATA_LOCATION_ACCESS` permission for that location allowing the calling principal the permissions to perform all supported Lake Formation operations on the registered data location.
2. When you enable the S3 tables integration, you need to choose an IAM role for Lake Formation to vend credentials to allow data access. Create an IAM role for Lake Formation data access to your S3 table buckets. The IAM role used when registering the table bucket with Lake Formation requires the following permissions:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "LakeFormationPermissionsForS3ListTableBucket",
            "Effect": "Allow",
            "Action": [
                "s3tables:ListTableBuckets"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "LakeFormationDataAccessPermissionsForS3TableBucket",
            "Effect": "Allow",
            "Action": [
                "s3tables:CreateTableBucket",
                "s3tables:GetTableBucket",
                "s3tables:CreateNamespace",
                "s3tables:GetNamespace",
                "s3tables:ListNamespaces",
                "s3tables:DeleteNamespace",
                "s3tables:DeleteTableBucket",
                "s3tables:CreateTable",
                "s3tables:DeleteTable",
                "s3tables:GetTable",
                "s3tables:ListTables",
                "s3tables:RenameTable",
                "s3tables:UpdateTableMetadataLocation",
                "s3tables:GetTableMetadataLocation",
                "s3tables:GetTableData",
                "s3tables:PutTableData"
            ],
            "Resource": [
                "arn:aws:s3tables:us-east-1:123456789012:bucket/*"
            ]
        }
    ]
}
```

For more information, see Requirements for roles used to register locations.

3. Add the following trust policy to the IAM role to allow the Lake Formation service to assume the role and vend temporary credentials to the integrated analytical engines.

```
{
```

```
  "Effect": "Allow",
  "Principal": {
    "Service": "lakeformation.amazonaws.com"
  },
  "Action": [
    "sts:AssumeRole",
    "sts:SetSourceIdentity",
    "sts:SetContext"  # add action to trust relationship when using IAM Identity
 center principals with Lake Formation
  ]
}
```

Note

The AWS analytics services integration process has been updated. If you've set up the integration with the preview release, you can continue to use your current integration. However, the updated integration process provides performance improvements. To update the integration:

1. First, delete your existing S3 tables catalog in Lake Formation. To delete the catalog, select the `s3tablescatalog` catalog from the catalogs list, and choose **Delete** from **Actions**.
2. Next, deregister the data location for the `s3tablescatalog`.
   a. On the Lake Formation console, under the **Administrations** section, choose **Data Locations**.
   b. Select a location, and from the **Actions** menu, choose **Remove**.
   c. When prompted for confirmation, choose **Remove**.

   For detailed instructions on deregistering a data location, see the Deregistering an Amazon S3 location section.
3. Then, follow the updated integration steps in the Enabling Amazon S3 Tables integration section.

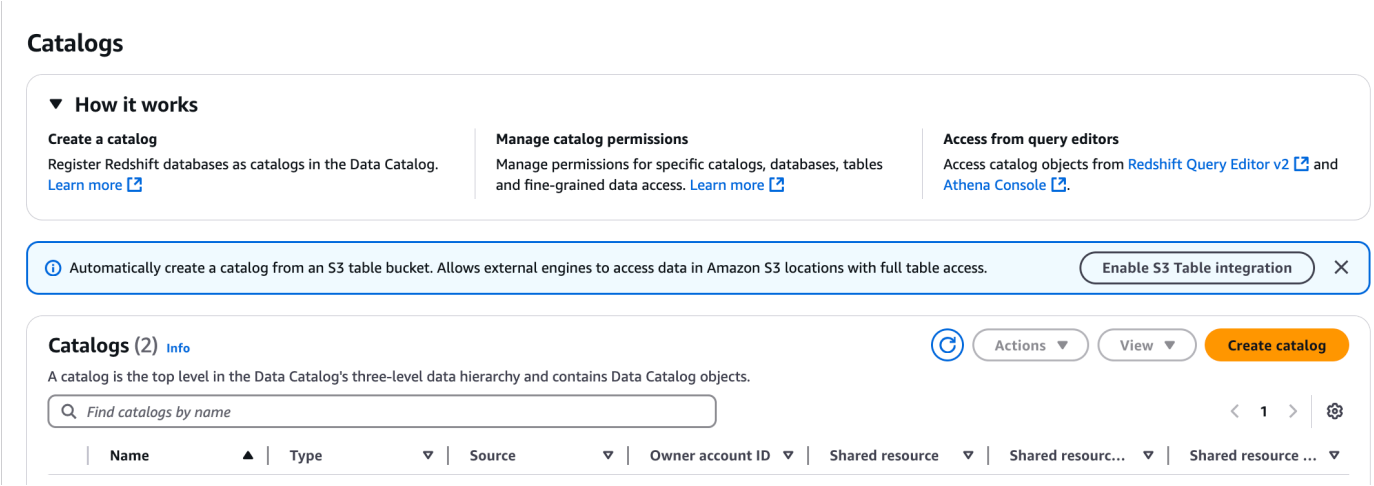
Enabling Amazon S3 Tables integration

You can create Amazon S3 table buckets using Amazon S3 console, and integrate it with AWS analytics services. For more information, see Using Amazon S3 Tables with AWS analytics services.

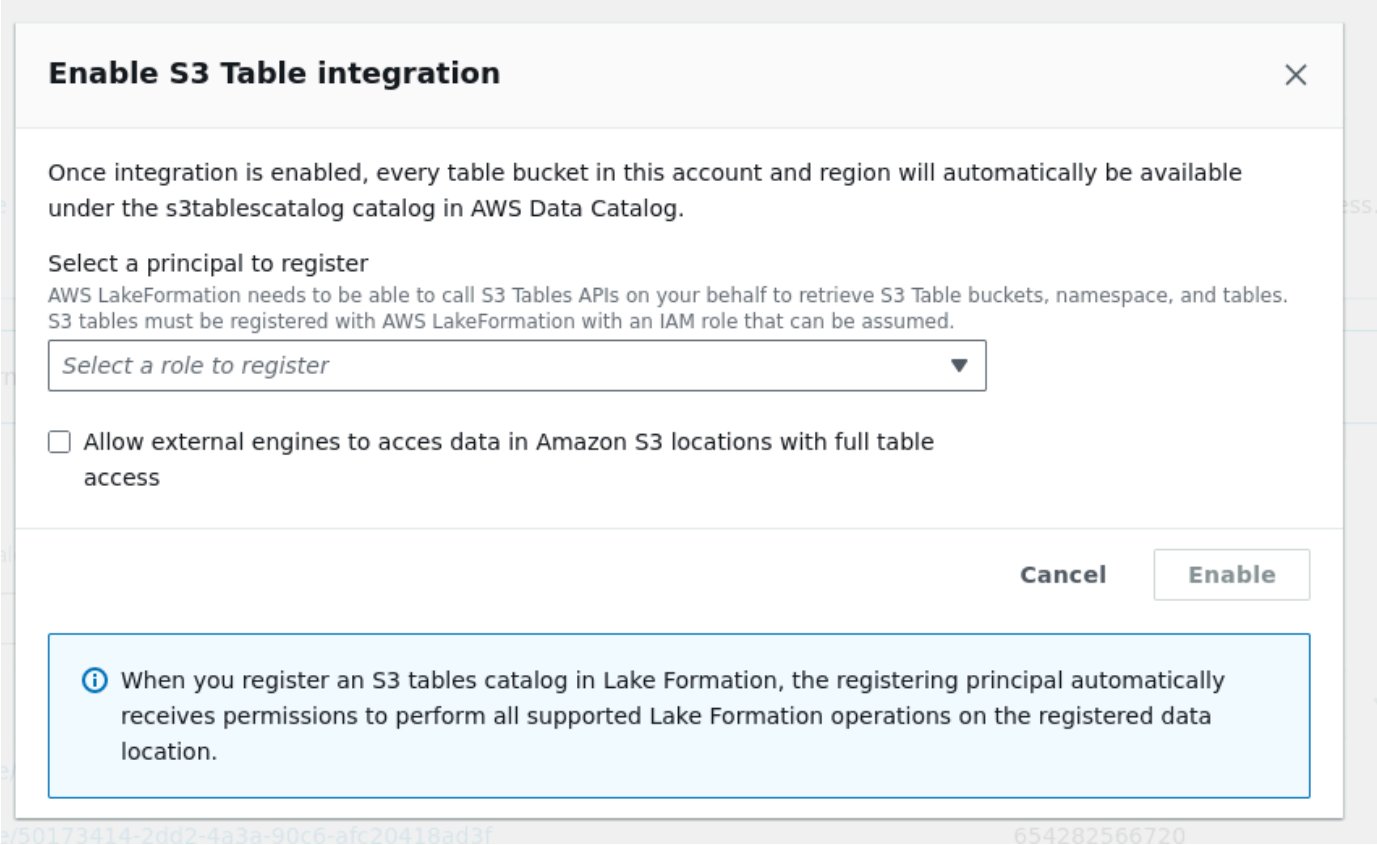
In AWS Lake Formation, you can enable Amazon S3 Tables integration with AWS Glue Data Catalog and AWS Lake Formation using the Lake Formation console or use AWS CLI.

To integrate Amazon S3 Tables with the Data Catalog and Lake Formation (console)

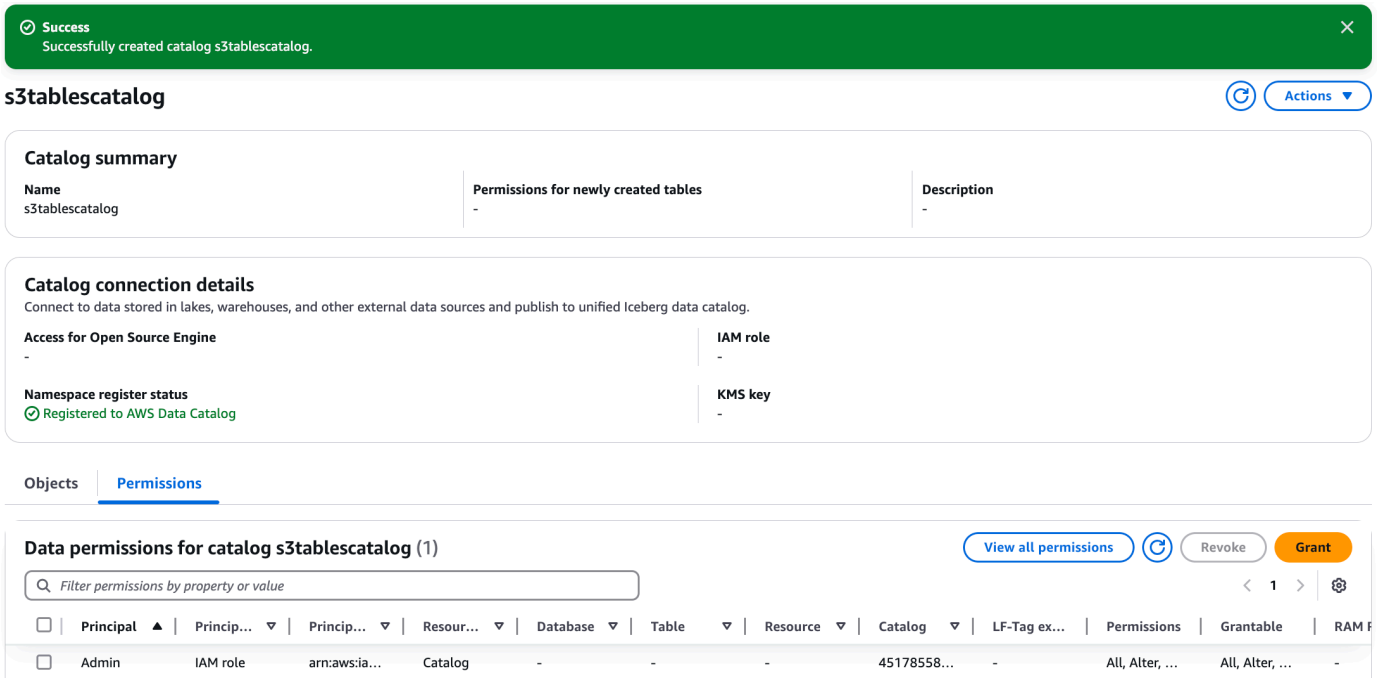
1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, choose **Catalogs** under **Data Catalog**.
3. Choose **Enable S3 Table integration** on the **Catalogs** page.



4. Choose an IAM role with the required permissions for Lake Formation to assume to vend credentials to the analytical query engines. For the permissions required for the role to accessing data, see step3-permissions in the prerequisites section.



5. Select **Allow external engines to access data in Amazon S3 locations with full table access** option. When you enable full table access for third-party engines, Lake Formation returns credentials to the third-party engine directly without performing IAM session tag validation. This means you cannot apply Lake Formation fine-grained access controls to the tables being accessed.
6. Choose **Enable**. The new catalog for S3 Tables is added to the catalog list. When you enable the S3 tables catalog integration, the service registers the data location of the S3 table bucket with Lake Formation.
7. Choose the catalog to view catalog objects and grant permissions to other principals.



To create multi-level catalogs, see the Creating a table bucket section in the Amazon Simple Storage Service User Guide.

To integrate Amazon S3 tables with the Data Catalog and Lake Formation (CLI)

Following the prerequisites section, create an IAM service role that allows Lake Formation to access your table resources.

1. Create a file called `Role-Trust-Policy.json` that contains the following trust policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "LakeFormationDataAccessPolicy",
        "Effect": "Allow",
        "Principal": {
          "Service": "lakeformation.amazonaws.com"
        },
        "Action": [
            "sts:AssumeRole",
            "sts:SetContext",
            "sts:SetSourceIdentity"
        ],
```

```
        "Condition": {
          "StringEquals": {
            "aws:SourceAccount": "111122223333"
          }
        }
      }
    ]
}
```

2. Create the IAM service role by using the following command:

```
aws iam create-role \
  --role-name S3TablesRoleForLakeFormation \
  --assume-role-policy-document file://Role-Trust-Policy.json
```

3. Create a file called `LF-GluePolicy.json` that contains the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LakeFormationPermissionsForS3ListTableBucket",
            "Effect": "Allow",
            "Action": [
                "s3tables:ListTableBuckets"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "LakeFormationDataAccessPermissionsForS3TableBucket",
            "Effect": "Allow",
            "Action": [
                "s3tables:CreateTableBucket",
                "s3tables:GetTableBucket",
                "s3tables:CreateNamespace",
                "s3tables:GetNamespace",
                "s3tables:ListNamespaces",
                "s3tables:DeleteNamespace",
                "s3tables:DeleteTableBucket",
                "s3tables:CreateTable",
                "s3tables:DeleteTable",
```

```
                "s3tables:GetTable",
                "s3tables:ListTables",
                "s3tables:RenameTable",
                "s3tables:UpdateTableMetadataLocation",
                "s3tables:GetTableMetadataLocation",
                "s3tables:GetTableData",
                "s3tables:PutTableData"
            ],
            "Resource": [
                "arn:aws:s3tables:us-east-1:111122223333:bucket/*"
            ]
        }
    ]
}
```

4. Attach the policy to the role by using the following command:

```
aws iam put-role-policy \
  --role-name S3TablesRoleForLakeFormation \
  --policy-name LakeFormationDataAccessPermissionsForS3TableBucket \
  --policy-document file://LF-GluePolicy.json
```

5. Create a file called `input.json` that contains the following:

```
{
    "ResourceArn": "arn:aws:s3tables:us-east-1:111122223333:bucket/*",
    "WithFederation": true,
    "RoleArn": "arn:aws:iam::111122223333:role/S3TablesRoleForLakeFormation"
}
```

6. Register table buckets with Lake Formation by using the following command:

```
aws lakeformation register-resource \
  --region us-east-1 \
  --with-privileged-access \
  --cli-input-json file://input.json
```

7. Create a file called `catalog.json` that contains the following catalog:

```
{
   "Name": "s3tablescatalog",
   "CatalogInput": {
      "FederatedCatalog": {
```

```
            "Identifier": "arn:aws:s3tables:us-east-1:111122223333:bucket/*",
            "ConnectionName": "aws:s3tables"
        },
        "CreateDatabaseDefaultPermissions": [],
        "CreateTableDefaultPermissions": [],
        "AllowFullTableExternalDataAccess": "True"
    }
}
```

8. Create the `s3tablescatalog` catalog by using the following command. Creating this catalog populates the AWS Glue Data Catalog with objects corresponding to table buckets, namespaces, and tables.

```
aws glue create-catalog \
  --region us-east-1 \
  --cli-input-json file://catalog.json
```

9. Verify that the `s3tablescatalog` catalog was added in AWS Glue by using the following command:

```
aws glue get-catalog --catalog-id s3tablescatalog
```

Creating databases and tables in the S3 tables catalog

You can create databases to organize your Apache Iceberg tables, and tables to define the schema and location of your data in the S3 tables catalog.

Create a database (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator or database creator.
2. In the navigation pane, choose **Databases** under **Data Catalog**.
3. Choose **Create database**.
4. On the **Create database** page, choose the **Database** option, and enter the following details:
   - **Name** – A unique name for the database
   - **Data catalog** – Choose the S3 tables catalog. The database will reside in this catalog.
   - **Description** –(Optional) Add a description and location.

- **IAM access control for new tables** – Optionally select Use only IAM access control for new tables in this database. For information about this option, see the Changing the default settings for your data lake section.
- Choose **Create database**.You can see the database created under the S3 tables catalog.

Create a database using AWS CLI

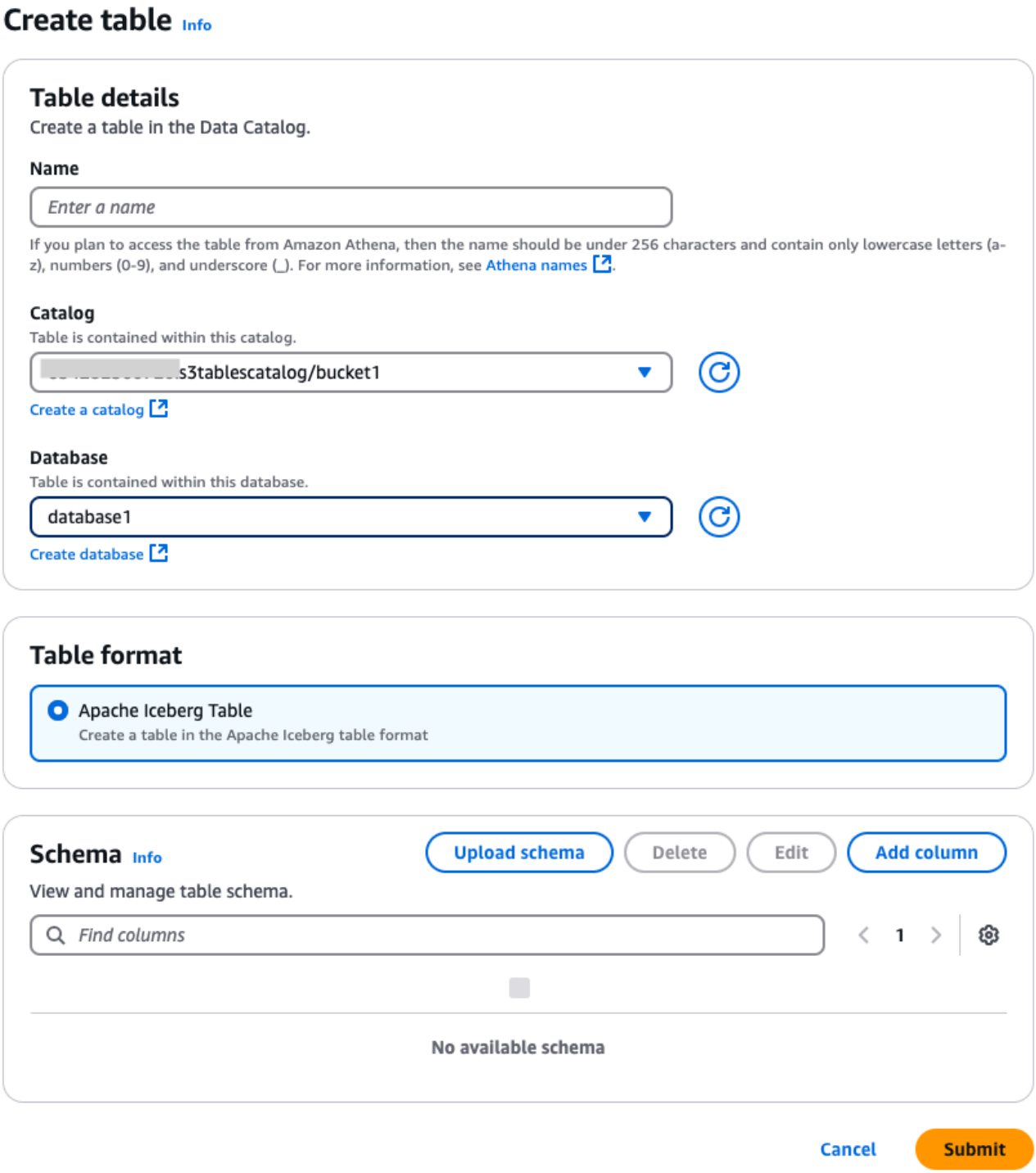
The following CLI command shows how to create a database in the S3 tables catalog.

```
aws glue create-database
--region us-east-1 \
--catalog-id "123456789012:s3tablescatalog/test" \
--database-input \
 '{ "Name": "testglueclidbcreation" }'
```

Create a table (AWS Management Console)

You can create Apache Iceberg metadata tables in the S3 tables catalog using Lake Formation console or the AWS Glue `CreateTable` API.

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator or a user with `CreateTable` permission.
2. In the navigation pane, choose **Tables** under Data Catalog.
3. Choose Create table.
4. On the **Create table** page, enter the table details:



- **Name** – Enter a unique name for the table.
- **Catalog** – Choose the S3 tables catalog as the catalog.
- **Database** – Choose the database under the S3 tables catalog.

- **Description** – Enter a description for the table.
- **Schema** – Choose Add columns to add columns and data types of the columns. You have the option to create an empty table and update the schema later. Iceberg allows you to evolve schema and partition after you create the table. You can use Athena queries to update the table schema and Spark queries for updating partitions.

5. Choose **Submit**.

Create a table (AWS CLI)

```
aws glue create-table \
--database-name "testglueclidbcreation" \
--catalog-id "123456789012:s3tablescatalog/test" \
--region us-east-1 \
--table-input \
'{ "Name": "testtablegluecli", "Parameters": { "format": "ICEBERG" },
 "StorageDescriptor": { "Columns": [ {"Name": "x", "Type": "int", "Parameters":
 {"required": "true"}} ] } }'
```

Registering an Amazon S3 table bucket in another AWS account

You can register individual Amazon S3 table buckets (ARN format:`arn:aws:s3tables:us-east-1:`*`account-id`*`:bucket/`*`bucket-name`*) from one AWS account with Lake Formation in another account. For example, you can register a table bucket from account A in account B's Lake Formation.

Prerequisites

Before beginning the cross-account bucket registration:

- Create a table bucket in account A.
- Create an IAM role in account B with appropriate permissions for bucket registration.

  For more information about the permissions required to register a table bucket with Lake Formation, see Prerequisites for integrating Amazon S3 tables catalog with the Data Catalog and Lake Formation.
- Register the table bucket in the account where you are going to create the S3 table catalog.

- For cross-account access, the role specified when registering the table bucket must be an in-account role with appropriate permissions to access the cross-account bucket. The role needs necessary S3 Tables IAM actions to access bucket resources.

Cross-account table bucket registration

In the following procedures, account A is the resource owning account, and account B is where the table bucket will be registered for managing access permissions.

1. Sign in to the AWS Management Console in account A.

   Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a table bucket. For more information, see Creating a table bucket in the Amazon S3 User Guide.
3. Register the table bucket in account B.

   Use the AWS CLI to register the table bucket from account A with Lake Formation in account B.

   ```
   aws lakeformation register-resource \
   --resource-arn 'arn:aws:s3tables:us-east-1:account-A-id:bucket/single-bucket-name'
    \
   --role-arn arn:aws:iam::account-B-id:role/role-name \
   --region us-east-1
   ```

   Replace account-A-id, single-bucket-name, account-B-id, and role-name with your specific values.
4. Next, create a catalog for the table bucket in account B.

   Create a catalog using the AWS CLI.

   ```
   aws glue create-catalog --region us-east-1 \
   --cli-input-json \
   '{
      "Name": "catalog-name",
      "CatalogInput" : {
         "FederatedCatalog": {
            "Identifier": "arn:aws:s3tables:us-east-1:account A:bucket/single-bucket-
   name",
            "ConnectionName": "aws:s3tables"
         },
   ```

```
      "CreateDatabaseDefaultPermissions": [],
      "CreateTableDefaultPermissions": []
   }
}'
```

Replace catalog-name, account-A-id, and bucket-name with your specific values.

The following CLI example shows how to view the details of the catalog.

```
aws glue get-catalog \
  --catalog-id account-id:catalog-name \
  --region us-east-1
```

5. Next, create databases and tables in newly created catalog in account B.

   Create a database.

```
aws glue create-database \
  --region us-east-1 \
  --catalog-id "account-B-id:catalog-name" \
  --database-input \
'{
  "Name": "database-name"
}'
```

   Create a table.

```
aws glue create-table \
  --database-name database-name \
  --catalog-id account-B-id:catalog-name\
  --region us-east-1 \
  --table-input \
  '{
        "Name": "table-name",
        "Parameters": {
            "format": "ICEBERG"
        },
        "StorageDescriptor": {
           "Columns": [
        {"Name": "x", "Type": "int", "Parameters": {"required": "true"}}
          ]
```

```
        }
}'
```

Replace database-name, account-B-id, catalog-name, and table-name with your specific values.

6. The following examples show how to view the objects in the catalog.

   View database details.

```
aws glue get-database \
  --name database-name \
  --catalog-id account-B-id:catalog-name \
  --region us-east-1
```

   View table details.

```
aws glue get-table \
  --name table-name \
  --database-name database-name \
  --catalog-id account-B-id:catalog-name \
  --region us-east-1
```

   Replace database-name, account-B-id, catalog-name, and table-name with your specific values.

Granting permissions

After integrating your S3 tables with AWS Lake Formation, you can grant permissions on the S3 tables catalog and the catalog objects (table buckets, databases, tables) to other IAM roles and users in your account. Lake Formation permissions allows you to define access controls at table, column, and row-level granularity for users of integrated analytical engines such as Amazon Redshift Spectrum and Athena.

You can grant permissions by using either the named resource method or the Lake Formation tag-based access control (LF-TBAC) method. Before granting permissions using LF-Tags and LF-Tag expressions, you must define them and assing them to Data Catalog objects.

For more information, see Managing LF-Tags for metadata access control.

You can share databases and tables with external AWS accounts by granting Lake Formation permissions to the external accounts. Users can then run queries and jobs that join and query tables across multiple accounts. When you share a catalog resource with another account, principals in that account can operate on that resource as if the resource were in their Data Catalog.

When you share databases and tables with external accounts, the **Super user** permission is not available.

For detailed instructions about granting permissions, see the Managing Lake Formation permissions section.

AWS CLI example for granting permissions on an Amazon S3 Table

```
aws lakeformation grant-permissions \
--cli-input-json \
'{
    "Principal": {
        "DataLakePrincipalIdentifier":"arn:aws:iam::111122223333:role/DataAnalystRole"
    },
    "Resource": {
        "Table": {
            "CatalogId":"111122223333:s3tablescatalog/amzn-s3-demo-bucket1",
            "DatabaseName":"S3 table bucket namespace <example_namespace>",
            "Name":"S3 table bucket table name <example_table>"
        }
    },
    "Permissions": [
        "SELECT"
    ]
}'
```

The following are the parameters to include in the command:

- DataLakePrincipalIdentifier – IAM user, role, or group ARN to grant permissions
- CatalogId – 12-digit AWS account ID that owns the Data Catalog
- DatabaseName – Name of the Amazon S3 table bucket name space
- Name – Amazon S3 table bucket table name
- Permissions – Permissions to grant. Options include: SELECT, INSERT, DELETE, DESCRIBE, ALTER, DROP, ALLL, and SUPER

Accessing shared Amazon S3 tables

After you grant cross-account permissions on a database or table in the S3 tables catalog, to access the resources, you need to create resource links to the shared databases and tables.

1. In the destination account (the account that receives the shared resources), create a database resource link. For detailed instructions, see Creating a resource link to a shared Data Catalog database.

   CLI example for creating a database resource link

```
aws glue create-database
--region us-east-1
--catalog-id "111122223333"
--database-input \
'{
  "Name": "s3table_resourcelink",
  "TargetDatabase": {
    "CatalogId": "011426214932:s3tablescatalog/chmni-s3-table-bucket-011426214932",
    "DatabaseName": "s3_table_ns"
  },
  "CreateTableDefaultPermissions": []
}'
```

2. Grant cross account permission on the table.

   CLI example for cross-account permission grant

```
aws lakeformation grant-permissions \
--region us-east-1 \
--cli-input-json \
'{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
S3TablesTestExecRole"
    },
    "Resource": {
        "Table": {
            "CatalogId": "011426214932:s3tablescatalog/chmni-s3-table-
bucket-011426214932",
            "DatabaseName": "s3_table_ns",
            "Name": "test_s3_iceberg_table"
        }
```

```
    },
    "Permissions": [
        "ALL"
    ]
}'
```

3. Grant Lake Formation DESCRIBE permission on the resource link.

   CLI example for granting describe permission on the resource link.

```
aws lakeformation grant-permissions \
    --principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:role/
S3TablesTestExecRole
    --resource Database='{CatalogId=111122223333;, Name=s3table_resourcelink}' \
    --permissions DESCRIBE
```

Changing access controls for S3 Tables integration

After you have integrated Amazon S3 Tables with the AWS Glue Data Catalog, you can change how access to your catalog resources is controlled. This section how to change access control depending on your current and desired access control model. Enabling Lake Formation allows you to use fine-grained permissions such as column-level and row-level security through Lake Formation grants, and allows Lake Formation to vend temporary credentials on behalf of principals through a registered role. Changing access control from AWS Lake Formation to IAM returns access control to standard IAM policies, which may be appropriate if your workloads do not require fine-grained access and you prefer to manage permissions entirely through IAM. Both migration paths involve updating the Data Catalog defaults, adjusting resource registrations with Lake Formation, and coordinating permission grants to avoid access disruptions during the transition.

Topics

- Enable Lake Formation with S3 Tables integration with Data Catalog
- Change access control from AWS Lake Formation to IAM

Enable Lake Formation with S3 Tables integration with Data Catalog

This section describes the workflow to migrate access control from IAM privileges to IAM with AWS Lake Formation grants for Amazon S3 Tables integrated with the AWS Glue Data Catalog.

Important

Enabling AWS Lake Formation access control will revoke all existing IAM-based access to your S3 Tables resources. After completing Step 1, users and roles that previously accessed data through IAM permissions will immediately lose access. You must grant Lake Formation permissions in Step 2 before users can query data again. Plan this migration during a maintenance window and coordinate with your data team.

Prerequisites

For read/write access to S3 Tables, in addition to Lake Formation permissions, principals also need the `lakeformation:GetDataAccess` IAM permission. With this permission, Lake Formation grants the request for temporary credentials to access the data.

Using AWS CLI

1. **Step 1: Register bucket with Lake Formation using IAM role**

   Register the S3 Tables resource with Lake Formation.

   Note

   If you have an existing role, ensure hybrid access is false.

   ```
   aws lakeformation register-resource \
     --resource-arn "arn:aws:s3tables:us-east-1:AWSAccountID:bucket/*" \
     --role-arn "arn:aws:iam::AWSAccountID:role/service-role/LFAccessRole" \
     --with-federation
   ```

2. **Step 2: Update AWS Glue catalog to enable Lake Formation access control**

   Update the catalog with empty `CreateDatabaseDefaultPermissions` and `CreateTableDefaultPermissions` (set to `[]`). This ensures that databases and tables in the catalog are managed using Lake Formation grants instead of IAM-based default permissions.

   ```
   aws glue update-catalog \
   ```

```
  --catalog-id "s3tablescatalog" \
  --catalog-input '{
    "FederatedCatalog": {
        "Identifier": "arn:aws:s3tables:us-east-1:AWSAccountID:bucket/*",
        "ConnectionName": "aws:s3tables"
    },
    "CreateDatabaseDefaultPermissions": [],
    "CreateTableDefaultPermissions": [],
    "AllowFullTableExternalDataAccess": "True"
  }'
```

3. **Step 3: Grant Lake Formation permissions to your data team**

   Grant Lake Formation permissions to the principals (roles, users, or groups) that need access. For example, to grant full-table read access to a role:

```
aws lakeformation grant-permissions \
  --principal
 DataLakePrincipalIdentifier=arn:aws:iam::AWSAccountID:role/DataTeamRole \
  --resource '{
    "Table": {
        "CatalogId": "AWSAccountID",
        "DatabaseName": "s3tablescatalog/table-bucket-name/namespace",
        "TableWildcard": {}
    }
  }' \
  --permissions "SELECT" "DESCRIBE"
```

   Repeat for each principal and resource combination as needed.

Change access control from AWS Lake Formation to IAM

This section describes the workflow to change access control from AWS Lake Formation grants to IAM privileges for Amazon S3 Tables integrated with the AWS Glue Data Catalog.

> **Important**
>
> Changing access control from AWS Lake Formation grants to IAM will revoke all existing Lake Formation-based access to your S3 Tables resources. After completing Step 2, users and roles that previously accessed data through Lake Formation grants will immediately

lose access. You must grant IAM access in Step 1 before updating the catalog. Plan this migration during a maintenance window and coordinate with your data team.

Important

Fine-grained access controls, such as column-level access and data cell filters, with Data Catalog objects are available when using AWS Lake Formation only. Before proceeding to migrate access controls from AWS Lake Formation to IAM, audit your existing Lake Formation grants using `aws lakeformation list-permissions` and determine whether equivalent IAM policies can provide the access your users need. Any principal that relied on fine-grained Lake Formation grants will require full table-level IAM access after migrating access control.

Prerequisites

Before you begin, ensure the following:

- You have identified all Lake Formation grants currently in effect for the resources being migrated. Run `aws lakeformation list-permissions --resource-type TABLE` to review them.
- You have prepared IAM policies that provide equivalent access for all affected principals.
- The IAM role registered with Lake Formation still has `lakeformation:GetDataAccess` (needed during the hybrid transition period).

Using AWS CLI

1. **Step 1: Grant IAM permissions to principals**

   Attach IAM policies to the users or roles that need access. The policy must include both AWS Glue metadata permissions and S3 Tables data permissions.

   Note

   The following example policy only provides read access.

```
aws iam put-user-policy \
  --user-name GlueIAMAccessUser \
  --policy-name S3TablesIAMAccessPolicy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GlueMetadataAccess",
            "Effect": "Allow",
            "Action": [
                "glue:GetCatalog",
                "glue:GetDatabase",
                "glue:GetTable"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:AWSAccountID:catalog/s3tablescatalog",
                "arn:aws:glue:us-east-1:AWSAccountID:database/
s3tablescatalog/table-bucket-name/namespace",
                "arn:aws:glue:us-east-1:AWSAccountID:table/s3tablescatalog/table-
bucket-name/namespace/*"
            ]
        },
        {
            "Sid": "S3TablesDataAccess",
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTableBucket",
                "s3tables:GetTable",
                "s3tables:GetTableMetadataLocation",
                "s3tables:GetTableData"
            ],
            "Resource": [
                "arn:aws:s3tables:us-east-1:AWSAccountID:bucket/table-bucket-name",
                "arn:aws:s3tables:us-east-1:AWSAccountID:bucket/table-bucket-name/
table/*"
            ]
        }
    ]
  }'
```

Verify that all affected users and roles can access the expected tables using their IAM credentials before proceeding.

2. **Step 2: Update the catalog to restore IAM default permissions**

   Update the catalog so that `CreateDatabaseDefaultPermissions` and `CreateTableDefaultPermissions` grant `ALL` to `IAM_ALLOWED_PRINCIPALS`. Set `OverwriteChildResourcePermissionsWithDefault` to `Accept` so the change propagates to all existing child resources, not just newly created ones.

```
aws glue update-catalog \
  --catalog-id "s3tablescatalog" \
  --catalog-input '{
    "FederatedCatalog": {
        "Identifier": "arn:aws:s3tables:us-east-1:AWSAccountID:bucket/*",
        "ConnectionName": "aws:s3tables"
    },
    "CreateDatabaseDefaultPermissions": [{
        "Principal": {"DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"},
        "Permissions": ["ALL"]
    }],
    "CreateTableDefaultPermissions": [{
        "Principal": {"DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"},
        "Permissions": ["ALL"]
    }],
    "OverwriteChildResourcePermissionsWithDefault": "Accept"
  }'
```

3. **Step 3: Deregister the resource from Lake Formation**

   Once you have confirmed that all access is working through IAM policies and no principals depend on Lake Formation grants, you can deregister the resource from Lake Formation to complete the migration.

```
aws lakeformation deregister-resource \
  --resource-arn "arn:aws:s3tables:us-east-1:AWSAccountID:bucket/*"
```

> **Note**
>
> After deregistering the resource, remove `lakeformation:GetDataAccess` from IAM principals that no longer need it.

No `revoke-permissions` step is required.

Creating an Amazon Redshift managed catalog in the AWS Glue Data Catalog

You might not have an Amazon Redshift producer cluster or an Amazon Redshift datashare available today, but want to create and manage Amazon Redshift tables using the AWS Glue Data Catalog. You can get started by creating an AWS Glue managed catalog using the `glue:CreateCatalog` API or the AWS Lake Formation console by setting the catalog type as `Managed` and `Catalog source` as **Redshift.** This step does the following:

- Creates a catalog in the Data Catalog
- Registers the catalog as a Lake Formation data location
- creates an Amazon Redshift managed serverless-workgroup
- Links Amazon Redshift serverless workgroup and Data Catalog using a datashare object

To create a managed catalog and set up permissions (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, choose **Catalogs** under **Data Catalog**.
3. Select the option **Create catalog**.
4. On the **Set Catalog** details page, enter the following information:
   - **Name** – A unique name for your managed catalog. The name can't be changed, and must be in lower case. The name can consist of a maximum of 255 characters maximum. account.
   - **Type** – Choose `Managed catalog` as the catalog type.
   - **Storage** – Choose `Redshift` for storage.
   - **Description** – Enter a description for the catalog created from the data source.

5. You can use Apache Spark applications running on Amazon EMR on Amazon EC2 to access the Amazon Redshift databases in the AWS Glue Data Catalog.

   To enable Apache Spark to read and write to Amazon Redshift managed storage, AWS Glue creates a managed Amazon Redshift cluster with the compute and storage resources required to perform read and write operations without impacting Amazon Redshift data warehouse workloads. You also need to provide an IAM role with the permissions required to transfer data to and from the Amazon S3 bucket. For the permissions required for the data transfer role, see step 5 in the Prerequisites for managing Amazon Redshift namespaces in the AWS Glue Data Catalog section.

6. By default, the data in the Amazon Redshift cluster is encrypted using an AWS managed key. Lake Formation provides an option to create your custom KMS key for encryption. If you're using a customer managed key, you must add specific key policies to the key.

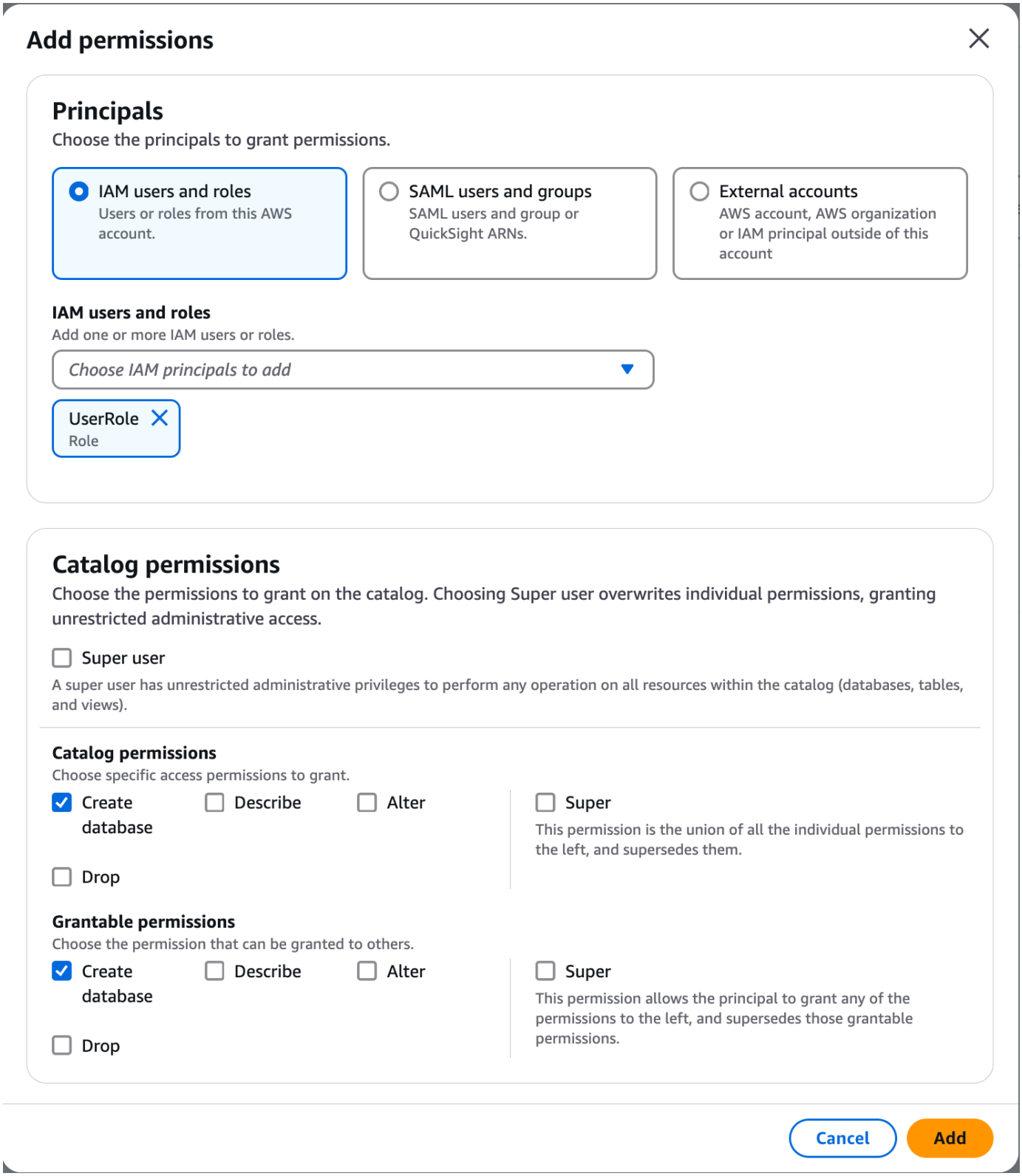
7. Choose the Customize encryption settings if you're using a customer managed key to encrypt the data in the Amazon Redshift managed storage. To use a custom key, you must add additional custom managed key policy to your KMS key. For more information, see Prerequisites for managing Amazon Redshift namespaces in the AWS Glue Data Catalog.

8. **Encryption options** – Choose **Customize encryption settings** option if you want to use a custom key to encrypt the catalog. To use a custom key, you must add additional custom managed key policy to your KMS key.

9. Choose **Next** to grant permissions to other principals.

10. On the **Grant permissions** page, choose **Add permissions**.

11. On the **Add permissions** screen, choose the principals and the types of permissions to grant.



- In the **Principals** section, choose a principal type and then specify principals to grant permissions.

- **IAM users and roles** – Choose one or more users or roles from the IAM users and roles list.
- **SAML users and groups** – For SAML and Amazon Quick users and groups, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon Quick users or groups. Press **Enter** after each ARN.

  For information about how to construct the ARNs, see AWS CLI grant and revoke AWS CLI commands.

- In the **Permissions** section, select permissions and grantable permissions.

  Under **Catalog permissions**, select one or more permissions to grant.

  Choose **Super user** to grant unrestricted administrative permissions on all resources within the catalog.

  Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account. This option is not supported when you are granting permissions to an IAM principal from an external account.

12. Choose **Next** to review the information and create the catalog. The **Catalogs** list shows the new managed catalog.

To create a federated catalog (CLI)

- The following example shows how to create a federated catalog.

```
aws glue create-catalog --cli-input-json file://input.json

{
    "Name": "CatalogName",
    "CatalogInput": {
      "Description": "Redshift published Catalog",
      "CreateDatabaseDefaultPermissions" : [],
      "CreateTableDefaultPermissions": [],
      "CatalogProperties": {
        "DataLakeAccessProperties" : {
          "DataLakeAccess" : "true",
          "DataTransferRole" : "DTR arn",
          "KMSKey": "kms key arn",  // Optional
          "CatalogType": "aws:redshift"
        }
      }
```

```
    }
}
```

Glue get-catalog response

```
aws glue get-catalog \
  --catalog-id account-id:catalog-name \
  --region us-east-1

Response:
{
    "Catalog": {
        "Name": "CatalogName",
        "Description": "Glue Catalog for Redshift z-etl use case",
        "CreateDatabaseDefaultPermissions" : [],
        "CreateTableDefaultPermissions": [],
         "CatalogProperties": {
          "DataLakeAccessProperties" : {
            "DataLakeAccess": "true",
            "DataTransferRole": "DTR arn",
            "KMSKey": "kms key arn",
            "ManagedWorkgroupName": "MWG name",
            "ManagedWorkgroupStatus": "MWG status",
            "RedshiftDatabaseName": "RS db name",
            "NamespaceArn": "namespace key arn",
            "CatalogType": "aws:redshift"
          }
       }
    }
```

Managing permissions for data in an Amazon Redshift datashare

With AWS Lake Formation, you can manage data securely in a datashare from Amazon Redshift. Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the AWS Cloud. Using the data sharing capability, Amazon Redshift helps you to share data across AWS accounts. For more information about Amazon Redshift data sharing, see Overview of data sharing in Amazon Redshift.

In Amazon Redshift, the producer cluster administrator creates a datashare, and shares it with the data lake administrator. For step-by-step instructions on creating a data lake administrator, see Create a data lake administrator.

After you (data lake administrator) accept the datashare, you must create an AWS Glue Data Catalog database for the specific datashare. This is so that you can control access to it using Lake Formation permissions. Lake Formation maps each datashare to a corresponding Data Catalog database. These appear as federated databases in the Data Catalog.

A database is referred to as a *federated database* when it points to an entity outside of the Data Catalog. Tables and views in the Amazon Redshift datashare are listed as individual tables in the Data Catalog. You can share the federated database with selected IAM principals and SAML users within the same account or in another account with Lake Formation. You can also include row and column filter expressions to restrict access to certain data. For more information, see Data filtering and cell-level security in Lake Formation.

To provide users access to an Amazon Redshift datashare, you must do the following:

1. Update **Data Catalog settings** to enable Lake Formation permissions.
2. Accept the datashare invitation from the Amazon Redshift producer cluster administrator and register the datashare in Lake Formation.

   After completing this step, you can manage the datashare within the Lake Formation Data Catalog.
3. Create a federated database and define permissions on that database.
4. Grant permissions to users on databases and tables. You can share the entire database or a subset of tables with users in the same account or another account.

For limitations, see Amazon Redshift data sharing limitations.

Topics

- Prerequisites for setting up permissions on Amazon Redshift datashares
- Setting up permissions for Amazon Redshift datashares
- Querying federated databases

Prerequisites for setting up permissions on Amazon Redshift datashares

Update default Data Catalog settings

To enable Lake Formation permissions for the Data Catalog resources, we recommend that you disable the default **Data Catalog settings** in Lake Formation. For more information, see Change the default permission model or use hybrid access mode.

Update permissions

In addition to data lake administrator permissions (`AWSLakeFormationDataAdmin`), the following permissions are also required to accept an Amazon Redshift datashare in Lake Formation:

- `glue:PassConnection on aws:redshift`
- `redshift:AssociateDataShareConsumer`
- `redshift:DescribeDataSharesForConsumer`
- `redshift:DescribeDataShares`

The data lake administrator IAM user has the following permissions implicitly.

- data_location_access
- create_database
- lakefomation:registerResource

Setting up permissions for Amazon Redshift datashares

This topic describes the steps you need to follow to accept a datashare invitation, create a federated database, and grant permissions. You can use the Lake Formation console or the AWS Command Line Interface (AWS CLI). The examples in this topic show the producer cluster, the Data Catalog, and the data consumer in the same account.

To learn more about Lake Formation cross-account capabilities, see Cross-account data sharing in Lake Formation.

To set up permissions for a datashare

1. Review a datashare invitation and accept it.

Console

1. Sign in to the Lake Formation console as a data lake administrator at https://console.aws.amazon.com/lakeformation/. Navigate to the **Data sharing** page.
2. Review the datashares that you're authorized to access. The **Status** column indicates your current participation status for the datashare. The **Pending** status indicates that you have been added to a datashare, but you have not yet accepted it or have rejected the invitation.
3. To respond to a datashare invitation, select the datashare name and choose **Review invitation**. In **Accept or reject datashare**, review the invitation details. Choose **Accept** to accept the invitation or **Reject** to decline the invitation. You don't get access to the datashare if you reject the invitation.

AWS CLI

The following examples show how to view, accept, and register the invitation. Replace the AWS account ID with a valid AWS account ID. Replace the `data-share-arn` with the actual Amazon Resource Name (ARN) that references the datashare.

1. View a pending invitation.

```
aws redshift describe-data-shares \
 --data-share-arn 'arn:aws:redshift:us-
east-1:111122223333:datashare:abcd1234-1234-ab12-cd34-1a2b3c4d5e6f/
federatedds' \
```

2. Accept a datashare.

```
 aws redshift associate-data-share-consumer \
 --data-share-arn 'arn:aws:redshift:us-
east-1:111122223333:datashare:abcd1234-1234-ab12-cd34-1a2b3c4d5e6f/
federatedds' \
 --consumer-arn 'arn:aws:glue:us-east-1:111122223333:catalog
```

3. Register the datashare in the Lake Formation account. Use the RegisterResource API operation to register the datashare in Lake Formation. `DataShareArn` is the input parameter for `ResourceArn`.

> **Note**
>
> This is a mandatory step.

```
aws lakeformation register-resource \
 --resource-arn 'arn:aws:redshift:us-
east-1:111122223333:datashare:abcd1234-1234-ab12-cd34-1a2b3c4d5e6f/
federatedds'
```

2. Create a database.

After you've accepted a datashare invitation, you need to create a database that points to the Amazon Redshift database associated with the datashare. You must be a data lake administrator to create a database.

Console

1. Select the datashare from the **Invitations** pane and choose **Set database details**.
2. In **Set database details**, enter a unique name and identifier for the datashare. You use this identifier for mapping the datashare internally in the metadata hierarchy (dbName.schema.table).
3. Choose **Next** to grant permissions to other users on the shared database and tables.

AWS CLI

Use the following example code to create a database that points to the Amazon Redshift database shared with Lake Formation using the AWS CLI.

```
aws glue create-database --cli-input-json \

'{
 "CatalogId": "111122223333",
 "DatabaseInput": {
  "Name": "tahoedb",
  "FederatedDatabase": {
       "Identifier": "arn:aws:redshift:us-
east-1:111122223333:datashare:abcd1234-1234-ab12-cd34-1a2b3c4d5e6f/federatedds",
       "ConnectionName": "aws:redshift"
```

```
    }
  }
  }'
```

3. Grant permissions.

After you've created the database, you can grant permissions to users in your account or to external AWS accounts and organizations. You'll not be able to grant write data permissions (insert, delete) and metadata permissions (alter, drop, create) on the federated database that is mapped to an Amazon Redshift datashare. For more information on granting permissions, see Managing Lake Formation permissions.

> **Note**
>
> As a data lake administrator, you can only view tables in the federated databases. To perform any other action, you need to grant yourself more permissions on those tables.

Console

1. On the **Grant permissions** screen, select the users to grant permissions to.
2. Choose **Grant**.

AWS CLI

Use the following examples to grant database and table permissions using the AWS CLI:

```
aws lakeformation grant-permissions --input-cli-json file://input.json

{
   "Principal": {
           "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/non-
admin"
   },
   "Resource": {
          "Database": {
                "CatalogId": "111122223333",
                 "Name": "tahoedb"
           }
```

```
    },
    "Permissions": [
             "DESCRIBE"
     ],
    "PermissionsWithGrantOption": [

     ]
 }
```

```
aws lakeformation grant-permissions --input-cli-json file://input.json

{
                   "Principal": {
                         "DataLakePrincipalIdentifier":
 "arn:aws:iam::111122223333:user/non-admin"
                   },
                  "Resource": {
                         "Table": {
                              "CatalogId": "111122223333",
                              "DatabaseName": "tahoedb",
                              "Name": "public.customer"
                          }
                  },
                 "Permissions": [
                        "SELECT"
                  ],
                 "PermissionsWithGrantOption": [
                         "SELECT"
                   ]
 }
```

Querying federated databases

After you grant permissions, users can sign in and start querying the federated database using Amazon Redshift. Users can now use the local database name to reference the Amazon Redshift datashare in SQL queries. In Amazon Redshift, the customer table in the public schema that is shared through the datashare will have a corresponding table created as `public.customer` in the Data Catalog.

1. Before querying the federated database using Amazon Redshift, the cluster administrator creates a database from the Data Catalog database using the following command:

```
CREATE DATABASE sharedcustomerdb FROM ARN
 'arn:aws:glue:<region>:111122223333:database/tahoedb' WITH DATA CATALOG SCHEMA
 tahoedb
```

2. The cluster admin grants usage permissions on the database.

```
GRANT USAGE ON DATABASE sharedcustomerdb TO IAM:user;
```

3. You (the federated user) can now log in into SQL tools to query the table.

```
Select * from sharedcustomerdb.public.customer limit 10;
```

For more information, see Querying AWS Glue Data Catalog in Amazon Redshift Management Guide.

Managing permissions on datasets that use external metastores

With AWS Glue Data Catalog metadata federation (Data Catalog federation), you can connect the Data Catalog to external metastores that store metadata for your Amazon S3 data, and securely manage data access permissions using AWS Lake Formation. You don't have to migrate the metadata from the external metastore into the Data Catalog.

The Data Catalog provides a centralized metadata repository that makes managing and discovering data across disparate systems easier. When your organization manages data in the Data Catalog, you can use AWS Lake Formation to control access to your datasets in Amazon S3.

> **Note**
>
> Currently, we support only Apache Hive (version 3 and above) metastore federation.

To set up Data Catalog federation, we provide an AWS Serverless Application Model (AWS SAM) application called GlueDataCatalogFederation-HiveMetastore in the AWS Serverless Application Repository.

The reference implementation is provided on GitHub as an open source project at AWS Glue Data Catalog Federation - Hive Metastore.

The AWS SAM application creates and deploys the following resources that are required for connecting the Data Catalog to the Hive metastore:

- **An AWS Lambda function** – Hosts the implementation of the federation service that communicates between the Data Catalog and the Hive metastore. AWS Glue invokes this Lambda function to retrieve metadata objects from the Hive metastore.
- **Amazon API Gateway** – The connection endpoint for your Hive metastore that acts as a proxy to route all invocations to the Lambda function.
- **An IAM role** – A role with necessary permissions to create the connection between the Data Catalog and the Hive metastore.
- **AWS Glue connection** – An Amazon API Gateway type of AWS Glue connection that stores the Amazon API Gateway endpoint and an IAM role to invoke it.

When you query tables, the AWS Glue service makes a runtime call to the Hive metastore and fetches the metadata. The Lambda function acts as a translator between the Hive metastore and Data Catalog.

After establishing the connection, in order to sync the metadata in the Hive metastore with the Data Catalog, you need to create a *federated database* in the Data Catalog using the Hive metastore connection details, and map this database to the Hive database. A database is referred as a federated database when it points to an entity outside the Data Catalog.

You can apply Lake Formation permissions using tag-based access control and the named resource method on the federated database, and share it across multiple AWS accounts, AWS Organizations, and organizational units (OUs). You can also share the federated database directly with IAM principals from another account.

You can define fine-grained permissions at column level, row level, and cell level using Lake Formation data filters on the external Hive tables. You can use Amazon Athena, Amazon Redshift, or Amazon EMR to query the Lake Formation managed external Hive tables.

For more information on cross-account data sharing and data filtering, see:

- Cross-account data sharing in Lake Formation
- Data filtering and cell-level security in Lake Formation

Data Catalog metadata federation high-level steps

1. You create IAM users and roles that have appropriate permissions to deploy the AWS SAM application and to create federated databases.
2. You register the Amazon S3 data location with Lake Formation by selecting the `Enable Data Catalog federation` option for datasets that use an external Hive metastore.
3. You configure the AWS SAM application settings (AWS Glue connection name, URL to the Hive metastore, and Lambda function parameters) and deploy the AWS SAM application.
4. The AWS SAM application deploys the resources that are required to connect the external Hive metastore with the Data Catalog.
5. To apply Lake Formation permissions on the Hive database and tables, you create a database in the Data Catalog using the Hive metastore connection details, and map this database to the Hive database.
6. Grant permissions on the federated databases to principals in your account or in another account.

> **Note**
>
> You can connect the Data Catalog to an external Hive mestastore, create federated databases, and run queries and ETL scripts on Hive databases and tables without applying Lake Formation permissions. For source data in Amazon S3 that isn't registered with Lake Formation, access is determined by IAM permissions policies for Amazon S3 and AWS Glue actions.

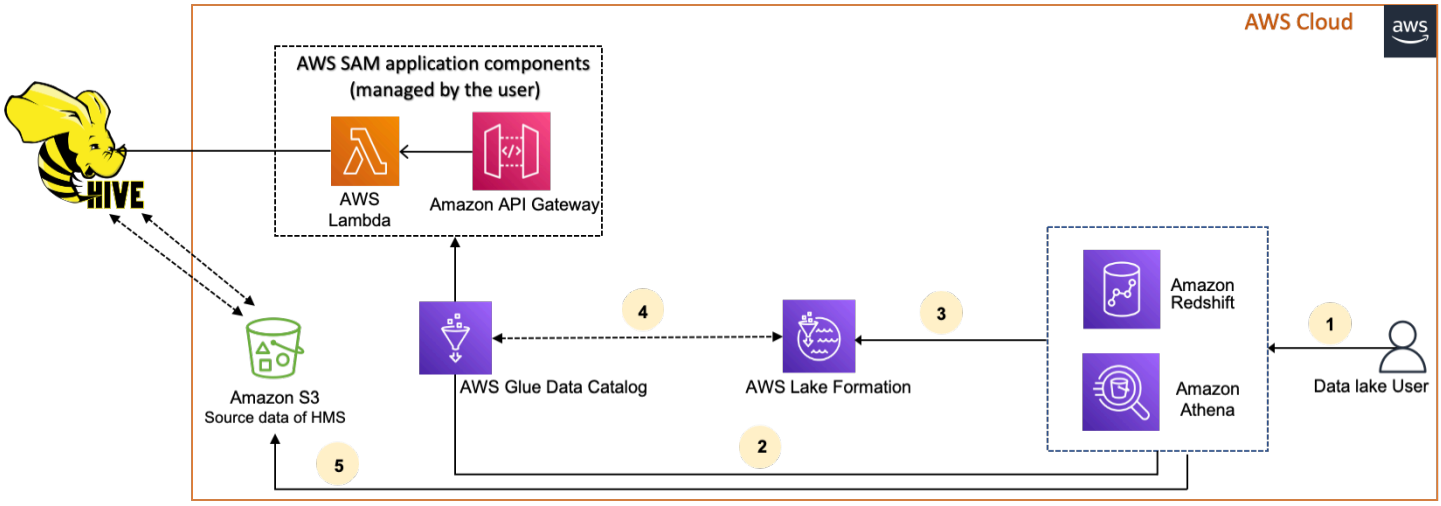
For limitations, see Hive metadata store data sharing considerations and limitations.

Topics

- Workflow
- Prerequisites for connecting the Data Catalog to the Hive metastore
- Connecting the Data Catalog to an external Hive metastore
- Additional resources

Workflow

The following diagram shows the workflow for connecting the AWS Glue Data Catalog to an external Hive metastore.



1. A principal submits a query using an integrated service such as Athena or Redshift Spectrum.
2. The integrated service makes a call to the Data Catalog for the metadata, which in turn calls the Hive metastore endpoint available behind Amazon API Gateway, and receives responses to metadata requests.
3. The integrated service sends the request to Lake Formation to verify table information and credentials to access the table.
4. Lake Formation authorizes the request and vends temporary credentials to the integrated application, which allows data access.
5. Using the temporary credentials received from Lake Formation, the integrated service reads the data from Amazon S3, and shares the results to the principal.

Prerequisites for connecting the Data Catalog to the Hive metastore

To connect the AWS Glue Data Catalog to an external Apache Hive metastore and set up data access permissions, you need to complete the following requirements:

> **Note**
>
> We recommend that a Lake Formation administrator deploys the AWS SAM application, and only a privileged user uses the Hive metastore connection to create the corresponding federated databases.

1. **Create IAM roles.**

   To deploy the AWS SAM application

   - Create a role that has the necessary permissions for deploying resources (Lambda function, Amazon API Gateway, IAM role, and the AWS Glue connection) required to create a connection to the Hive metastore.

   To create federated databases

   The following permissions are required on resources:

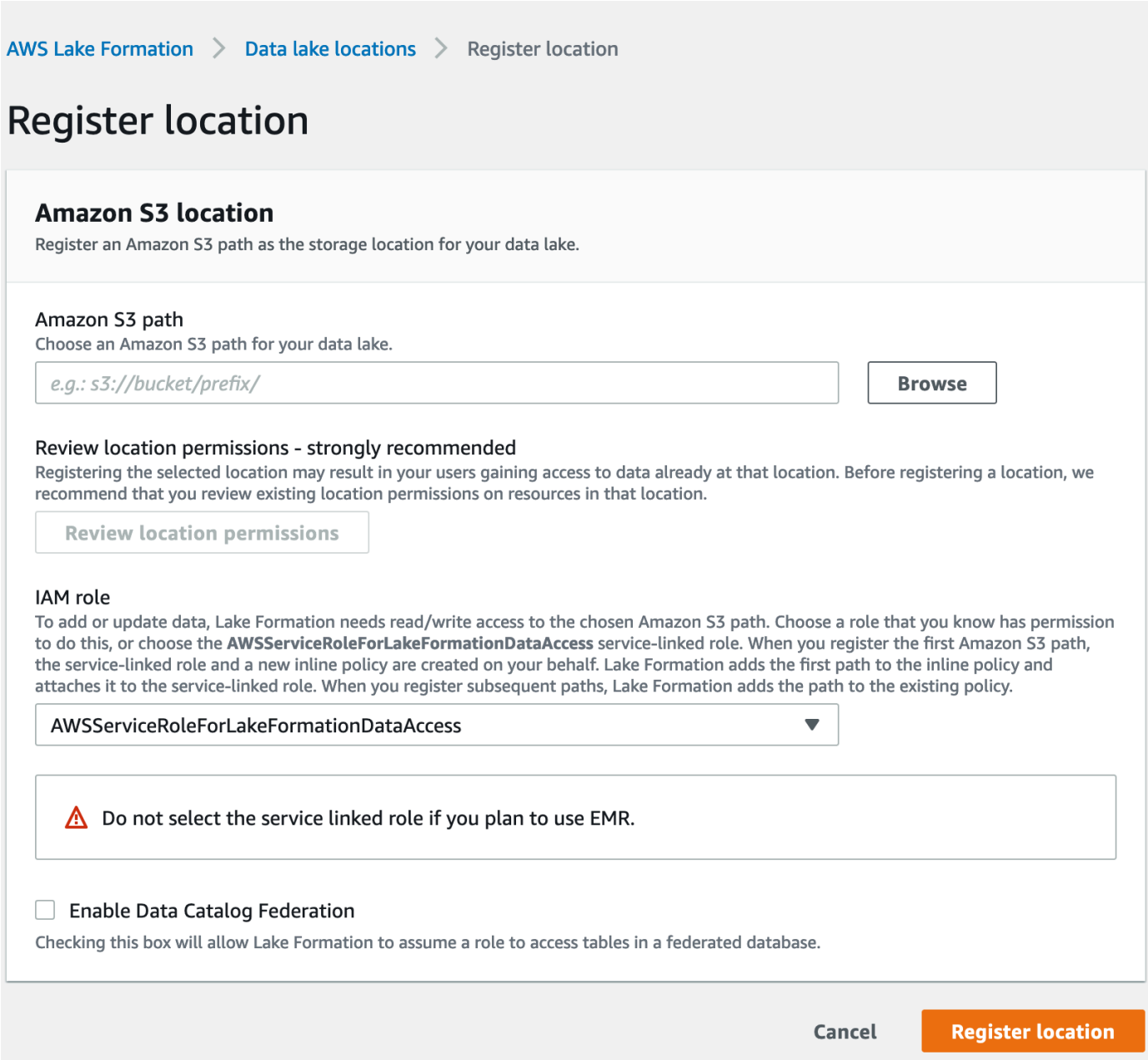
   - `glue:CreateDatabase on resource arn:aws:glue:region:account-id:database/gluedatabasename`
   - `glue:PassConnection on resource arn:aws:glue:region:account-id:connection/hms_connection`

2. **Register the Amazon S3 location with Lake Formation.**

   To use Lake Formation to manage and secure the data in your data lake, you must register the Amazon S3 location that has the data for tables in the Hive metastore with Lake Formation. By doing so, Lake Formation can vend credentials to AWS analytical services such as Athena, Redshift Spectrum, and Amazon EMR.

   For more information on registering an Amazon S3 location, see Adding an Amazon S3 location to your data lake.

   When you register the Amazon S3 location, select the **Enable Data Catalog Federation** check box to allow Lake Formation to assume a role to access tables in a federated database.



For more information about registering a data location with Lake Formation, see Configure an Amazon S3 location for your data lake.

3. **Use the correct Amazon EMR version.**

   To use Amazon EMR with the federated Hive metastore databases, you need to have Hive version 3.x or higher and Amazon EMR version 6.x or higher.

Connecting the Data Catalog to an external Hive metastore

To connect the AWS Glue Data Catalog to a Hive metastore, you need to deploy an AWS SAM application called GlueDataCatalogFederation-HiveMetastore. It creates the resources required to connect the external Hive metastore with the Data Catalog. You can access the AWS SAM application in the AWS Serverless Application Repository.

The AWS SAM application creates the connection for the Hive metastore behind Amazon API Gateway using a Lambda function. The AWS SAM application uses a uniform resource identifier (URI) as an input from the user and connects the external Hive metastore to the Data Catalog. When a user runs a query on Hive tables, the Data Catalog calls the API Gateway endpoint. The endpoint invokes the Lambda function to retrieve the metadata of the Hive tables.

To connect the Data Catalog to the Hive metastore and set up permissions

1. **Deploy the AWS SAM application.**

    1. Sign in to the AWS Management Console and open the AWS Serverless Application Repository.
    2. In the navigation pane, choose **Available applications**.
    3. Choose **Public applications**.
    4. Select the option **Show apps that create custom IAM roles or resource policies**.
    5. In the search box, enter the name **GlueDataCatalogFederation-HiveMetastore**.
    6. Choose the **GlueDataCatalogFederation-HiveMetastore** application.
    7. Under **Application Settings**, enter the following minimum required settings for your Lambda function:
        - **Application name** - A name for your AWS SAM application.
        - **GlueConnectionName** - A name for the connection.
        - **HiveMetastoreURIs** - The URI of your Hive metastore host.
        - **LambdaMemory** - The amount of Lambda memory in MB from 128-10240. The default is 1024.
        - **LambdaTimeout** - The maximum Lambda invocation runtime in seconds. The default is 30.
        - **VPCSecurityGroupIds** and **VPCSubnetIds** - Information for the VPC where the Hive metastore exists.

8. Select **I acknowledge that this app creates custom IAM roles and resource policies**. For more information, choose the **Info** link.
9. In the **Application settings** section, choose **Deploy**. When the deployment is complete, the Lambda function appears in the **Resources** section in the Lambda console.

The application is deployed to Lambda. Its name is prepended with **serverlessrepo-** to indicate that the application was deployed from the AWS Serverless Application Repository. Selecting the application takes you to the **Resources** page where each of the resources of the application that were deployed are listed. The resources include the Lambda function that allows communication between the Data Catalog and the Hive metastore, the AWS Glue connection, and other resources that are needed for the database federation.

2. **Create a federated database in the Data Catalog.**

After you've created a connection to the Hive metastore, you can create federated databases in the Data Catalog that point to the external Hive metastore databases. You need to create a corresponding database in the Data Catalog for every Hive metastore database that you're connecting to the Data Catalog.

Lake Formation console

1. On the **Data sharing** page, choose the **Shared databases** tab, and then choose **Create database**.
2. For **Connection name**, choose the name of your Hive metastore connection from the dropdown menu.
3. Enter a unique database name and the federation source identifier for the database. This is the name that you use in your SQL statements when you query tables. The name can consist of a maximum of 255 characters maximum and must be unique within your account.
4. Choose **Create database**.

AWS CLI

```
aws glue create-database \
'{
 "CatalogId": "<111122223333>",
  "database-input": {
```

```
    "Name":"<fed_glue_db>",
    "FederatedDatabase":{
        "Identifier":"<hive_db_on_emr>",
        "ConnectionName":"<hms_connection>"
     }
   }
}'
```

3. **View tables in the federated database.**

   After you've created the federated database, you can view the list of tables in your Hive metastore using the Lake Formation console or the AWS CLI.

   Lake Formation console

   1. Select the database name from the **Shared databases** tab.
   2. On the **Databases** page, choose **View tables**.

   AWS CLI

   The following examples show how to retrieve the connection definition, the database name, and some or all tables in the database. Replace the ID of the Data Catalog with the valid AWS account ID that you used to created the database. Replace `hms_connection` with the connection name.

```
aws glue get-connection \
--name <hms_connection>  \
--catalog-id 111122223333
```

```
aws glue get-database \
--name <fed_glu_db> \
--catalog-id 111122223333
```

```
aws glue get-tables \
--database-name <fed_glue_db> \
--catalog-id 111122223333
```

```
aws glue get-table \
```

```
--database-name <fed_glue_db> \
--name <hive_table_name> \
--catalog-id 111122223333
```

4. **Grant permissions.**

   After you've created the database, you can grant permissions to other IAM users and roles in your account or to external AWS accounts and organizations. You will not be able to grant write data permissions (insert, delete) and metadata permissions (alter, drop, create) on the federated databases. For more information on granting permissions, see Managing Lake Formation permissions.

5. **Query the federated databases.**

   After you grant permissions, users can sign in and start querying the federated database using Athena and Amazon Redshift. Users can now use the local database name to reference the Hive database in SQL queries.

   Example Amazon Athena query syntax

   Replace `fed_glue_db` with the local database name that you created earlier.

   ```
   Select * from fed_glue_db.customers limit 10;
   ```

Additional resources

The following blog post contains detailed instructions to set up Lake Formation permissions on a Hive metastore database and tables, and query them using Athena. We also illustrate a cross-account sharing use case, where a Lake Formation principal in producer account A shares a federated Hive database and tables using LF-Tag to consumer account B.

- Query your Apache Hive metastore with AWS Lake Formation permissions

Managing Lake Formation permissions

Lake Formation provides central access controls for data in your data lake. You can define security policy-based rules for your users and applications by role in Lake Formation, and integration with AWS Identity and Access Management authenticates those users and roles. Once the rules are defined, Lake Formation enforces your access controls at table, column, and ro-level granularity for users of Amazon Redshift Spectrum and Amazon Athena.

Topics

- Granting data location permissions
- Granting permissions on Data Catalog resources
- Permissions example scenario
- Data filtering and cell-level security in Lake Formation
- Row-level security policies in AWS Lake Formation
- Viewing database and table permissions in Lake Formation
- Revoking permission using the Lake Formation console
- Cross-account data sharing in Lake Formation
- Accessing and viewing shared Data Catalog tables and databases
- Creating resource links
- Accessing tables across Regions

Granting data location permissions

Data location permissions in AWS Lake Formation enable principals to create and alter Data Catalog resources that point to designated registered Amazon S3 locations. Data location permissions work in addition to Lake Formation data permissions to secure information in your data lake.

Lake Formation does not use the AWS Resource Access Manager (AWS RAM) service for data location permission grants, so you don't need to accept resource share invitations for data location permissions.

You can grant data location permissions by using the Lake Formation console, API, or AWS Command Line Interface (AWS CLI).

Note

For a grant to succeed, you must first register the data location with Lake Formation.

See Also:

- Underlying data access control

Topics

- Granting data location permissions (same account)
- Granting data location permissions (external account)
- Granting permissions on a data location shared with your account

Granting data location permissions (same account)

Follow these steps to grant data location permissions to principals in your AWS account. You can grant permissions by using the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

AWS Management Console

To grant data location permissions (same account)

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as a data lake administrator or as a principal who has grant permissions on the desired data location.
2. In the navigation pane, under **Permissions**, choose **Data locations**.
3. Choose **Grant**.
4. In the **Grant permissions** dialog box, ensure that the **My account** tile is selected. Then provide the following information:
   - For **IAM users and roles**, choose one or more principals.

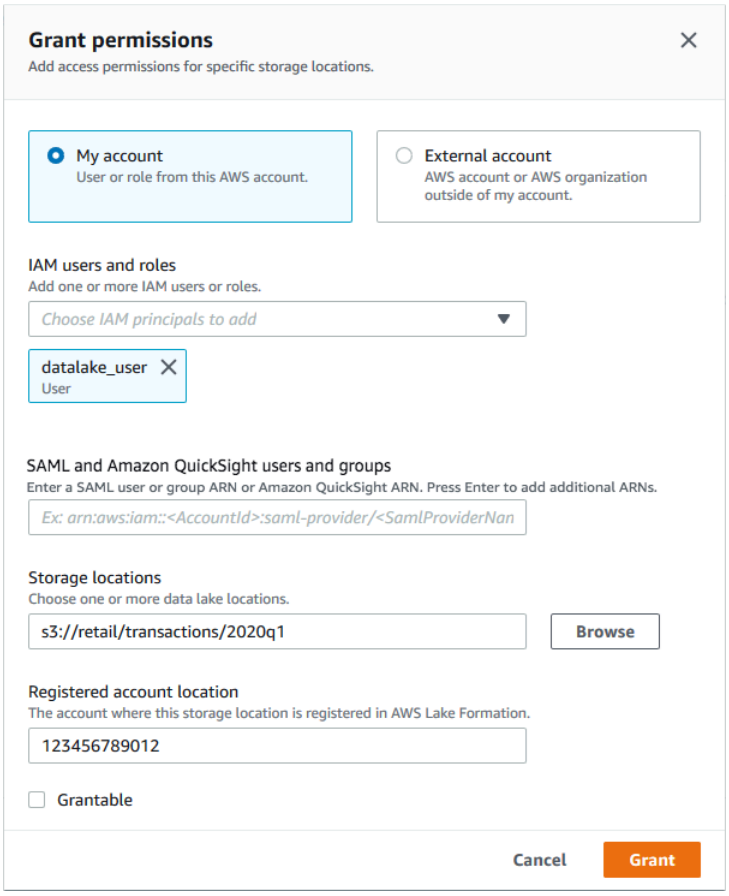
- For **SAML and Amazon Quick users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML or ARNs for Amazon Quick users or groups.

  Enter one ARN at a time, and press **Enter** after each ARN. For information about how to construct the ARNs, see Lake Formation grant and revoke AWS CLI commands.

- For **Storage locations**, choose **Browse**, and choose an Amazon Simple Storage Service (Amazon S3) storage location. The location must be registered with Lake Formation. Choose **Browse** again to add another location. You can also type the location, but ensure that you precede the location with `s3://`.

- For **Registered account location**, enter the AWS account ID where the location is registered. This defaults to your account ID. In a cross-account scenario, data lake administrators in a recipient account can specify the owner account here when granting the data location permission to other principals in the recipient account.

- (Optional) To enable the selected principals to grant data location permissions on the selected location, select **Grantable**.



5. Choose **Grant**.

AWS CLI

To grant data location permissions (same account)

- Run a `grant-permissions` command, and grant `DATA_LOCATION_ACCESS` to the principal, specifying the Amazon S3 path as the resource.

  Example

  The following example grants data location permissions on `s3://retail` to user `datalake_user1`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:user/datalake_user1
 --permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
 {"ResourceArn":"arn:aws:s3:::retail"}}'
```

  Example

  The following example grants data location permissions on `s3://retail` to `ALLIAMPrincipals` group.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333:IAMPrincipals --
permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
 {"ResourceArn":"arn:aws:s3:::retail", "CatalogId": "111122223333"}}'
```

See Also:

- Lake Formation permissions reference

Granting data location permissions (external account)

Follow these steps to grant data location permissions to an external AWS account or organization.

You can grant permissions by using the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Before you begin

Ensure that all cross-account access prerequisites are satisfied. For more information, see Prerequisites.

AWS Management Console

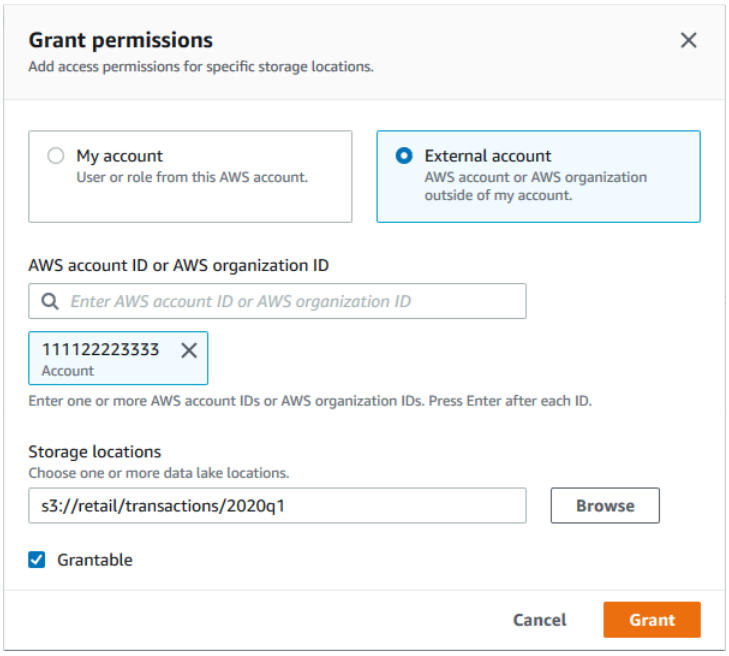
To grant data location permissions (external account, console)

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as a data lake administrator.
2. In the navigation pane, under **Permissions**, choose **Data locations**, and then choose **Grant**.
3. In the **Grant permissions** dialog box, choose the **External account** tile.
4. Provide the following information:
   - For **AWS account ID or AWS organization ID**, enter valid AWS account numbers, organization IDs, or organizational unit IDs.

     Press **Enter** after each ID.

     An organization ID consists of "o-" followed by 10 to 32 lower-case letters or digits.

     An organizational unit ID consists of "ou-" followed by 4 to 32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" (hyphen) and 8 to 32 additional lowercase letters or digits.
   - Under **Storage locations**, choose **Browse**, and choose an Amazon Simple Storage Service (Amazon S3) storage location. The location must be registered with Lake Formation.



5. Select **Grantable**.
6. Choose **Grant**.

AWS CLI

To grant data location permissions (external account, AWS CLI)

- To grant permissions to an external AWS account, enter a command similar to the following.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333  --permissions
 "DATA_LOCATION_ACCESS" --permissions-with-grant-option
 "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
 {"CatalogId":"123456789012","ResourceArn":"arn:aws:s3::retail/
transactions/2020q1"}}'
```

This command grants `DATA_LOCATION_ACCESS` with the grant option to account 1111-2222-3333 on the Amazon S3 location `s3://retail/transactions/2020q1`, which is owned by account 1234-5678-9012.

To grant permissions to an organization, enter a command similar to the following.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:organizations::111122223333:organization/
o-abcdefghijkl --permissions "DATA_LOCATION_ACCESS" --permissions-
with-grant-option "DATA_LOCATION_ACCESS" --resource '{"DataLocation":
 {"CatalogId":"123456789012","ResourceArn":"arn:aws:s3::retail/
transactions/2020q1"}}'
```

This command grants `DATA_LOCATION_ACCESS` with grant option to the organization `o-abcdefghijkl` on the Amazon S3 location `s3://retail/transactions/2020q1`, which is owned by account 1234-5678-9012.

To grant permissions to a principal in an external AWS account, enter a command similar to the following.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
 {"ResourceArn":"arn:aws:s3::retail/transactions/2020q1", "CatalogId":
 "123456789012"}}'
```

This command grants `DATA_LOCATION_ACCESS` to a principal in account 1111-2222-3333 on the Amazon S3 location `s3://retail/transactions/2020q1`, which is owned by account 1234-5678-9012.

Example

The following example grants data location permissions on `s3://retail` to `ALLIAMPrincipals` group in an external account.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333:IAMPrincipals --
permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
 {"ResourceArn":"arn:aws:s3:::retail", "CatalogId": "123456789012"}}'
```

See Also:

- Lake Formation permissions reference

Granting permissions on a data location shared with your account

After a Data Catalog resource is shared with your AWS account, as a data lake administrator, you can grant permissions on the resource to other principals in your account. If the `ALTER` permission is granted on a shared table, and the table points to a registered Amazon S3 location, you must also grant data location permissions on the location. Likewise, if the `CREATE_TABLE` or `ALTER` permission is granted on a shared database and the database has a location property that points to a registered location, you must also grant data location permissions on the location.

To grant data location permissions on a shared location to a principal in your account, your account must have been granted the `DATA_LOCATION_ACCESS` permission on the location with the grant option. When you then grant `DATA_LOCATION_ACCESS` to another principal in your account, you must include the Data Catalog ID (AWS account ID) of the owner account. The owner account is the account that registered the location.

You can use the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI to grant data location permissions.

To grant permissions on a data location shared with your account (console)

- Follow the steps in Granting data location permissions (same account).

  For **Storage locations**, you must type the locations. For **Registered account location**, enter the AWS account ID of the owner account.

To grant permissions on a data location shared with your account (AWS CLI)

- Enter one of the following commands to grant permissions to either a user or a role.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:user/<user-name>
 --permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
 {"CatalogId":"<owner-account-ID>","ResourceArn":"arn:aws:s3:::<s3-location>"}}'
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:role/<role-name>
 --permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
 {"CatalogId":"<owner-account-ID>","ResourceArn":"arn:aws:s3:::<s3-location>"}}'
```

Granting permissions on Data Catalog resources

You can grant **Data permissions** to principals in AWS Lake Formation so that the principals can create and manage Data Catalog resources, and can access underlying data. You can grant **Data lake permissions** on catalogs, databases, tables, and views. When you grant permissions on tables, you can limit access to specific table columns or rows for even more fine-grained access control.

You can grant permissions on individual catalogs, databases, tables and views, or with a single grant operation, you can grant permissions on all databases, tables and views in a catalog or a database. If you grant permissions on all tables in a database to IAM principals, you are implicitly granting the `DESCRIBE` permission on the database. The database then appears on the **Databases** page on the console, and is returned by the `GetDatabases` API operation. The same principle applies at the catalog level - when you receive permissions for databases within a catalog, you also get `DESCRIBE` permissions for that catalog.

Important

The implicit `DESCRIBE` permission applies only when granting permissions to IAM principals within the same AWS account. For cross-account resources, you must explicitly grant `DESCRIBE` permissions. The automatic `DESCRIBE` permission grant doesn't apply when using attribute-based access control (ABAC). When granting permissions on all tables in a database using attributes, Lake Formation doesn't implicitly grant `DESCRIBE` permission to the database.

You can grant permissions by using either the named resource method or the Lake Formation tag-based access control (LF-TBAC) method.

You can grant permissions to principals in the same AWS account or to external accounts or organizations. When you grant to external accounts or organizations, you are sharing Data Catalog objects that you own with those accounts or organizations. Principals in those accounts or organizations can then access Data Catalog objects that you own and the underlying data.

Note

Currently, the LF-TBAC method supports granting cross-account permissions to IAM principals, AWS accounts, organizations, and organizational units (OUs).

When you grant permissions to external accounts or organizations, you must include the grant option. Only the data lake administrator in the external account can access the shared objects until the administrator grants permissions on the shared objects to other principals in the external account.

You can grant Data Catalog permissions by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Note

When you delete a Data Catalog object, all permissions that are associated with the object become invalid. Recreating the same resource with the same name, will not recover Lake Formation permissions. Users will have to setup new permissions again.

See also:

- Sharing Data Catalog tables and databases across AWS Accounts
- Metadata access control
- Lake Formation permissions reference

IAM permissions required to grant or revoke Lake Formation permissions

All principals, including the data lake administrator, need the following AWS Identity and Access Management (IAM) permissions to grant or revoke AWS Lake Formation Data Catalog permissions or data location permissions with the Lake Formation API or the AWS CLI:

- `lakeformation:GrantPermissions`
- `lakeformation:BatchGrantPermissions`
- `lakeformation:RevokePermissions`
- `lakeformation:BatchRevokePermissions`
- `glue:GetTable`, `glue:GetDatabase`, or `glue:GetCatalog` for a table, database, or catalog that you're granting permissions using the named resource method.

> **Note**
>
> Data lake administrators have implicit Lake Formation permissions to grant and revoke Lake Formation permissions. But they still need the IAM permissions on the Lake Formation grant and revoke API operations.
> IAM roles with `AWSLakeFormationDataAdmin` AWS managed policy cannot add new data lake administrators because this policy contains an explicit deny for the Lake Formation API operation, `PutDataLakeSetting`.

The following IAM policy is recommended for principals who are not data lake administrators and who want to grant or revoke permissions using the Lake Formation console.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:ListPermissions",
                "lakeformation:GrantPermissions",
                "lakeformation:BatchGrantPermissions",
                "lakeformation:RevokePermissions",
                "lakeformation:BatchRevokePermissions",
                "glue:GetCatalogs",
                "glue:GetDatabases",
                "glue:SearchTables",
                "glue:GetTables",
                "glue:GetCatalog",
                "glue:GetDatabase",
                "glue:GetTable",
                "iam:ListUsers",
                "iam:ListRoles",
                "sso-directory:DescribeUser",
                "sso-directory:DescribeGroup",
                "sso:DescribeInstance"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

All of the `glue:` and `iam:` permissions in this policy are available in the AWS managed policy `AWSGlueConsoleFullAccess`.

To grant permissions by using Lake Formation tag-based access control (LF-TBAC), principals need additional IAM permissions. For more information, see Lake Formation tag-based access control best practices and considerations and Lake Formation personas and IAM permissions reference.

Cross-account permissions

Users who want to grant cross-account Lake Formation permissions by using the named resource method must also have the permissions in the `AWSLakeFormationCrossAccountManager` AWS managed policy.

Data lake administrators need those same permissions for granting cross-account permissions, plus the AWS Resource Access Manager (AWS RAM) permission to enable granting permissions to organizations. For more information, see Data lake administrator permissions.

The administrative user

A principal with administrative permissions—for example, with the `AdministratorAccess` AWS managed policy—has permissions to grant Lake Formation permissions and create data lake administrators. To deny a user or role access to Lake Formation administrator operations, attach or add into its policy a `Deny` statement for administrator API operations.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lakeformation:GetDataLakeSettings",
                "lakeformation:PutDataLakeSettings"
            ],
            "Effect": "Deny",
            "Resource": [
                "*"
            ]
```

```
        }
    ]
}
```

> **Important**
>
> To prevent users from adding themselves as an administrator with an extract, transform, and load (ETL) script, make sure that all non-administrator users and roles are denied access to these API operations. The `AWSLakeFormationDataAdmin` AWS managed policy contains an explict deny for the Lake Formation API operation, `PutDataLakeSetting` that prevents users from adding new data lake administrators.

Granting data permissions using the named resource method

The named Data Catalog resource method is a way of granting permissions to AWS Glue Data Catalog objects, such as catalogs, databases, tables, columns, and views, using a centralized approach. It allows you to define resource-based policies that control access to specific resources within your data lake.

When you use the named resource method to grant permissions, you can specify the resource type and the permissions that you want to grant or revoke for that resource. You can also revoke the permission later if needed, thereby removing the permissions from the associated resources.

You can grant permissions by using the AWS Lake Formation console, APIs, or the AWS Command Line Interface (AWS CLI).

Topics

- Granting catalog permissions using the named resource method
- Granting database permissions using the named resource method
- Granting table permissions using the named resource method
- Granting permissions on views using the named resource method

Granting catalog permissions using the named resource method

The following steps explain how to grant catalog permissions by using the named resource method.

Console

Use the **Grant permissions** page on the Lake Formation console. The page is divided into the following sections:

- **Principal type** – You can grant permissions to specific principals or use attribute tags.
  - **Principals** – The IAM users, roles, IAM Identity Center users and groups, SAML users and groups, AWS accounts, organizations, or organizational units to grant permissions.

    Principal by attributes – Add tag key-value pairs from IAMroles or IAM session tags. Principals with matching attributes receive access to the specified resource.
  - **LF-Tags or catalog resources** – The catalogs, databases, tables, views, or resource links to grant permissions on.
  - **Permissions** – The Lake Formation permissions to grant.

> **Note**
>
> To grant permissions on a database resource link, see Granting resource link permissions.

1. Open the **Grant permissions** page.

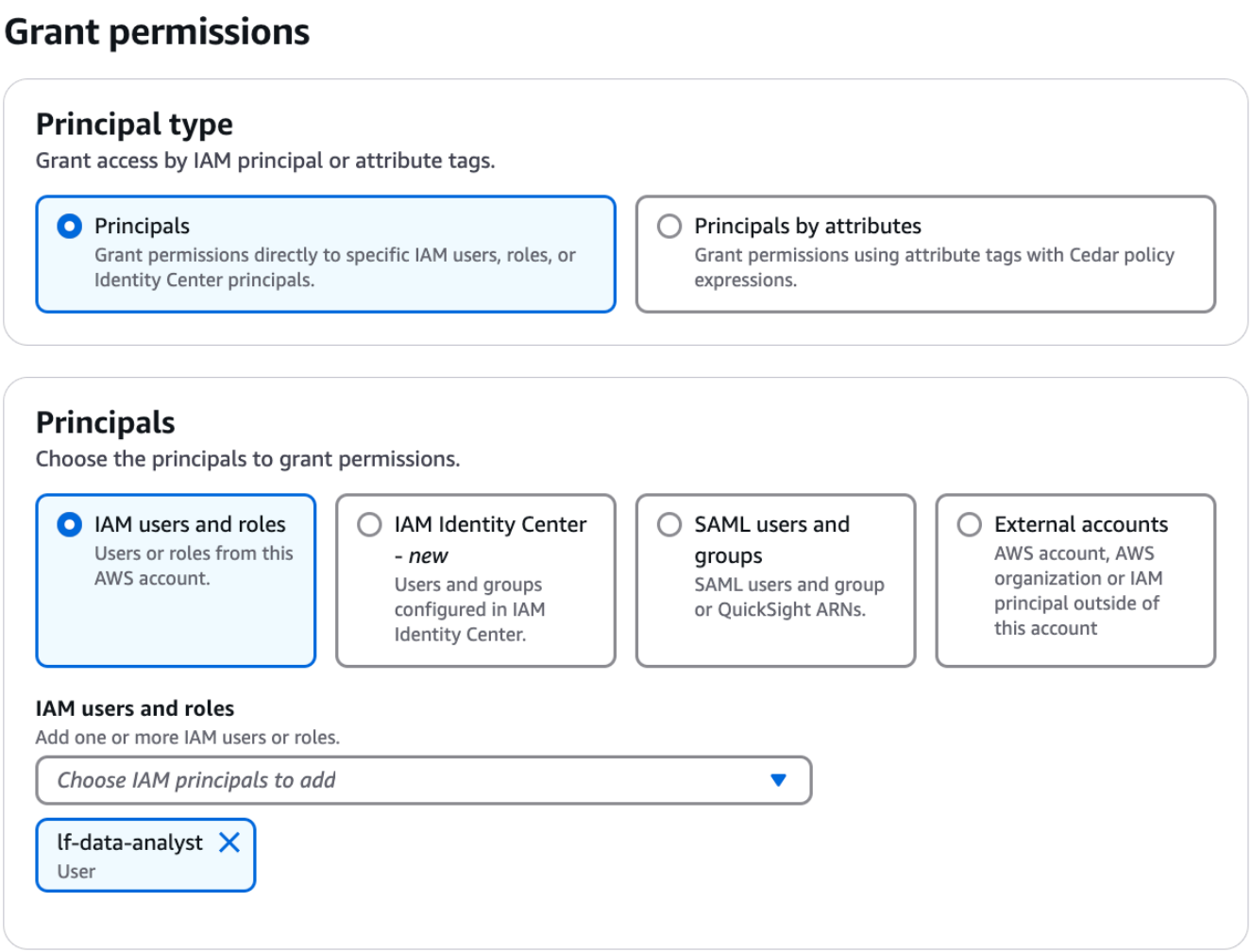
   Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator, the catalog creator, or an IAM user who has **Grantable permissions** on the catalog.

   Do one of the following:

   - In the navigation pane, under **Permissions**, choose **Data permissions**. Then choose **Grant**.
   - In the navigation pane, choose **Catalogs** under **Data Catalog**. Then, on the **Catalogs** page, choose a catalog, and from the **Actions** menu, under **Permissions**, choose **Grant**.

> **Note**
>
> You can grant permissions on a catalog through its resource link. To do so, on the **Catalogs** page, choose a catalog link container, and on the **Actions** menu, choose **Grant on target**. For more information, see How resource links work in Lake Formation.

2. Next, in the **Principal type** section, choose principals or specify attributes attached to the principals.



Specify principals

IAM users and roles

Choose one or more users or roles from the **IAM users and roles** list.

IAM Identity Center

Choose one or more users or groups from the **Users and groups** list. Select **Add** to add more users or groups.

SAML users and groups

For **SAML and Quick users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon Quick users or groups. Press Enter after each ARN.

For information about how to construct the ARNs, see Lake Formation grant and revoke AWS CLI commands.

> **Note**
>
> Lake Formation integration with Quick is supported only for Quick Enterprise Edition.

External accounts

For **AWS account, AWS organization**, or **IAM Principal** enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or ARN for the IAM user or role. Press **Enter** after each ID.

An organization ID consists of "o-" followed by 10–32 lower-case letters or digits.

An organizational unit ID starts with "ou-" followed by 4–32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

Principals by attributes

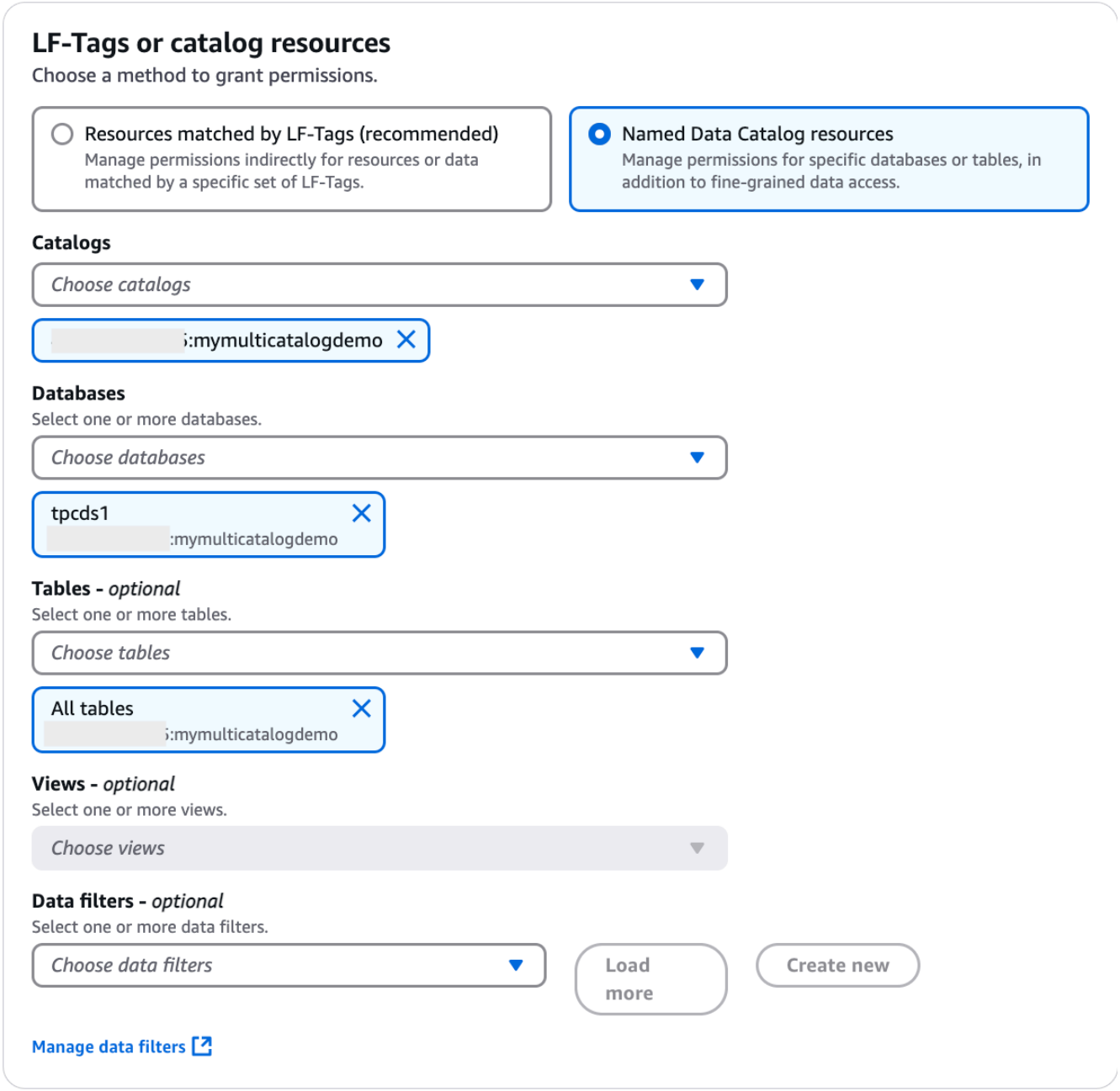
Attributes

Add the IAM tag key-value pairs from the IAM role.

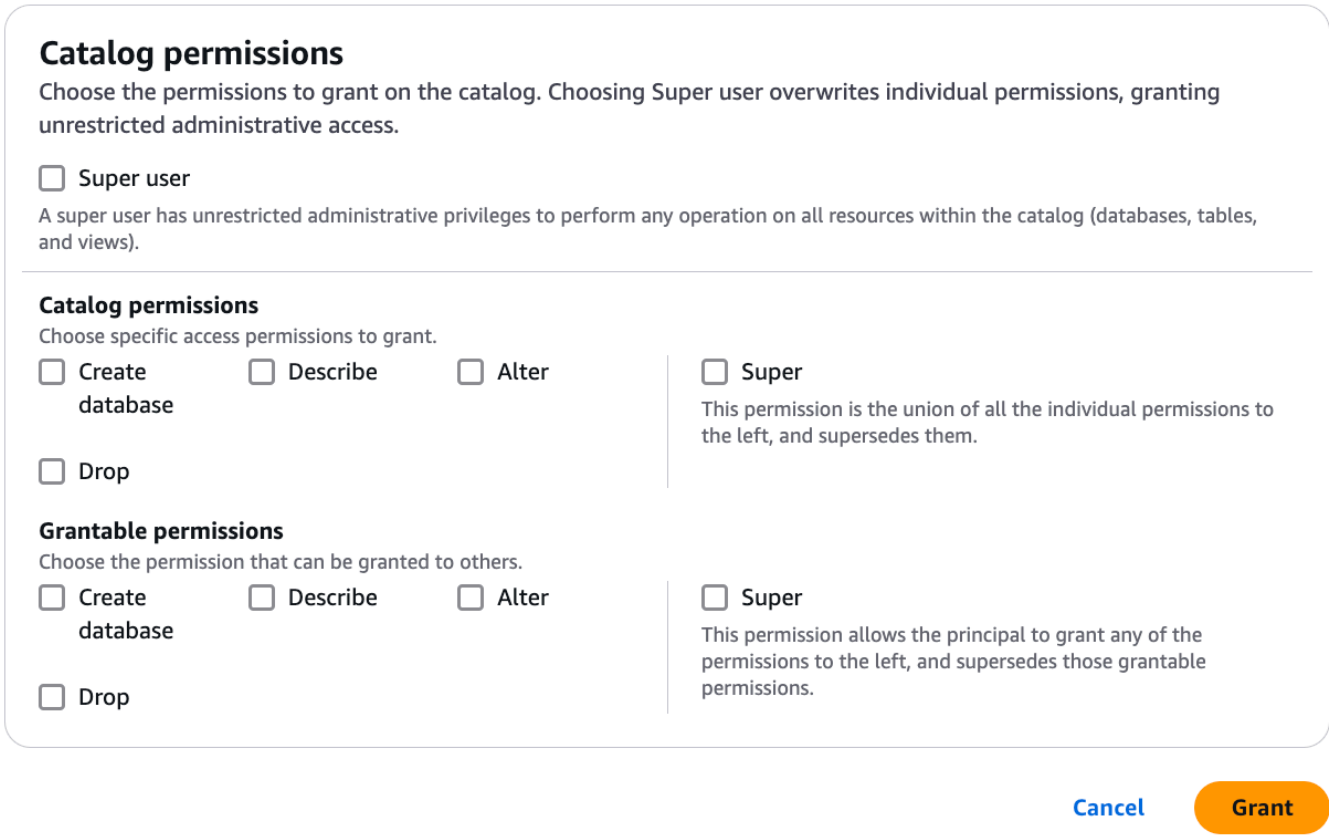
Permission scope

Specify if you're granting permissions to principals with matching attributes in the same account or in another account.

3. In the **LF-Tags or catalog resources** section, choose **Named data catalog resources**.



4. Choose one or more catalogs from the **Catalogs** list. You can also choose one or more **Databases**, **Tables**, and/or **Data filters**.
5. In the **Catalog permissions** section, select permissions and grantable permissions. Under **Catalog permissions**, select one or more permissions to grant.



Choose **Super user** to grant unrestricted administrative privileges to perform any operation on all resources within the catalog (databases, tables, and views).

> **Note**
>
> After granting `Create database` or `Alter` on a catalog that has a location property that points to a registered location, be sure to also grant data location permissions on the location to the principals. For more information, see Granting data location permissions.

6. (Optional) Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account. This option is not supported when you are granting permissions to an IAM principal from an external account.
7. Choose **Grant**.

   The **Data permissions** page shows the permission details. If you used **Principals by attribute** option to grant permissions, you can view the permission grant to `ALLPrincipals` in the list.

AWS CLI

For granting catalog permissions using AWS CLI, see Creating Amazon Redshift federated catalogs.

Granting database permissions using the named resource method

The following steps explain how to grant database permissions by using the named resource method.

Console

Use the **Grant permissions** page on the Lake Formation console. The page is divided into the following sections:

- **Principal type** – The **Principals** section include the IAM users, roles, IAM Identity Center users and groups, SAML users and groups, AWS accounts, organizations, or organizational units to grant permissions. In the **Principals by attributes** section, you can specify the key and values for the attributes attached to the IAM roles.
- **LF-Tags or catalog resources** – The databases, tables, views, or resource links to grant permissions on.
- **Permissions** – The Lake Formation permissions to grant.

> **Note**
>
> To grant permissions on a database resource link, see Granting resource link permissions.

1. Open the **Grant permissions** page.

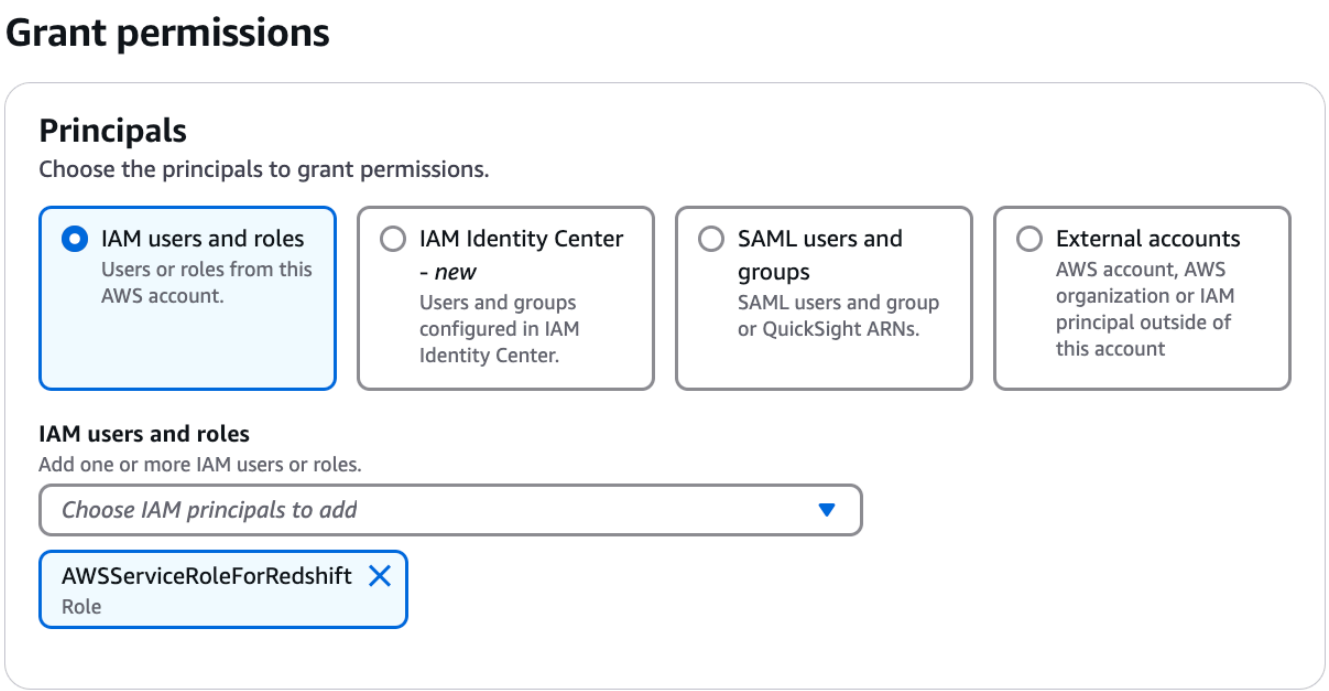
   Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator, the database creator, or an IAM user who has **Grantable permissions** on the database.

   Do one of the following:

- In the navigation pane, under **Permissions**, choose **Data permissions**. Then choose **Grant**.
- In the navigation pane, choose **Databases** under **Data Catalog**. Then, on the **Databases** page, choose a database, and from the **Actions** menu, under **Permissions**, choose **Grant**.

> **Note**
>
> You can grant permissions on a database through its resource link. To do so, on the **Databases** page, choose a resource link, and on the **Actions** menu, choose **Grant on target**. For more information, see How resource links work in Lake Formation.

2. In the **Principal type** section, specify principals or grant permissions to principals using attributes.



IAM users and roles

Choose one or more users or roles from the **IAM users and roles** list.

IAM Identity Center

Choose one or more users or groups from the **Users and groups** list. Select **Add** to add more users or groups.

SAML users and groups

For **SAML and Quick users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon Quick users or groups. Press Enter after each ARN.

For information about how to construct the ARNs, see Lake Formation grant and revoke AWS CLI commands.

> **Note**
>
> Lake Formation integration with Quick is supported only for Quick Enterprise Edition.

External accounts

For **AWS account, AWS organization**, or **IAM Principal** enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or ARN for the IAM user or role. Press **Enter** after each ID.

An organization ID consists of "o-" followed by 10–32 lower-case letters or digits.

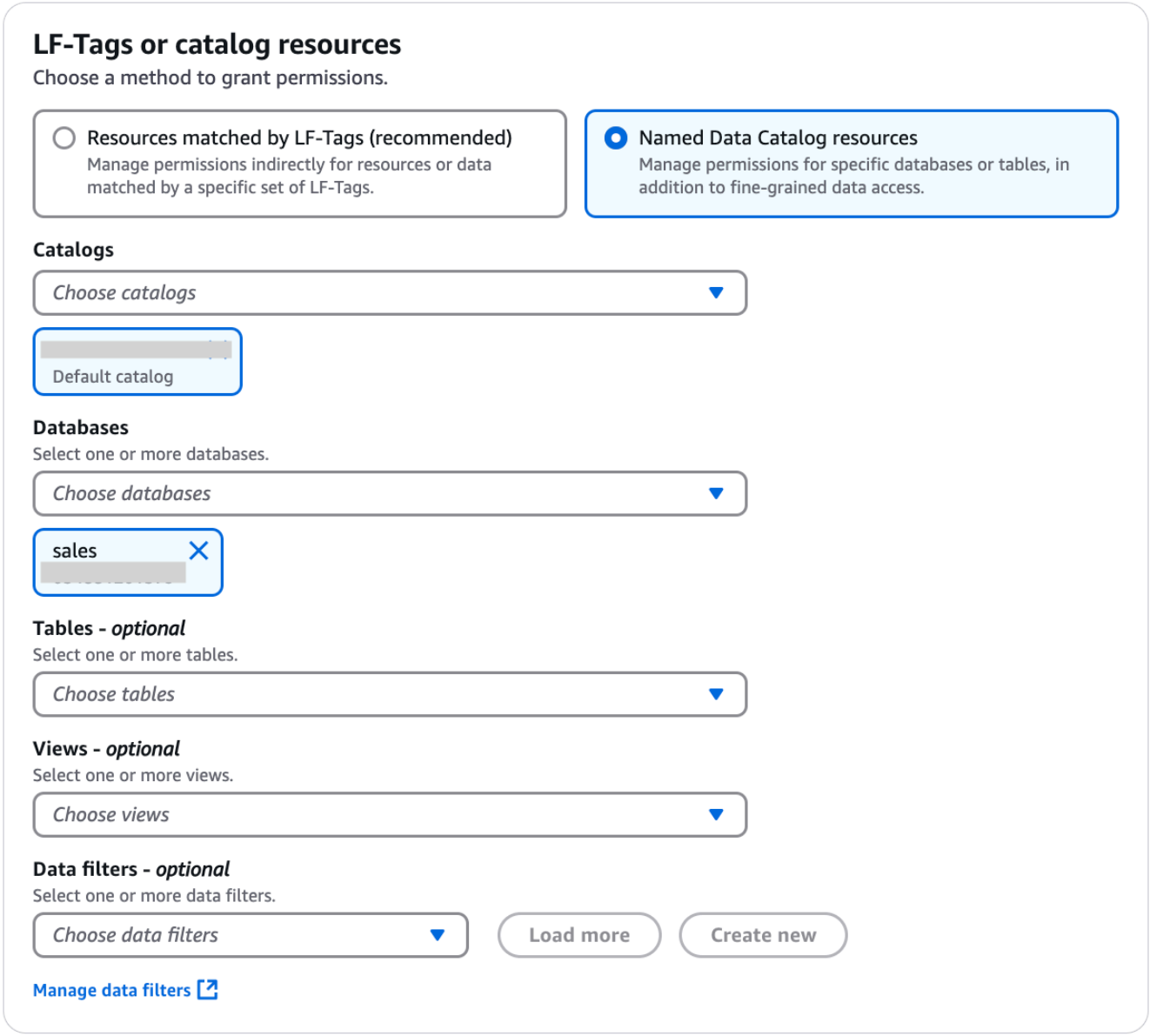
An organizational unit ID starts with "ou-" followed by 4–32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

Principals by attributes

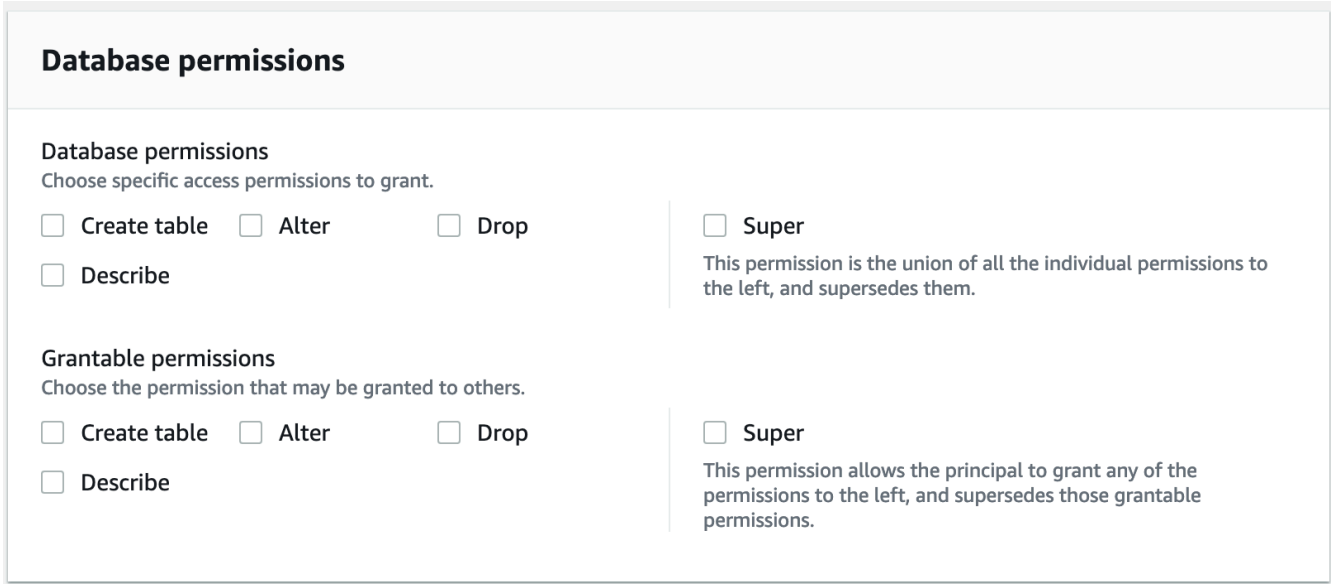
Specify the attribute key and value(s). If you choose more than one value, you are creating an attribute expression with an OR operator. This means that if any of the attribute tag values assigned to an IAM role or user match, the role/user gains access permissions on the resource.

Choose the permission scope by specifying if you're granting permissions to principals with matching attributes in the same account or in another account.

3. In the **LF-Tags or catalog resources** section, choose **Named data catalog resources**.



4. Choose one or more databases from the **Database** list. You can also choose one or more **Tables** and/or **Data filters**.
5. In the **Permissions** section, select permissions and grantable permissions. Under **Database permissions**, select one or more permissions to grant.



> **Note**
>
> After granting `Create Table` or `Alter` on a database that has a location property that points to a registered location, be sure to also grant data location permissions on the location to the principals. For more information, see Granting data location permissions.

6. (Optional) Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account. This option is not supported when you are granting permissions to an IAM principal from an external account.
7. Choose **Grant**.

AWS CLI

You can grant database permissions by using the named resource method and the AWS Command Line Interface (AWS CLI).

To grant database permissions using the AWS CLI

- Run a `grant-permissions` command, and specify a database or the Data Catalog as the resource, depending on the permission being granted.

  In the following examples, replace *<account-id>* with a valid AWS account ID.

Example– Grant to create a database

This example grants CREATE_DATABASE to user datalake_user1. Because the resource on which this permission is granted is the Data Catalog, the command specifies an empty CatalogResource structure as the resource parameter.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:user/datalake_user1 --
permissions "CREATE_DATABASE" --resource '{ "Catalog": {}}'
```

Example– Grant to create tables in a designated database

The next example grants CREATE_TABLE on the database retail to user datalake_user1.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:user/datalake_user1 --
permissions "CREATE_TABLE" --resource '{ "Database": {"Name":"retail"}}'
```

Example– Grant to an external AWS account with the Grant option

The next example grants CREATE_TABLE with the grant option on the database retail to external account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333 --permissions "CREATE_TABLE"
 --permissions-with-grant-option "CREATE_TABLE" --resource '{ "Database":
 {"Name":"retail"}}'
```

Example– Grant to an organization

The next example grants ALTER with the grant option on the database issues to the organization o-abcdefghijkl.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:organizations::111122223333:organization/
o-abcdefghijkl --permissions "ALTER" --permissions-with-grant-option "ALTER" --
resource '{ "Database": {"Name":"issues"}}'
```

Example- Grant to ALLIAMPrincipals in the same account

The next example grants CREATE_TABLE permission on the database `retail` to all principals in the same account. This option enables every principal in the account to create a table in the database and create a table resource link allowing integrated query engines to access shared databases and tables. This option is especially useful when a principal receives a cross-account grant, and does not have the permission to create resource links. In this scenario, the data lake administrator can create a placeholder database and grant CREATE_TABLE permission to the `ALLIAMPrincipal` group, enabling every IAM principal in the account to create resource links in the placeholder database.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333:IAMPrincipals
 --permissions "CREATE_TABLE"  --resource '{ "Database":
 {"Name":"temp","CatalogId":"111122223333"}}'
```

Example- Grant to ALLIAMPrincipals in an external account

The next example grants CREATE_TABLE on the database `retail` to all principals in an external account. This option enables every principal in the account to create a table in the database.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333:IAMPrincipals
 --permissions "CREATE_TABLE"  --resource '{ "Database":
 {"Name":"retail","CatalogId":"123456789012"}}'
```

Note

After granting CREATE_TABLE or ALTER on a database that has a location property that points to a registered location, be sure to also grant data location permissions on the location to the principals. For more information, see Granting data location permissions.

See also

- Lake Formation permissions reference
- Granting permissions on a database or table shared with your account
- Accessing and viewing shared Data Catalog tables and databases

Granting table permissions using the named resource method

You can use the Lake Formation console or AWS CLI to grant Lake Formation permissions on Data Catalog tables. You can grant permissions on individual tables, or with a single grant operation, you can grant permissions on all tables in a database.

If you grant permissions on all tables in a database, you are implicitly granting the `DESCRIBE` permission on the database. The database then appears on the **Databases** page on the console, and is returned by the `GetDatabases` API operation. This automatic `DESCRIBE` permission grant doesn't apply when using attribute-based access control (ABAC). When granting permissions on all tables in a database using attributes, Lake Formation doesn't implicitly grant `DESCRIBE` permission to the database.

When you choose `SELECT` as the permission to grant, you have the option to apply a column filter, row filter, or cell filter.

Console

The following steps explain how to grant table permissions by using the named resource method and the **Grant data lake permissions** page on the Lake Formation console. The page is divided into these sections:

- **Principals types** – The users, roles, AWS accounts, organizations, or organizational units to grant permissions to. You can also grant permissions to principals with matching attributes.
- **LF-Tags or catalog resources** – The databases, tables, or resource links to grant permissions on.
- **Permissions** – The Lake Formation permissions to grant.

> **Note**
>
> To grant permissions on a table resource link, see Granting resource link permissions.

1. Open the Grant permissions page.

   Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator, the table creator, or a user who has been granted permissions on the table with the grant option.

   Do one of the following:

   - In the navigation pane, choose **Data permissions** under **Permissions**. Then choose **Grant**.
   - In the navigation pane, choose **Tables**. Then, on the **Tables** page, choose a table, and on the **Actions** menu, under **Permissions**, choose **Grant**.

   > **Note**
   >
   > You can grant permissions on a table through its resource link. To do so, on the **Tables** page, choose a resource link, and on the **Actions** menu, choose **Grant on target**. For more information, see How resource links work in Lake Formation.

2. Next, in the **Principal types** section, specify principals or principals with matching attrubutes to grant permissions.

   IAM users and roles

   Choose one or more users or roles from the **IAM users and roles** list.

   IAM Identity Center

   Choose one or more users or groups from the **Users and groups** list.

   SAML users and groups

   For **SAML and Quick users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Quick users or groups. Press Enter after each ARN.

For information about how to construct the ARNs, see Lake Formation grant and revoke AWS CLI commands.

> **Note**
>
> Lake Formation integration with Quick is supported for Quick Enterprise Edition only.

External accounts

For **AWS account , AWS organization**, or **IAM Principal** enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or the ARN for the IAM user or role. Press **Enter** after each ID.

An organization ID consists of "o-" followed by 10–32 lower-case letters or digits.

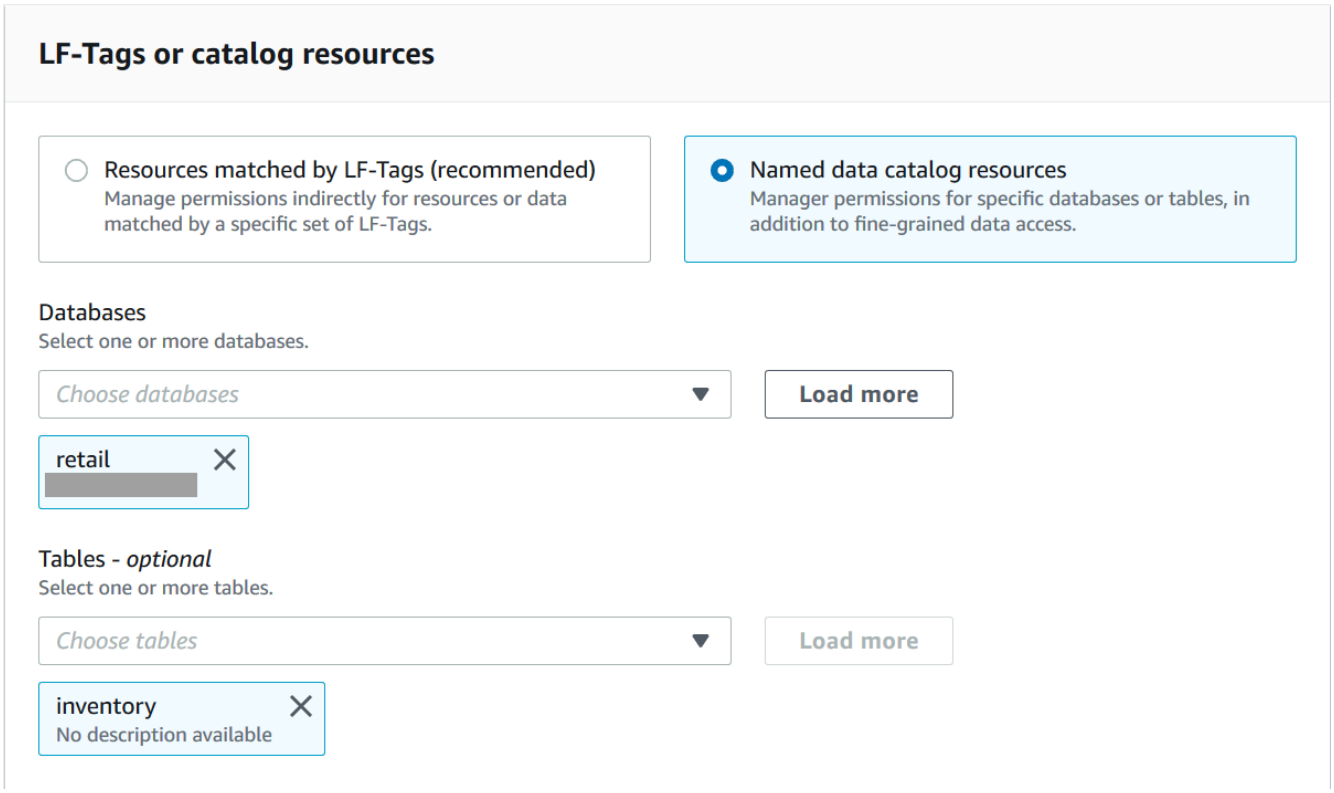
An organizational unit ID starts with "ou-" followed by 4–32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" character and 8 to 32 additional lowercase letters or digits.

Principals by attributes

Specify the attribute key and value(s). If you choose more than one value, you are creating an attribute expression with an OR operator. This means that if any of the attribute tag values assigned to an IAM role or user match, the role/user gains access permissions on the resource

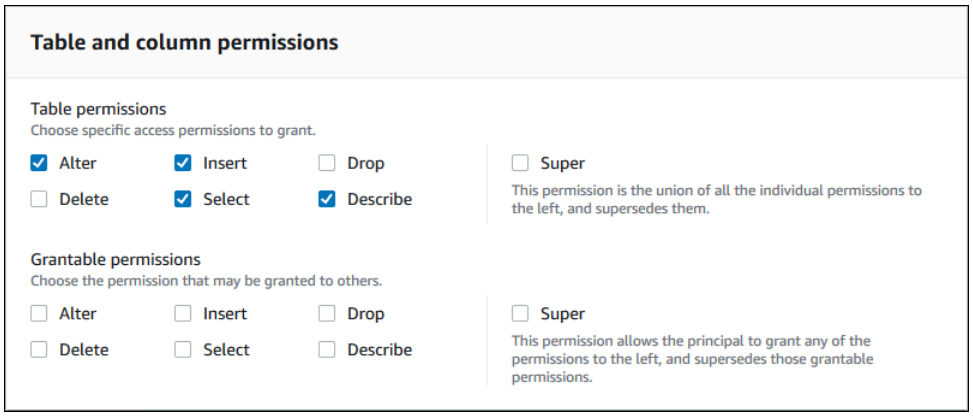
Choose the permission scope by specifying if you're granting permissions to principals with matching attributes in the same account or in another account.

3. In the **LF-Tags or catalog resources** section, choose a database. Then choose one or more tables, or **All tables**.

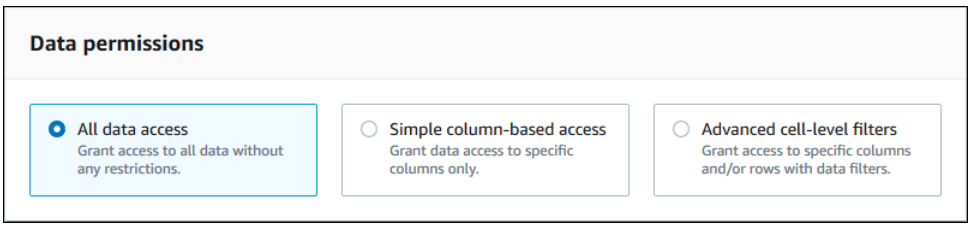


4. **Specify the permissions with no data filtering.**

   In the **Permissions** section, select the table permissions to grant, and optionally select grantable permissions.



   If you grant **Select**, the **Data permissions** section appears beneath the **Table and column permissions** section, with the **All data access** option selected by default. Accept the default.



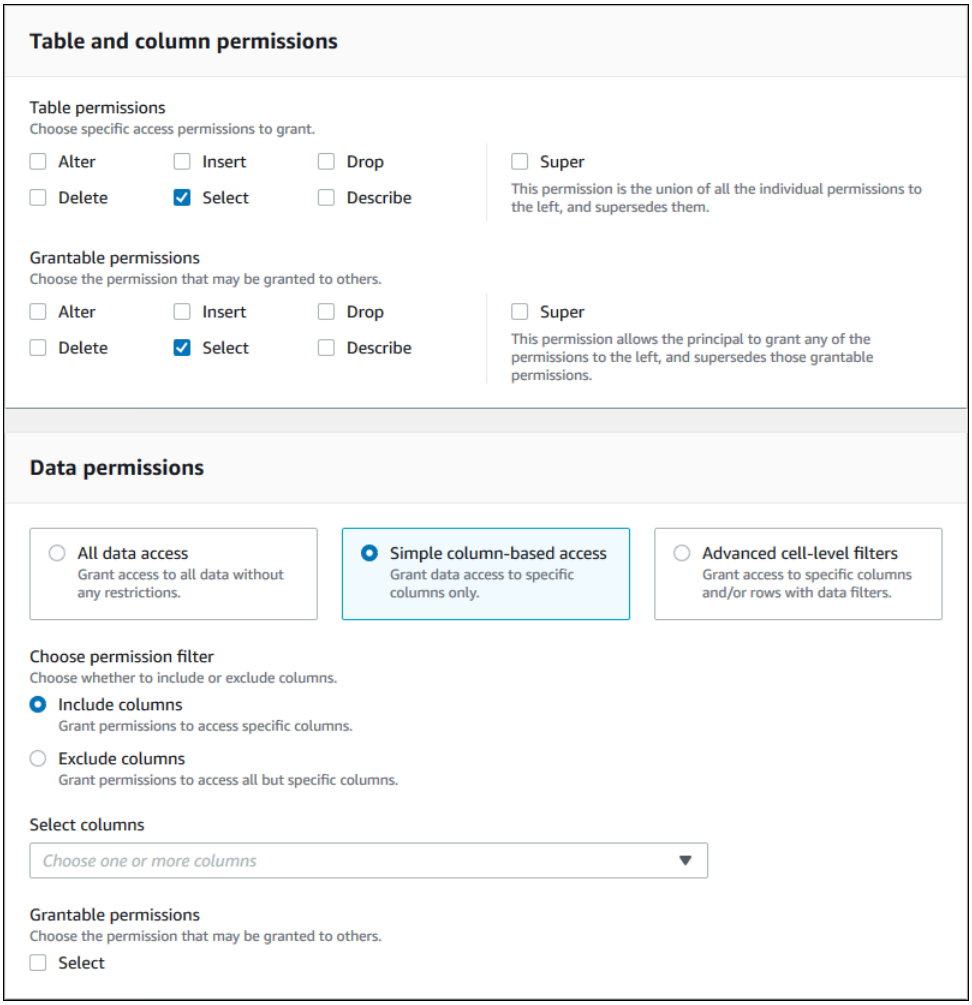
5. Choose **Grant**.
6. **Specify the Select permission with data filtering**

   Select the **Select** permission. Don't select any other permissions.

   The **Data permissions** section appears beneath the **Table and column permissions** section.
7. Do one of the following:

   - Apply simple column filtering only.

     1. Choose **Simple column-based access**.



2. Choose whether to include or exclude columns, and then choose the columns to include or exclude.

   Only include lists are supported when granting permissions to an external AWS account or organization.

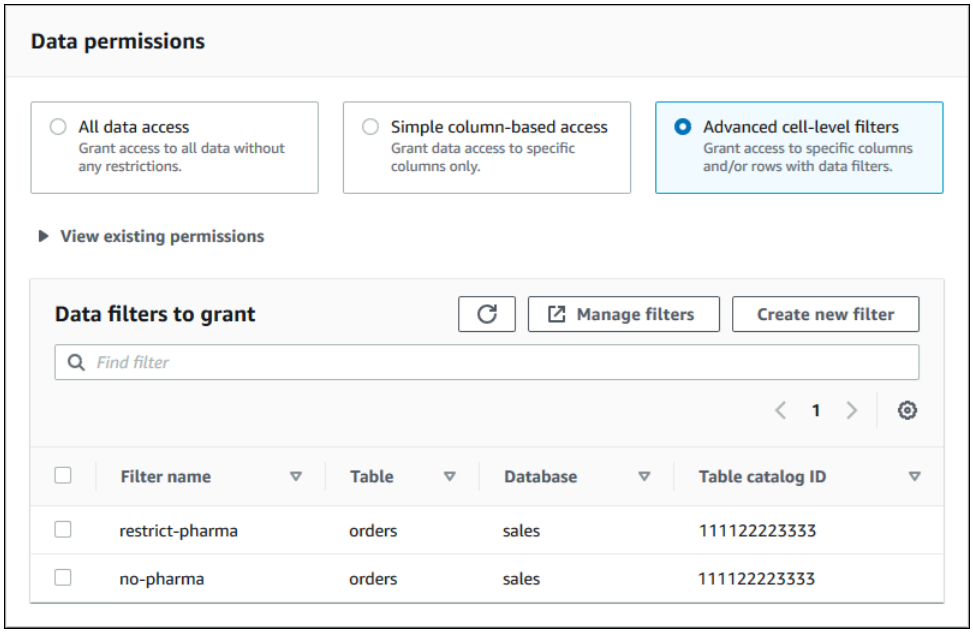
3. (Optional) Under **Grantable permissions**, turn on the grant option for the Select permission.

   If you include the grant option, the grant recipient can grant permissions only on the columns that you grant to them.

> **Note**
>
> You can also apply column filtering only by creating a data filter that specifies a column filter and specifies all rows as the row filter. However, this requires more steps.

- Apply column, row, or cell filtering.

  1. Choose **Advanced cell-level filters**.



  2. (Optional) Expand **View existing permissions**.
  3. (Optional) Choose **Create new filter**.
  4. (Optional) To view details for the listed filters, or to create new or delete existing filters, choose **Manage filters**.

The **Data filters** page opens in a new browser window.

When you are finished on the **Data filters** page, return to the **Grant permissions** page, and if necessary, refresh the page to view any new data filters that you created.

5. Select one or more data filters to apply to the grant.

> **Note**
>
> If there are no data filters in the list, it means that no data filters were created for the selected table.

8. Choose **Grant**.

AWS CLI

You can grant table permissions by using the named resource method and the AWS Command Line Interface (AWS CLI).

To grant table permissions using the AWS CLI

- Run a `grant-permissions` command, and specify a table as the resource.

Example– Grant on a single table - no filtering

The following example grants SELECT and ALTER to user `datalake_user1` in AWS account 1111-2222-3333 on the table `inventory` in the database `retail`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "SELECT" "ALTER" --resource '{ "Table": {"DatabaseName":"retail",
 "Name":"inventory"}}'
```

> **Note**
>
> If you grant the ALTER permission on a table that has its underlying data in a registered location, be sure to also grant data location permissions on the location to the principals. For more information, see Granting data location permissions.

Example– Grant on All Tables with the Grant option - no filtering

The next example grants SELECT with the grant option on all tables in database `retail`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "SELECT" --permissions-with-grant-option "SELECT" --resource '{ "Table":
 { "DatabaseName": "retail", "TableWildcard": {} } }'
```

Example– Grant with simple column filtering

This next example grants SELECT on a subset of columns in the table `persons`. It uses simple column filtering.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "SELECT" --resource '{ "TableWithColumns": {"DatabaseName":"hr",
 "Name":"persons", "ColumnNames":["family_name", "given_name", "gender"]}}'
```

Example– Grant with a data filter

This example grants SELECT on the `orders` table and applies the `restrict-pharma` data filter.

```
aws lakeformation grant-permissions --cli-input-json file://grant-params.json
```

The following are the contents of file `grant-params.json`.

```
{
    "Principal": {"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/
datalake_user1"},
    "Resource": {
        "DataCellsFilter": {
            "TableCatalogId": "111122223333",
            "DatabaseName": "sales",
            "TableName": "orders",
            "Name": "restrict-pharma"
        }
    },
```

```
    "Permissions": ["SELECT"],
    "PermissionsWithGrantOption": ["SELECT"]
}
```

See also

- Overview of Lake Formation permissions
- Data filtering and cell-level security in Lake Formation
- Lake Formation personas and IAM permissions reference
- Granting resource link permissions
- Accessing and viewing shared Data Catalog tables and databases

Granting permissions on views using the named resource method

The following steps explain how to grant permissions on views by using the named resource method and the **Grant permissions** page. The page is divided into the following sections:

- **Principal types** – The IAM users, roles, IAM Identity Center users and groups, AWS accounts, organizations, or organizational units to grant permissions. You can also grant permissions to principals with matching attributes.
- **LF-Tags or catalog resources** – The databases, tables, views, or resource links to grant permissions on.
- **Permissions** – The data lake permissions to grant.

Open the Grant permissions page

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator, the database creator, or an IAM user who has **Grantable permissions** on the database.
2. Do one of the following:
   - In the navigation pane, under **Permissions**, choose **Data permissions**. Then choose **Grant**.
   - In the navigation pane, choose **Views** under **Data Catalog**. Then, on the **Views** page, choose a view, and from the **Actions** menu, under **Permissions**, choose **Grant**.

> **Note**
>
> You can grant permissions on a view through its resource link. To do so, on the **Views** page, choose a resource link, and on the **Actions** menu, choose **Grant on target**. For more information, see How resource links work in Lake Formation.

Specify the principal types

In the **Principal types** section, either choose Principals or Principals by attributes. If you choose Principals, the following options are available:

IAM users and roles

Choose one or more users or roles from the **IAM users and roles** list.

IAM Identity Center

Choose one or more users or groups from the **Users and groups** list.

SAML users and groups

For **SAML and Quick users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon Quick users or groups. Press Enter after each ARN.

For information about how to construct the ARNs, see Lake Formation grant and revoke AWS CLI commands.

> **Note**
>
> Lake Formation integration with Quick is supported only for Quick Enterprise Edition.

External accounts

For **AWS account, AWS organization**, or **IAM Principal** enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or ARN for the IAM user or role. Press **Enter** after each ID.

An organization ID consists of "o-" followed by 10–32 lower-case letters or digits.

An organizational unit ID starts with "ou-" followed by 4–32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

> **See Also**
>
> - Accessing and viewing shared Data Catalog tables and databases

Principals by attributes

Specify the attribute key and value(s). If you choose more than one value, you are creating an attribute expression with an OR operator. This means that if any of the attribute tag values assigned to an IAM role or user match, the role/user gains access permissions on the resource

Choose the permission scope by specifying if you're granting permissions to principals with matching attributes in the same account or in another account.

Specify the views

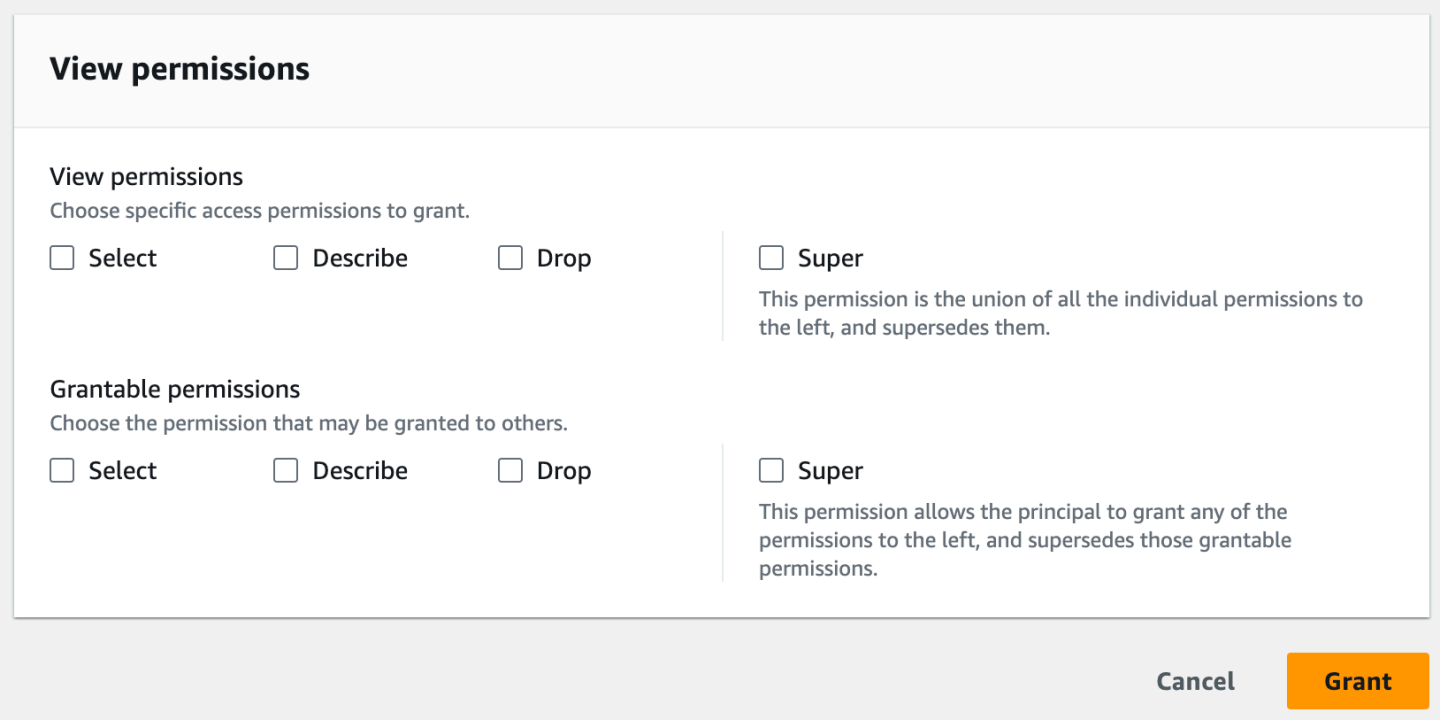
In the **LF-Tags or catalog resources** section, choose one or more views to grant permissions on.

1. Choose **Named data catalog resources**.
2. Choose one or more views from the **Views** list. You can also choose one or more catalogs, databases, tables, and/or data filters.

   Granting data lake permissions to `All tables` within a database will result in the grantee having permissions on all tables and views within the database.

Specify the permissions

In the **Permissions** section, select permissions and grantable permissions.



1. Under **View permissions**, select one or more permissions to grant.
2. (Optional) Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account. This option is not supported when you are granting permissions to an IAM principal from an external account.
3. Choose **Grant**.

See Also

- Lake Formation permissions reference
- Granting permissions on a database or table shared with your account

Lake Formation tag-based access control

Lake Formation tag-based access control (LF-TBAC) is an authorization strategy that defines permissions based on attributes. In Lake Formation, these attributes are called *LF-Tags*. You can attach LF-Tags to Data Catalog resources, and grant permissions to Lake Formation principals on those resources using these LF-Tags. Lake Formation allows operations on those resources when the principal has granted access to a tag value that matches the resource tag value.

LF-TBAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

LF-TBAC is the recommended method to use to grant Lake Formation permissions when there is a large number of Data Catalog objects including federated catalogs, databases, tables, and views. Lake Formation supports tag-based access control for federated catalogs of Amazon S3 tables, Amazon Redshift data warehouses, and federated data sources such as Amazon DynamoDB, SQL Server, and Snowflake.

> **Note**
>
> IAM tags are not the same as LF-Tags. These tags are not interchangeable. LF-Tags are used to grant Lake Formation permissions and IAM tags are used to define IAM policies.

How Lake Formation tag-based access control works

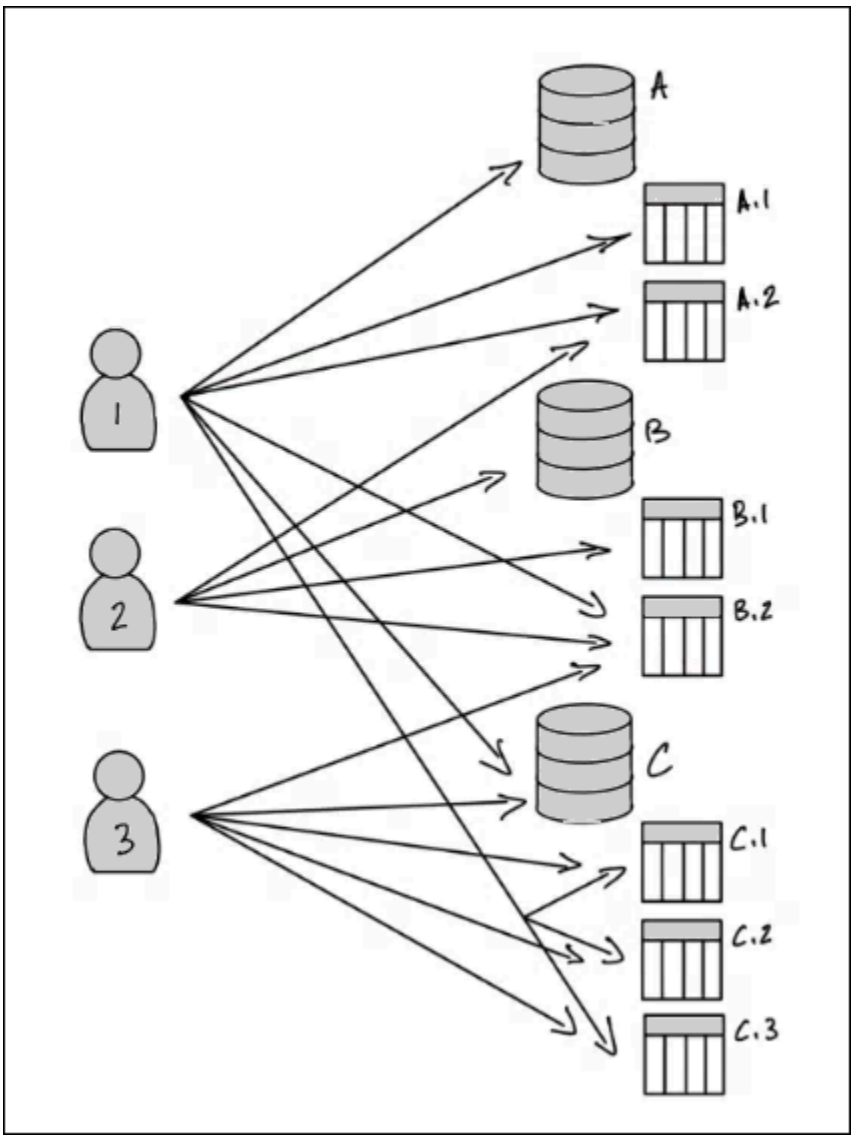
Each LF-Tag is a key-value pair, such as `department=sales` or `classification=restricted`. A key can have multiple defined values, such as `department=sales,marketing,engineering,finance`.

To use the LF-TBAC method, data lake administrators and data engineers perform the following tasks.

Task	Task details
1. Define the properties and relationships of LF-Tags.	-
2. Create the LF-Tag creators in Lake Formation.	Adding LF-Tag creators
3. Create the LF-Tag in Lake Formation.	Creating LF-Tags
4. Assign LF-Tags to Data Catalog resources.	Assigning LF-Tags to Data Catalog resources

Task	Task details
5. Grant permissions to other principals to assign LF-Tags to resources, optionally with the grant option.	Managing LF-Tag value permissions
6. Grant LF-Tag expressions to principals, optionally with the grant option.	Granting data lake permissions using the LF-TBAC method
7. (Recommended) After verifying that principals have access to the correct resources through the LF-TBAC method, revoke permissions that were granted by using the named resource method.	-

Consider the case where you must grant permissions to three principals on three databases and seven tables.



To achieve the permissions indicated in the preceding diagram by using the named resource method, you would have to make 17 grants, as follows (in pseudo-code).

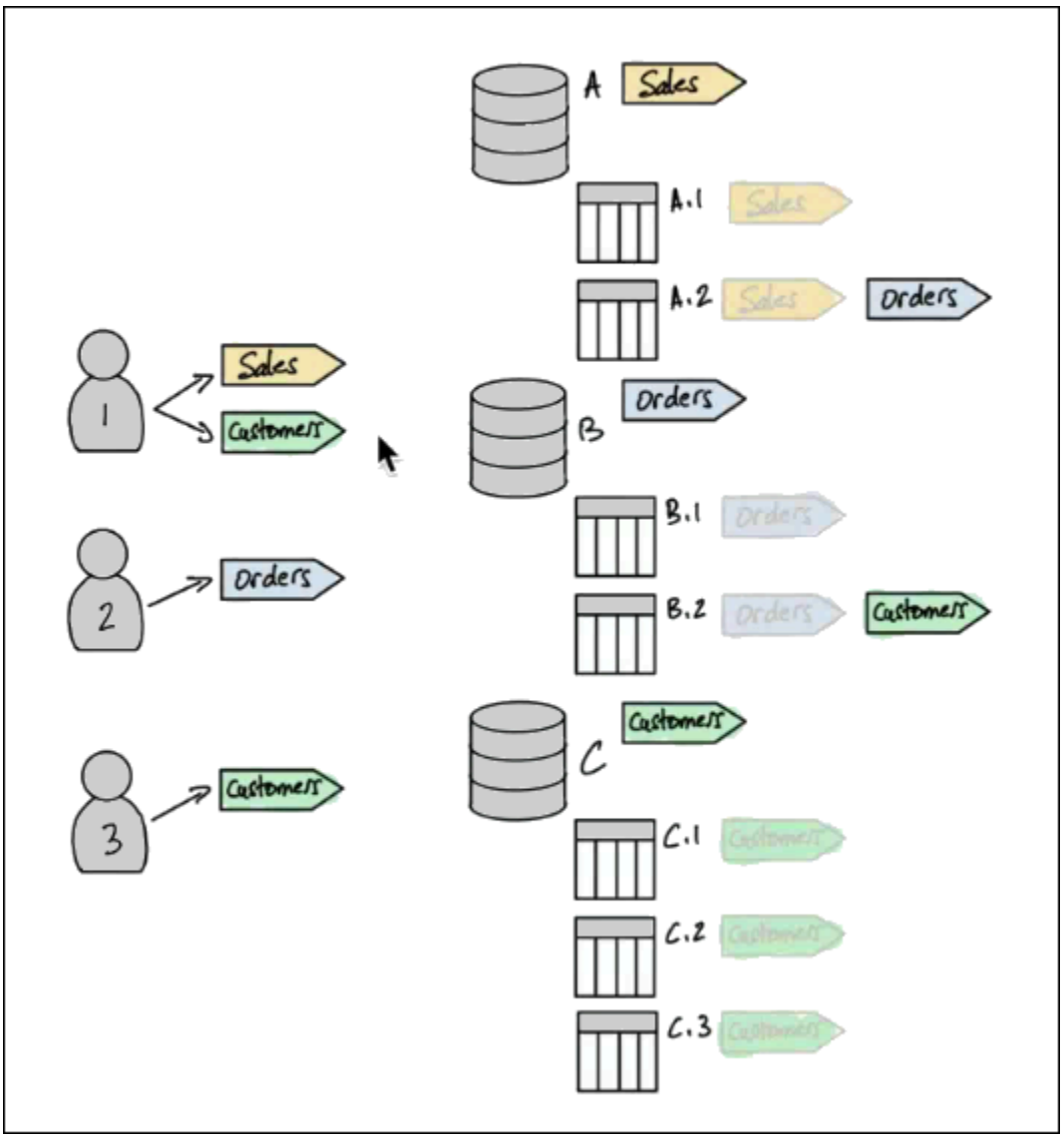
```
GRANT CREATE_TABLE ON Database A TO PRINCIPAL 1
GRANT SELECT, INSERT ON Table A.1 TO PRINCIPAL 1
GRANT SELECT, INSERT ON Table A.2 TO PRINCIPAL 1
GRANT SELECT, INSERT ON Table B.2 TO PRINCIPAL 1
...
GRANT SELECT, INSERT ON Table A.2 TO PRINCIPAL 2
GRANT CREATE_TABLE ON Database B TO PRINCIPAL 2
...
GRANT SELECT, INSERT ON Table C.3 TO PRINCIPAL 3
```

Now consider how you would grant permissions by using LF-TBAC. The following diagram indicates that you've assigned LF-Tags to databases and tables, and has granted permissions on LF-Tags to principals.

In this example, the LF-Tags represent areas of the data lake that contain analytics for different modules of an enterprise resource planning (ERP) application suite. You can use them to control access to the analytics data for the various modules. All LF-Tags have the key `module` and possible values `Sales`, `Orders`, and `Customers`. An example LF-Tag looks like this:

```
module=Sales
```

The diagram shows only the LF-Tag values.



Tag assignments to Data Catalog resources and inheritance

Tables inherit LF-Tags from databases and columns inherit LF-Tags from tables. Inherited values can be overridden. In the preceding diagram, dimmed LF-Tags are inherited.

Because of inheritance, the data lake administrator needs to make only the five following LF-Tag assignments to resources (in pseudo-code).

```
ASSIGN TAGS module=Sales TO database A
ASSIGN TAGS module=Orders TO table A.2
ASSIGN TAGS module=Orders TO database B
ASSIGN TAGS module=Customers TO table B.2
ASSIGN TAGS module=Customers TO database C
```

Tag grants to principals

After assigning LF-Tags to the databases and tables, the data lake administrator must make only four grants of LF-Tags to principals, as follows (in pseudo-code).

```
GRANT TAGS module=Sales TO Principal 1
GRANT TAGS module=Customers TO Principal 1
GRANT TAGS module=Orders TO Principal 2
GRANT TAGS module=Customers TO Principal 3
```

Now, a principal with the `module=Sales`LF-Tag can access Data Catalog resources with the `module=Sales` LF-Tag (for example, database A), a principal with the `module=Customers` LF-Tag can access resources with the `module=Customers` LF-Tag, and so on.

The preceding grant commands are incomplete. This is because although they indicate through LF-Tags the Data Catalog resources that the principals have permissions on, they don't indicate exactly which Lake Formation permissions (such as `SELECT`, `ALTER`) the principals have on those resources. Therefore, the following pseudo-code commands are a more accurate representation of how Lake Formation permissions are granted on Data Catalog resources through LF-Tags.

```
GRANT (CREATE_TABLE ON DATABASES) ON TAGS module=Sales TO Principal 1
GRANT (SELECT, INSERT ON TABLES)  ON TAGS module=Sales TO Principal 1
GRANT (CREATE_TABLE ON DATABASES) ON TAGS module=Customers TO Principal 1
GRANT (SELECT, INSERT ON TABLES)  ON TAGS module=Customers TO Principal 1
GRANT (CREATE_TABLE ON DATABASES) ON TAGS module=Orders TO Principal 2
GRANT (SELECT, INSERT ON TABLES)  ON TAGS module=Orders TO Principal 2
GRANT (CREATE_TABLE ON DATABASES) ON TAGS module=Customers TO Principal 3
GRANT (SELECT, INSERT ON TABLES)  ON TAGS module=Customers TO Principal 3
```

Putting it together - Resulting permissions on resources

Given the LF-Tags assigned to the databases and tables in the preceding diagram, and the LF-Tags granted to the principals in the diagram, the following table lists the Lake Formation permissions that the principals have on the databases and tables.

Principal	Permissions Granted Through LF-Tags
Principal 1	• `CREATE_TABLE` on database A • `SELECT`, `INSERT` on table A.1 • `SELECT`, `INSERT` on table B.2 • `CREATE_TABLE` on database C • `SELECT`, `INSERT` on table C.1 • `SELECT`, `INSERT` on table C.2 • `SELECT`, `INSERT` on table C.3
Principal 2	• `SELECT`, `INSERT` on table A.2 • `CREATE_TABLE` on database B • `SELECT`, `INSERT` on table B.1
Principal 3	• `SELECT`, `INSERT` on table B.2 • `CREATE_TABLE` on database C • `SELECT`, `INSERT` on table C.1 • `SELECT`, `INSERT` on table C.2 • `SELECT`, `INSERT` on table C.3

Bottom line

In this simple example, using five assignment operations and eight grant operations, the data lake administrator was able to specify 17 permissions. When there are tens of databases and hundreds of tables, the advantage of the LF-TBAC method over the named resource method becomes clear. In the hypothetical case of the need to grant every principal access to every resource, and where `n(P)` is the number of principals and `n(R)` is the number of resources:

- With the named resource method, the number of grants required is $n(P) \times n(R)$.

- With the LF-TBAC method, using a single LF-Tag, the total of the number of grants to principals and assignments to resources is `n(P) + n(R)`.

See also

- Managing LF-Tags for metadata access control
- Granting data lake permissions using the LF-TBAC method

Topics

- Managing LF-Tags for metadata access control
- Managing LF-Tag expressions for metadata access control
- Managing LF-Tag value permissions

Managing LF-Tags for metadata access control

To use the Lake Formation tag-based access control (LF-TBAC) method to secure Data Catalog objects such as catalogs, databases, tables, views, and columns, you create LF-Tags, assign them to resources, and grant LF-Tag permissions to principals.

Before you can assign LF-Tags to Data Catalog objects or grant permissions to principals, you need to define LF-Tags. Only a data lake administrator or a principal with LF-Tag creator permissions can create LF-Tags.

LF-Tag creators

LF-Tag creator is a non-admin principal who has permissions to create and manage LF-Tags. Data lake administrators can add LF-Tag creators using the Lake Formation console or CLI. LF-Tag creators have implicit Lake Formation permissions to update, and delete LF-Tags, to assign LF-Tags to resources, and to grant LF-Tag permissions and LF-Tag value permissions to other principals.

With LF-Tag creator roles, data lake administrators can delegate tag management tasks such as creating and updating tag keys and values to non-admin principals. Data lake administrators can also grant LF-Tag creators grantable `Create LF-Tag` permissions. Then, the LF-Tag creator can grant the permission to create LF-Tags to other principals.

You can grant two types of permissions on LF-Tags:

- LF-Tag permissions - `Create LF-Tag`, `Alter`, and `Drop`. These permissions are required to create, update, and delete LF-Tags.

  Data lake administrators and LF-Tag creators implicitly have these permissions on the LF-Tags they create and can grant these permissions explicitly to principals to manage tags in the data lake.
- LF-Tag key-value pair permissions - `Assign`, `Describe`, and `Grant with LF-Tag expressions`. These permissions are required to assign LF-Tags to Data Catalog objects, and to grant permissions on the resources to principals using Lake Formation tag-based access control. LF-Tag creators implicitly receive these permissions when creating LF-Tags.

After receiving the `Create LF-Tag` permission and successfully creating LF-Tags, the LF-Tag creator can assign LF-Tags to resources and grant LF-Tag permissions (`Create LF-Tag`, `Alter`, `Drop`, and) to other non-administrative principals to manage tags in the data lake. You can manage LF-Tags by using the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

> **Note**
>
> Data lake administrators have implicit Lake Formation permissions to create, update, and delete LF-Tags, to assign LF-Tags to resources, and to grant LF-Tag permissions to principals.

For best practices and considerations, see Lake Formation tag-based access control best practices and considerations

Topics

- Adding LF-Tag creators
- Creating LF-Tags
- Updating LF-Tags
- Deleting LF-Tags
- Listing LF-Tags
- Assigning LF-Tags to Data Catalog resources
- Viewing LF-Tags assigned to a resource
- Viewing the resources that a LF-Tag is assigned to

- Life cycle of a LF-Tag
- Comparison of Lake Formation tag-based access control to IAM attribute-based access control

See also

- Managing LF-Tag value permissions
- Granting data lake permissions using the LF-TBAC method
- Lake Formation tag-based access control

Adding LF-Tag creators

By default, data lake administrators can create, update, and delete LF-Tags, assign tags to Data Catalog objects, and grant tag permissions to principals. If you wish to delegate the tag creation and management operations to non-admin principals, the data lake administrator can create LF-Tag creator roles and grant Lake Formation `Create LF-Tag` permission to the roles. With grantable `Create LF-Tag` permission, LF-Tag creators can delegate tag creation and maintenance tasks to other non-administrative principals.

For data lake administrators to assign LF-Tags to Data Catalog resources, they are required to grant themselves associate permissions on LF-Tags that were not created by them.

Note

Cross-account permission grants can include only `Describe` and `Associate` permissions. You can't grant `Create LF-Tag`, `Drop`, `Alter`, and `Grant with LFTag expressions` permissions to principals in a different account.

Topics

- IAM permissions required to create LF-Tags
- Add LF-Tag creators

See also

- Managing LF-Tag value permissions

- Granting data lake permissions using the LF-TBAC method
- Lake Formation tag-based access control

IAM permissions required to create LF-Tags

You must configure permissions to allow a Lake Formation principal to create LF-Tags. Add the following statement to the permissions policy for the principal that needs to be a LF-Tag creator.

> **Note**
>
> Although data lake administrators have implicit Lake Formation permissions to create, update, and delete LF-Tags, to assign LF-Tags to resources, and to grant LF-Tags to principals, data lake administrators also need the following IAM permissions.

For more information, see Lake Formation personas and IAM permissions reference.

```
{
"Sid": "Transformational",
"Effect": "Allow",
    "Action": [
        "lakeformation:AddLFTagsToResource",
        "lakeformation:RemoveLFTagsFromResource",
        "lakeformation:GetResourceLFTags",
        "lakeformation:ListLFTags",
        "lakeformation:CreateLFTag",
        "lakeformation:GetLFTag",
        "lakeformation:UpdateLFTag",
        "lakeformation:DeleteLFTag",
        "lakeformation:SearchTablesByLFTags",
        "lakeformation:SearchDatabasesByLFTags"
     ]
 }
```

Principals who assign LF-Tags to resources and grant LF-Tags to principals must have the same permissions, except for the `CreateLFTag`, `UpdateLFTag`, and `DeleteLFTag` permissions.

Add LF-Tag creators

A LF-Tag creator can create a LF-Tag, update tag key and values, delete tags, associate tags to Data Catalog resources, and grant permissions on Data Catalog resources to principals using LF-TBAC method. The LF-Tag creator can also grant these permissions to principals.

You can create LF-Tag creator roles by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

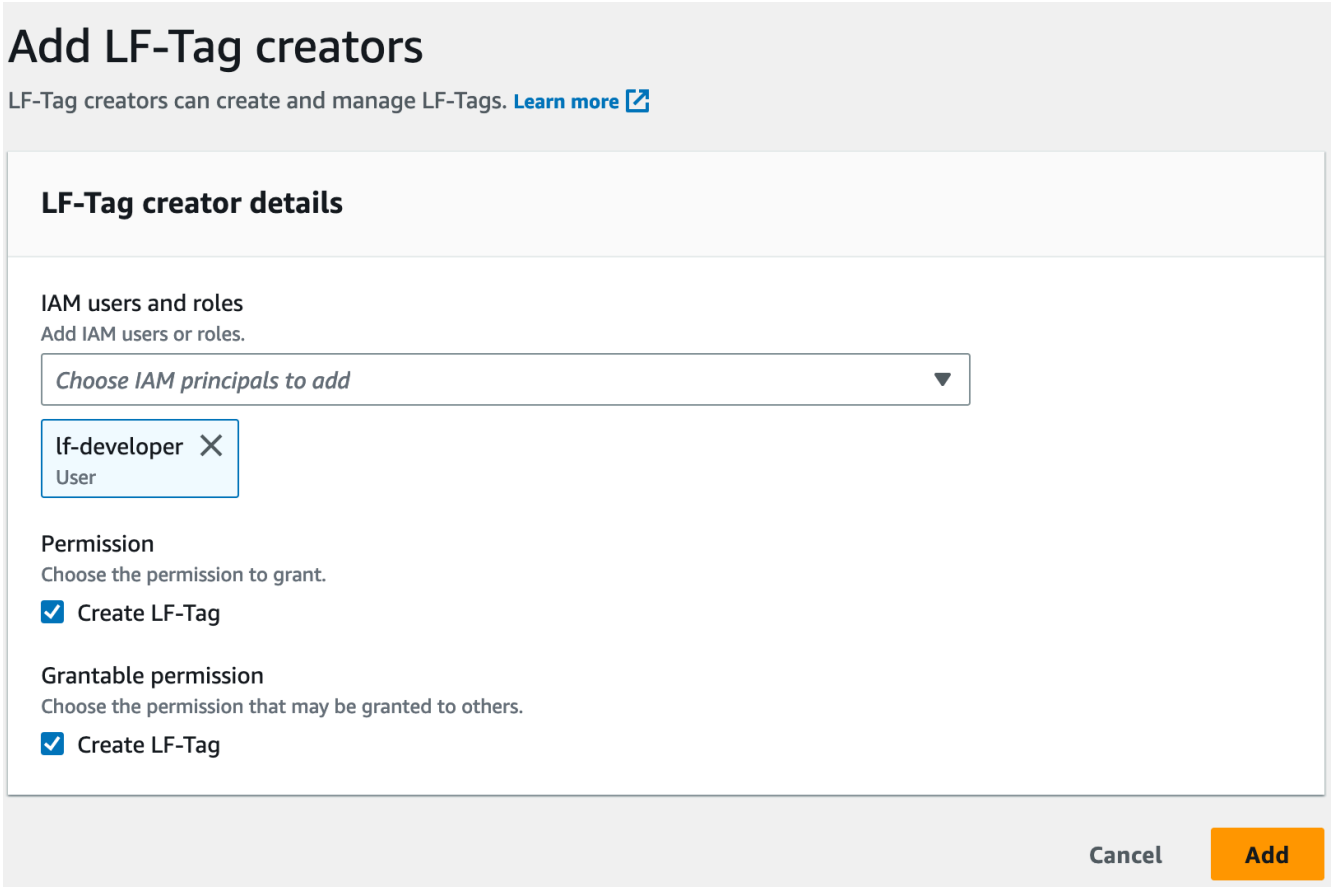
console

To add a LF-Tag creator

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a datalake administrator.
2. In the navigation pane, under **Permissions**, choose **LF-Tags and permissions**.

   On the **LF-Tags and permissions** page, choose **LF-Tag creators** section and choose **Add LF-Tag creators**.



3. On the **Add LF-Tag creators** page, choose an IAM role or user who has the required permissions to create LF-Tags.
4. Enable `Create LF-Tag` permission check box.
5. (Optional) To enable the selected principals to grant `Create LF-Tag` permission to principals, choose Grantable `Create LF-Tag` permission.
6. Choose **Add**.

AWS CLI

```
aws lakeformation grant-permissions --cli-input-json file://grantCreate
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::123456789012:user/tag-manager"
    },
    "Resource": {
        "Catalog": {}
    },
    "Permissions": [
        "CreateLFTag"
    ],
    "PermissionsWithGrantOption": [
        "CreateLFTag"
    ]
}
```

The following are the permissions available for a LF-Tag creator role:

Permission	Description
`Drop`	A principal with this permission on a LF-Tag can delete a LF-Tag from the data lake. The principal gets implicit `Describe` permission on all tag values of a LF-Tag resource.
`Alter`	A principal with this permission on a LF-Tag can add or remove tag value from a LF-Tag. The principal gets implicit `Alter` permission on all tag values of a LF-Tag.

Permission	Description
`Describe`	A principal with this permission on a LF-Tag can view the LF-Tag and its values when they assign LF-Tags to resources or grant permissions on LF-Tags. You can grant `Describe` on all key values or on specific values.
`Associate`	A principal with this permission on a LF-Tag can assign the LF-Tag to a Data Catalog resource. Granting `Associate` implicitly grants `Describe`.
`Grant with LF-Tag expression`	A principal with this permission on a LF-Tag can grant permissions on a Data Catalog resources using the LF-Tag key and values. Granting `Grant with LF-Tag expression` implicitly grants `Describe`.

These permissions are grantable. A principal who has been granted these permissions with the grant option can grant them to other principals.

Creating LF-Tags

All LF-Tags must be defined in Lake Formation before they can be used. A LF-Tag consists of a key and one or more possible values for the key.

After the data lake administrator has setup the required IAM permissions and Lake Formation permissions for the LF-Tag creator role, the principal can create a LF-Tag. The LF-Tag creator gets implicit permission to update or remove any tag value from the LF-Tag and delete the LF-Tag.

You can create LF-Tags by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

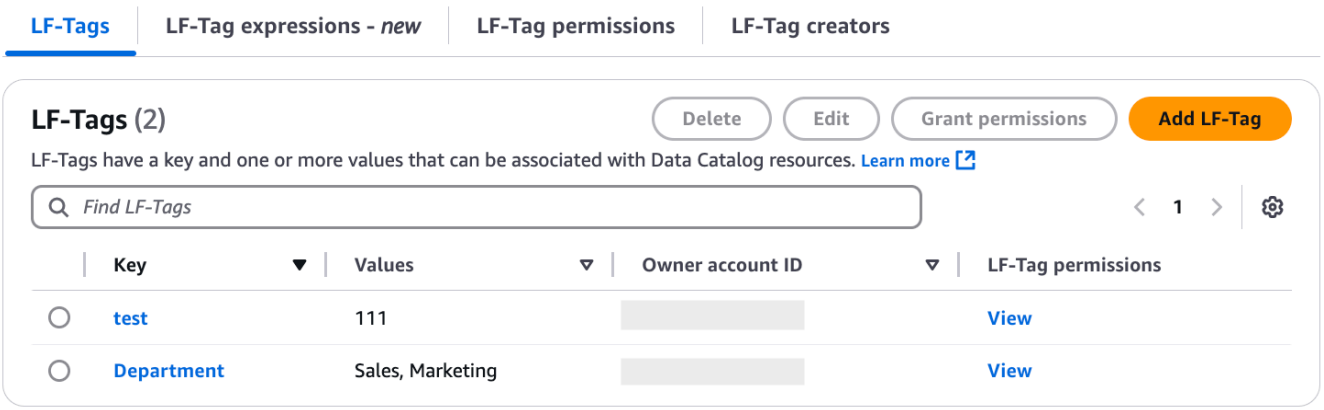
Console

To create a LF-Tag

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a principal with LF-Tag creator permissions or as data lake administrator.
2. In the navigation pane, under **Permissions**, **LF-Tags and permissions**, choose **LF-Tags**.

   The **LF-Tags** page appears.



3. Choose **Add LF-Tag**.
4. In the **Add LF-Tag** dialog box, enter a key and one or more values.

   Each key must have at least one value. To enter multiple values, either enter a comma-delimited list and then press **Enter**, or enter one value at a time and choose **Add** after each one. The maximum number of values permitted is 1000.
5. Choose **Add tag**.

AWS CLI

To create a LF-Tag

- Enter a `create-lf-tag` command.

  The following example creates a LF-Tag with key `module` and values `Customers` and `Orders`.

```
aws lakeformation create-lf-tag --tag-key module --tag-values Customers Orders
```

As tag creator , the principal gets `Alter` permission on this LF-Tag and can update or remove any tag value from this LF-Tag. The LF-Tag creator principal can also grant `Alter` permission to another principal to update and remove tag values on this LF-Tag.

Updating LF-Tags

You update a LF-Tag that you have the `Alter` permission on by adding or deleting permitted key values. You can't change the LF-Tag key. To change the key, delete the LF-Tag and add one with the

required key. In addition to `Alter` permission, you also need the `lakeformation:UpdateLFTag` IAM permission to update values.

When you delete a LF-Tag value, no check is performed for the presence of that LF-Tag value on any Data Catalog resource. If the deleted LF-Tag value is associated with a resource, it is no longer visible for the resource, and any principals that were granted permissions on that key-value pair no longer have the permissions.

Before deleting a LF-Tag value, you can optionally use the `remove-lf-tags-from-resource` command command to remove the LF-Tag from Data Catalog resources that have the value that you want to delete, and then retag the resource with the values that you want to keep.

Only data lake administrators, the LF-Tag creator, and principals that have `Alter` permissions on the LF-Tag can update a LF-Tag.

You can update a LF-Tag by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Console

To update a LF-Tag (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator, LF-Tag creator or a principal with `Alter` permission on the LF-Tag.
2. In the navigation pane, under **Permissions**, **LF-Tags and permissions**, choose **LF-Tags**.
3. On the **LF-Tags** page, select a LF-Tag, and then choose **Edit**.
4. In the **Edit LF-Tag** dialog box, add or remove LF-Tag values.

   To add multiple values, in the **Values** field, either enter a comma-delimited list and press **Enter**, or enter one value at a time or choose **Add** after each one.
5. Choose **Save**.

AWS CLI

To update a LF-Tag (AWS CLI)

- Enter an `update-lf-tag` command. Provide one or both of the following arguments:

- `--tag-values-to-add`
- `--tag-values-to-delete`

Example

The following example replaces the value `vp` with the value `vice-president` for the LF-Tag key `level`.

```
aws lakeformation update-lf-tag --tag-key level --tag-values-to-add vice-president
--tag-values-to-delete vp
```

Deleting LF-Tags

You can delete LF-Tags that are no longer in use. No check is performed for the presence of the LF-Tag on a Data Catalog resource. If the deleted LF-Tag is associated with a resource, it is no longer visible for the resource, and any principals that were granted permissions on that LF-Tag no longer have the permissions.

Before deleting a LF-Tag, you can optionally use the `remove-lf-tags-from-resource` command to remove the LF-Tag from all resources.

Only data lake administrators, the LF-Tag creator, or a principal that has `Drop` permission on the LF-Tag can delete a LF-Tag. In addition to the `Drop` permission, the principal also need `lakeformation:DeleteLFTag` IAM permission to delete a LF-Tag.

You can delete a LF-Tag by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Console

To delete a LF-Tag (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator.
2. In the navigation pane, under **Permissions**, **LF-Tags and permissions**, choose **LF-Tags**.
3. On the **LF-Tags** page, select a LF-Tag, and then choose **Delete**.

4. In the **Delete tag environment?** dialog box, to confirm the deletion, enter the LF-Tag key value in the designated field and then choose **Delete**.

AWS CLI

To delete a LF-Tag (AWS CLI)

- Enter a `delete-lf-tag` command. Provide the key of the LF-Tag to delete.

  Example

  The following example deletes the LF-Tag with the key `region`.

  ```
  aws lakeformation delete-lf-tag --tag-key region
  ```

Listing LF-Tags

You can list the LF-Tags that you have the `Describe` or `Associate` permissions on. The values listed with each LF-Tag key are the values that you have permissions on.

LF-Tag creator has implicit permissions to see the LF-Tags they have created.

Data lake administrators can see all LF-Tags that are defined in the local AWS account and all LF-Tags for which the `Describe` and `Associate` permissions have been granted to the local account from external accounts. The data lake administrator can see all values for all LF-Tags.

You can list LF-Tags by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Console

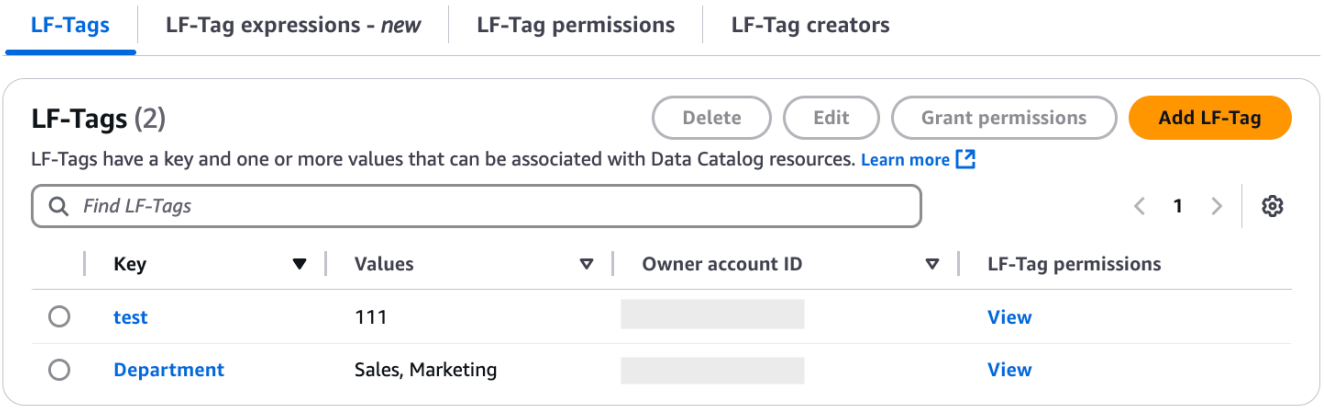
To list LF-Tags (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as the LF-Tag creator, as a data lake administrator, or as a principal that has been granted permissions on LF-Tags and that has the `lakeformation:ListLFTags` IAM permission.

2. In the navigation pane, under **Permissions**, **LF-Tags and permissions**, choose **LF-Tags**.

The **LF-Tags** page appears.



Check the **Owner account ID** column to determine the LF-Tags that were shared with your account from an external account.

AWS CLI

To list LF-Tags (AWS CLI)

- Run the following command as a data lake administrator or as a principal that has been granted permissions on LF-Tags and that has the `lakeformation:ListLFTags` IAM permission.

```
aws lakeformation list-lf-tags
```

The output is similar to the following.

```
{
    "LFTags": [
        {
            "CatalogId": "111122223333",
            "TagKey": "level",
            "TagValues": [
                "director",
                "vp",
                "c-level"
            ]
        },
        {
```

```
            "CatalogId": "111122223333",
            "TagKey": "module",
            "TagValues": [
                "Orders",
                "Sales",
                "Customers"
            ]
        }
    ]
}
```

To also see LF-Tags that were granted from external accounts, include the command option `--resource-share-type ALL`.

```
aws lakeformation list-lf-tags --resource-share-type ALL
```

The output is similar to the following. Note the `NextToken` key, which indicates that there is more to list.

```
{
    "LFTags": [
        {
            "CatalogId": "111122223333",
            "TagKey": "level",
            "TagValues": [
                "director",
                "vp",
                "c-level"
            ]
        },
        {
            "CatalogId": "111122223333",
            "TagKey": "module",
            "TagValues": [
                "Orders",
                "Sales",
                "Customers"
            ]
        }
    ],
    "NextToken": "eyJleHBpcmF0aW...ZXh0Ijp0cnVlfQ=="
```

```
}
```

Repeat the command, and add the `--next-token` argument to view any remaining local LF-Tags and LF-Tags that were granted from external accounts. LF-Tags from external accounts are always on a separate page.

```
aws lakeformation list-lf-tags --resource-share-type ALL
--next-token eyJleHBpcmF0aW...ZXh0Ijp0cnVlfQ==
```

```
{
    "LFTags": [
        {
            "CatalogId": "123456789012",
            "TagKey": "region",
            "TagValues": [
                "central",
                "south"
            ]
        }
    ]
}
```

API

You can use the SDKs available for Lake Formation to lists the tags that the requester has permission to view.

```
import boto3

client = boto3.client('lakeformation')
...

response = client.list_lf_tags(
    CatalogId='string',
    ResourceShareType='ALL',
    MaxResults=50'
)
```

This command returns a `dict` object with the following structure:

```
{
    'LFTags': [
        {
            'CatalogId': 'string',
            'TagKey': 'string',
            'TagValues': [
                'string',
            ]
        },
    ],
    'NextToken': 'string'
}
```

For more information about the required permissions, see Lake Formation personas and IAM permissions reference.

Assigning LF-Tags to Data Catalog resources

You can assign LF-Tags to Data Catalog resources (databases, tables, and columns) to control access to those resources. Only principals that are granted matching LF-Tags (and principals that are granted access with the named resource method) can access the resources.

If a table inherits a LF-Tag from a database or a column inherits a LF-Tag from a table, you can override the inherited value by assigning a new value to the LF-Tag key.

The maximum number of LF-Tags that you can assign to a resource is 50.

Topics

- Requirements for managing tags assigned to resources
- Assign LF-Tags to a table column
- Assign LF-Tags to a Data Catalog resource
- Updating LF-Tags for a resource
- Removing LF-Tag from a resource

Requirements for managing tags assigned to resources

To assign a LF-Tag to a Data Catalog resource, you must:

- Have the Lake Formation `ASSOCIATE` permission on the LF-Tag.

- Have the IAM `lakeformation:AddLFTagsToResource` permission.
- Have glue:GetDatabase permission on a Glue database.
- Be the resource owner (creator), have the `Super` Lake Formation permission on the resource with the `GRANT` option, or have the following permissions with the `GRANT` option:
  - For databases in the same AWS account: `DESCRIBE`, `CREATE_TABLE`, `ALTER`, and `DROP`
  - For databases in an external account: `DESCRIBE`, `CREATE_TABLE` and `ALTER`
  - For tables (and columns): `DESCRIBE`, `ALTER`, `DROP`, `INSERT`, `SELECT`, and `DELETE`

In addition, the LF-Tag and the resource that it is being assigned to must be in the same AWS account.

To remove a LF-Tag from a Data Catalog resource, you must meet these requirements, and also have the `lakeformation:RemoveLFTagsFromResource` IAM permission.

Assign LF-Tags to a table column

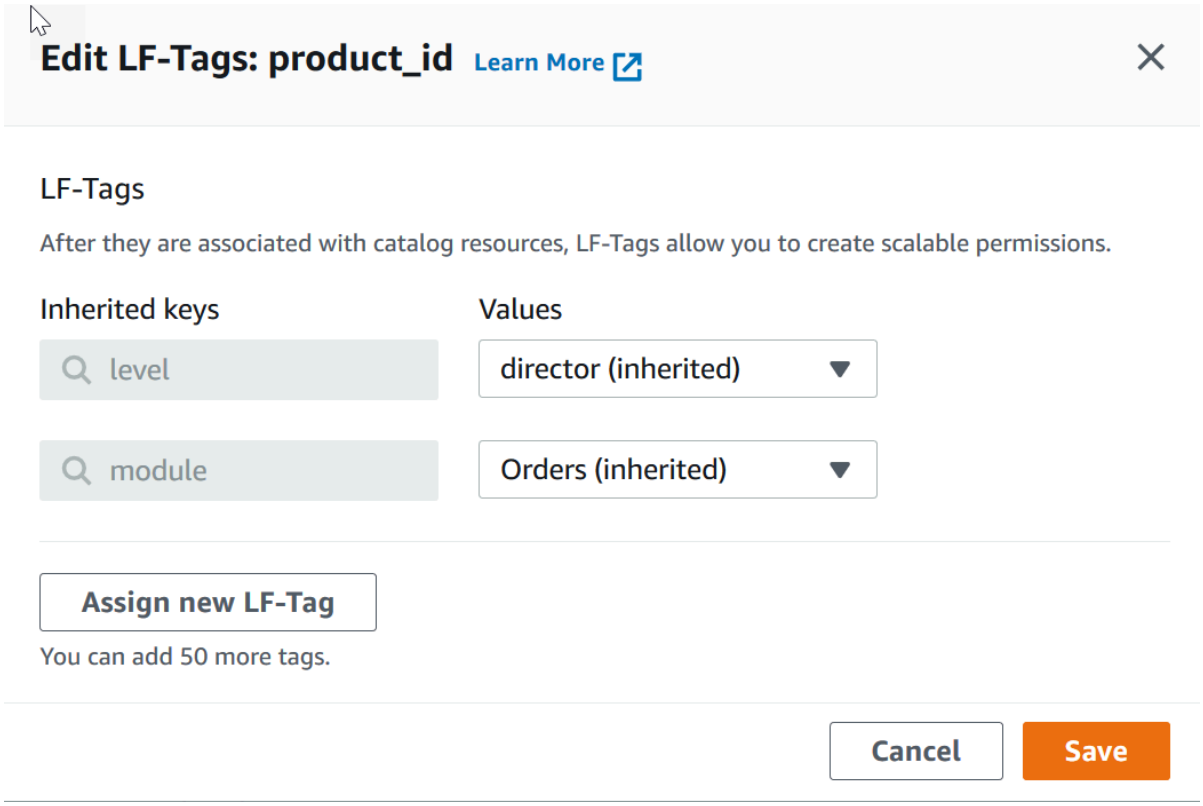
To assign LF-Tags to a table column (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

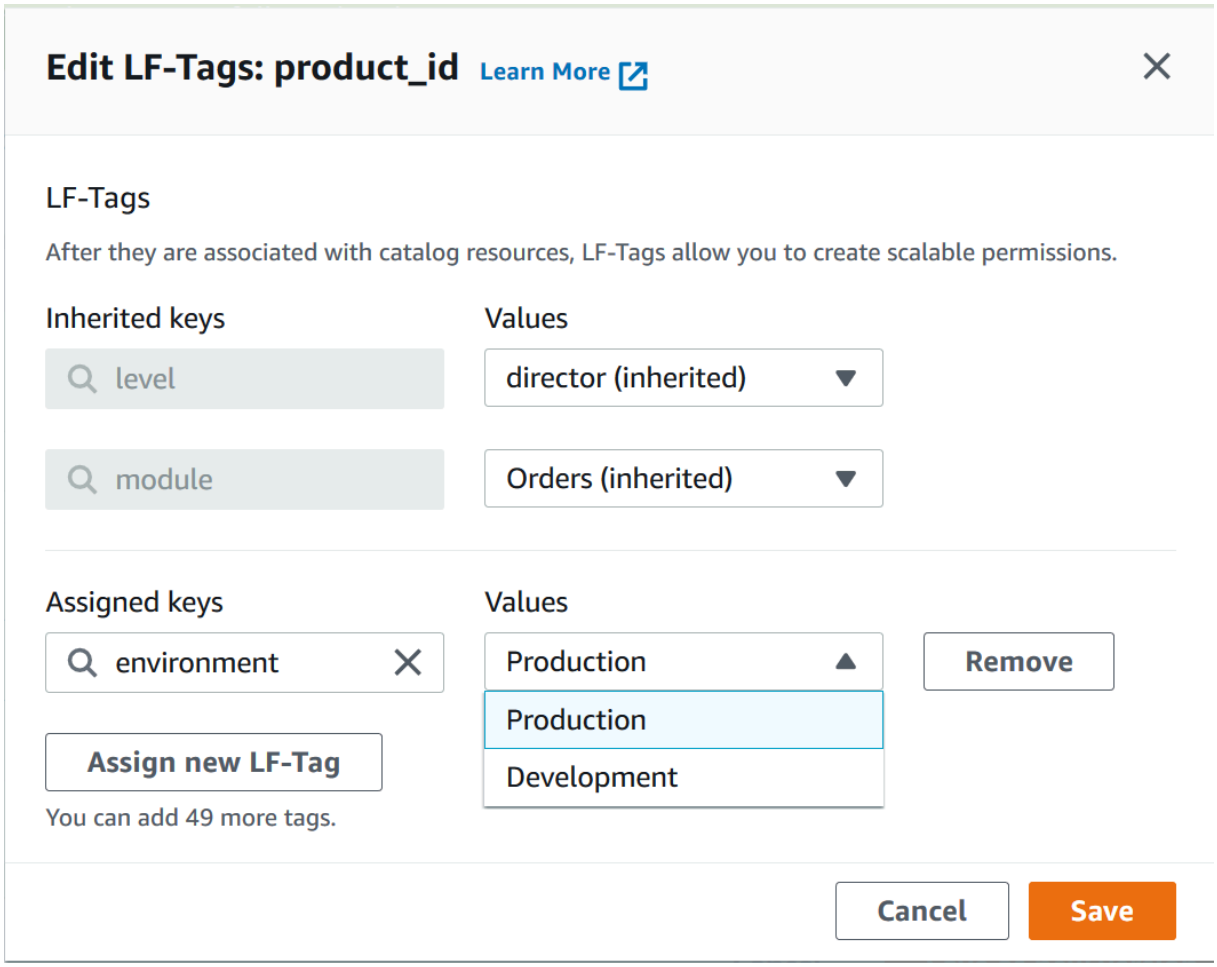
   Sign in as a user who meets the requirements listed above.
2. In the navigation pane, choose **Tables**.
3. Choose a table name (not the option button next to the table name).
4. On the table details page, in the **Schema** section, choose **Edit schema**.
5. On the **Edit schema** page, select one or more columns, and then choose **Edit LF-Tags**.

   > **Note**
   >
   > If you intend to add or delete columns and save a new version, do that first. Then edit the LF-Tags.

   The **Edit LF-Tags** dialog box appears, and displays any LF-Tags that are inherited from the table.



6. (Optional) For the **Values** list next to an **Inherited keys** field, choose a value to override the inherited value.
7. (Optional) Choose **Assign new LF-Tag**. Then for **Assigned keys**, choose a key, and for **Values**, choose a value for the key.



8. (Optional) Choose **Assign new LF-Tag** again to add another LF-Tag.
9. Choose **Save**.

Assign LF-Tags to a Data Catalog resource

Console

To assign LF-Tags to a Data Catalog database or table

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

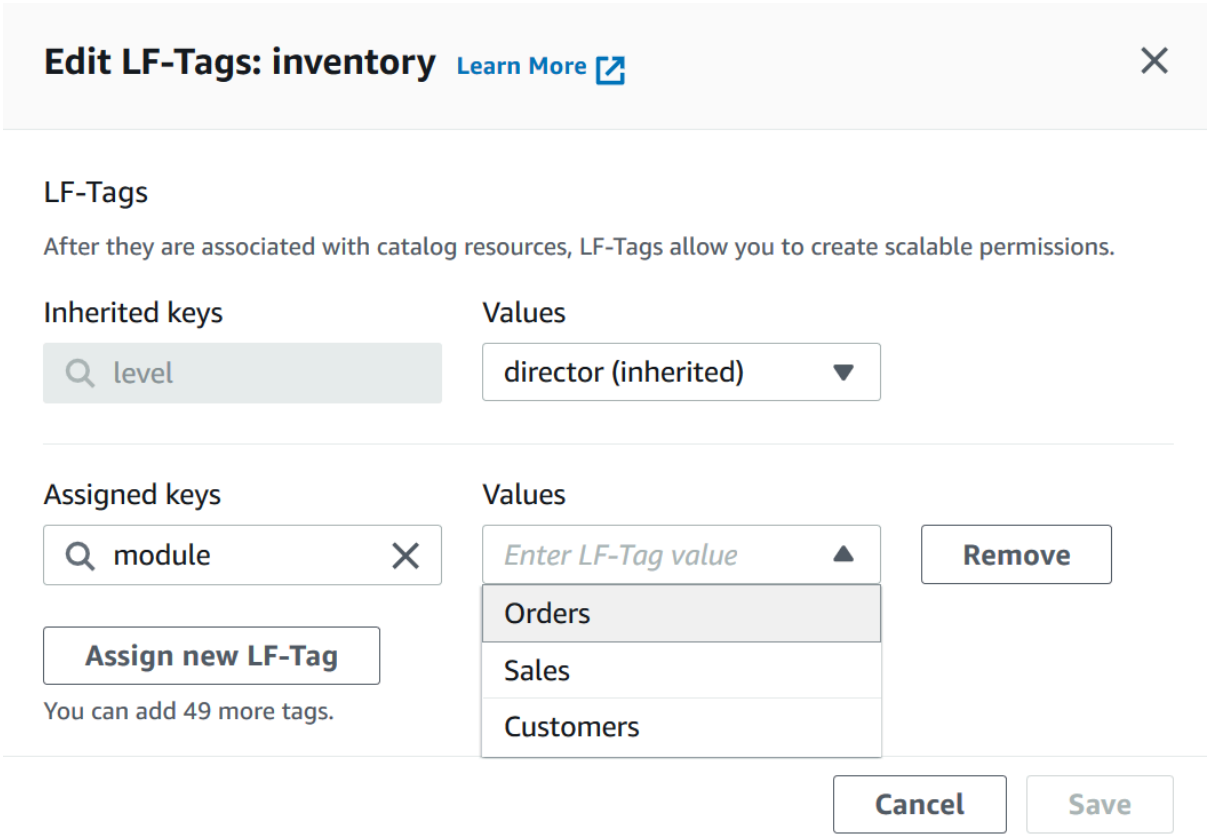
   Sign in as a user who meets the requirements listed earlier.
2. In the navigation pane, under **Data catalog**, do one of the following:

   - To assign LF-Tags to databases, choose **Databases**.
   - To assign LF-Tags to tables, choose **Tables**.

3. Choose a database or table, and on the **Actions** menu, choose **Edit LF-Tags**.

   The **Edit LF-Tags: *resource-name*** dialog box appears.

   If a table inherits LF-Tags from its containing database, the window displays the inherited LF-Tags. Otherwise, it displays the text "There are no inherited LF-Tags associated with the resource."



4. (Optional) If a table has inherited LF-Tags, for the **Values** list next to an **Inherited keys** field, you can choose a value to override the inherited value.
5. To assign new LF-Tags, perform these steps:

   a. Choose **Assign new LF-Tag**.

   b. In the **Assigned keys** field, choose a LF-Tag key, and in the **Values** field, choose a value.

   c. (Optional) Choose **Assign new LF-Tag** again to assign an additional LF-Tag.
6. Choose **Save**.

AWS CLI

To assign LF-Tags to a Data Catalog resource

- Run the `add-lf-tags-to-resource` command.

  The following example assigns the LF-Tag `module=orders` to the table `orders` in the database `erp`. It uses the shortcut syntax for the `--lf-tags` argument. The `CatalogID` property for `--lf-tags` is optional. If not provided, the catalog ID of the resource (in this case, the table) is assumed.

```
aws lakeformation add-lf-tags-to-resource --resource '{ "Table":
 {"DatabaseName":"erp", "Name":"orders"}}' --lf-tags
 CatalogId=111122223333,TagKey=module,TagValues=orders
```

  The following is the output if the command succeeds.

```
{
    "Failures": []
}
```

  This next example assigns two LF-Tags to the `sales` table, and uses the JSON syntax for the `--lf-tags` argument.

```
aws lakeformation add-lf-tags-to-resource --resource '{ "Table":
 {"DatabaseName":"erp", "Name":"sales"}}' --lf-tags '[{"TagKey":
 "module","TagValues": ["sales"]},{"TagKey": "environment","TagValues":
 ["development"]}]'
```

  This next example assigns the LF-Tag `level=director` to the `total` column of the table `sales`.

```
aws lakeformation add-lf-tags-to-resource --resource '{ "TableWithColumns":
 {"DatabaseName":"erp", "Name":"sales", "ColumnNames":["total"]}}' --lf-tags
 TagKey=level,TagValues=director
```

Updating LF-Tags for a resource

To update a LF-Tag for a Data Catalog resource (AWS CLI)

- Use the `add-lf-tags-to-resource` command, as described in the previous procedure.

  Adding a LF-Tag with the same key as an existing LF-Tag, but with a different value updates the existing value.

Removing LF-Tag from a resource

To remove a LF-Tag for a Data Catalog resource (AWS CLI)

- Run the `remove-lf-tags-from-resource` command.

  If a table has a LF-Tag value that overrides the value that is inherited from the parent database, removing that LF-Tag from the table restores the inherited value. This behavior also applies to a column that overrides key values inherited from the table.

  The following example removes the LF-Tag `level=director` from the `total` column of the `sales` table. The `CatalogID` property for `--lf-tags` is optional. If not provided, the catalog ID of the resource (in this case, the table) is assumed.

```
aws lakeformation remove-lf-tags-from-resource
--resource ' { "TableWithColumns":
{ "DatabaseName": "erp",  "Name": "sales",  "ColumnNames":[ "total"]}}'
--lf-tags  CatalogId=111122223333,TagKey=level,TagValues=director
```

Viewing LF-Tags assigned to a resource

You can view the LF-Tags that are assigned to a Data Catalog resource. You must have the `DESCRIBE` or `ASSOCIATE` permission on a LF-Tag to view it.

Console

To view the LF-Tags that are assigned to a resource (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

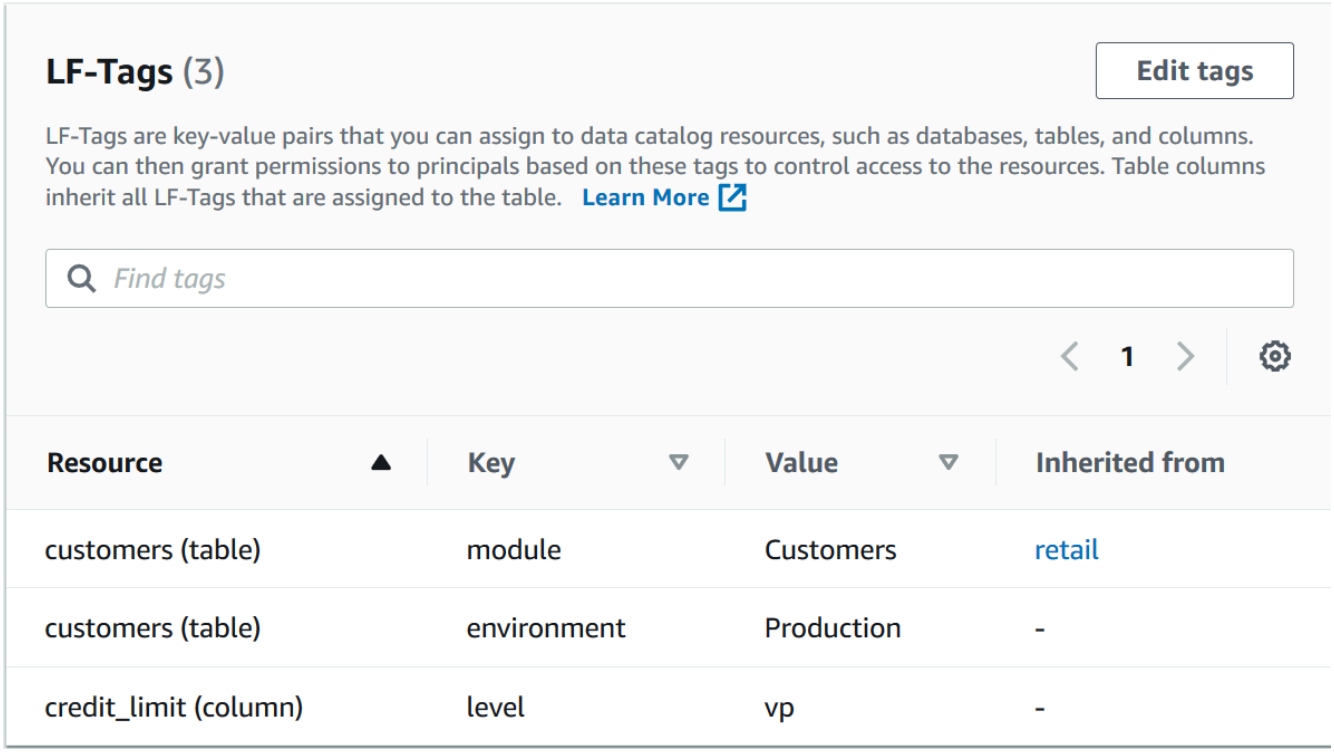
Sign in as the data lake administrator, the resource owner, or a user who has been granted Lake Formation permissions on the resource.

2. In the navigation pane, under the heading **Data catalog**, do one of the following:

   - To view LF-Tags assigned to a database, choose **Databases**.
   - To view LF-Tags assigned to a table, choose **Tables**.

3. On the **Tables** or **Databases** page, choose the name of the database or table. Then on the details page, scroll down to the **LF-Tags** section.

   The following screenshot shows the LF-Tags assigned to a `customers` table, which is contained in the `retail` database. The `module` LF-Tag is inherited from the database. The `credit_limit` column has the `level=vp` LF-Tag assigned.



AWS CLI

To view the LF-Tags that are assigned to a resource (AWS CLI)

- Enter a command similar to the following.

```
aws lakeformation get-resource-lf-tags --show-assigned-lf-tags --
resource '{ "Table": {"CatalogId":"111122223333", "DatabaseName":"erp",
 "Name":"sales"}}'
```

The command returns the following output.

```
{
    "TableTags": [
        {
            "CatalogId": "111122223333",
            "TagKey": "module",
            "TagValues": [
                "sales"
            ]
        },
        {
            "CatalogId": "111122223333",
            "TagKey": "environment",
            "TagValues": [
                "development"
            ]
        }
    ],
    "ColumnTags": [
        {
            "Name": "total",
            "Tags": [
                {
                    "CatalogId": "111122223333",
                    "TagKey": "level",
                    "TagValues": [
                        "director"
                    ]
                }
            ]
        }
    ]
}
```

This output shows only LF-Tags that are explicitly assigned, not inherited. If you want to see all LF-Tags on all columns, including inherited LF-Tags, omit the `--show-assigned-lf-tags` option.

Viewing the resources that a LF-Tag is assigned to

You can view all the Data Catalog resources that a particular LF-Tag key is assigned to. To do so, you need the following Lake Formation permissions:

- `Describe` or `Associate` on the LF-Tag.
- `Describe` or any other Lake Formation permission on the resource.

In addition, you need the following AWS Identity and Access Management (IAM) permissions:

- `lakeformation:SearchDatabasesByLFTags`
- `lakeformation:SearchTablesByLFTags`

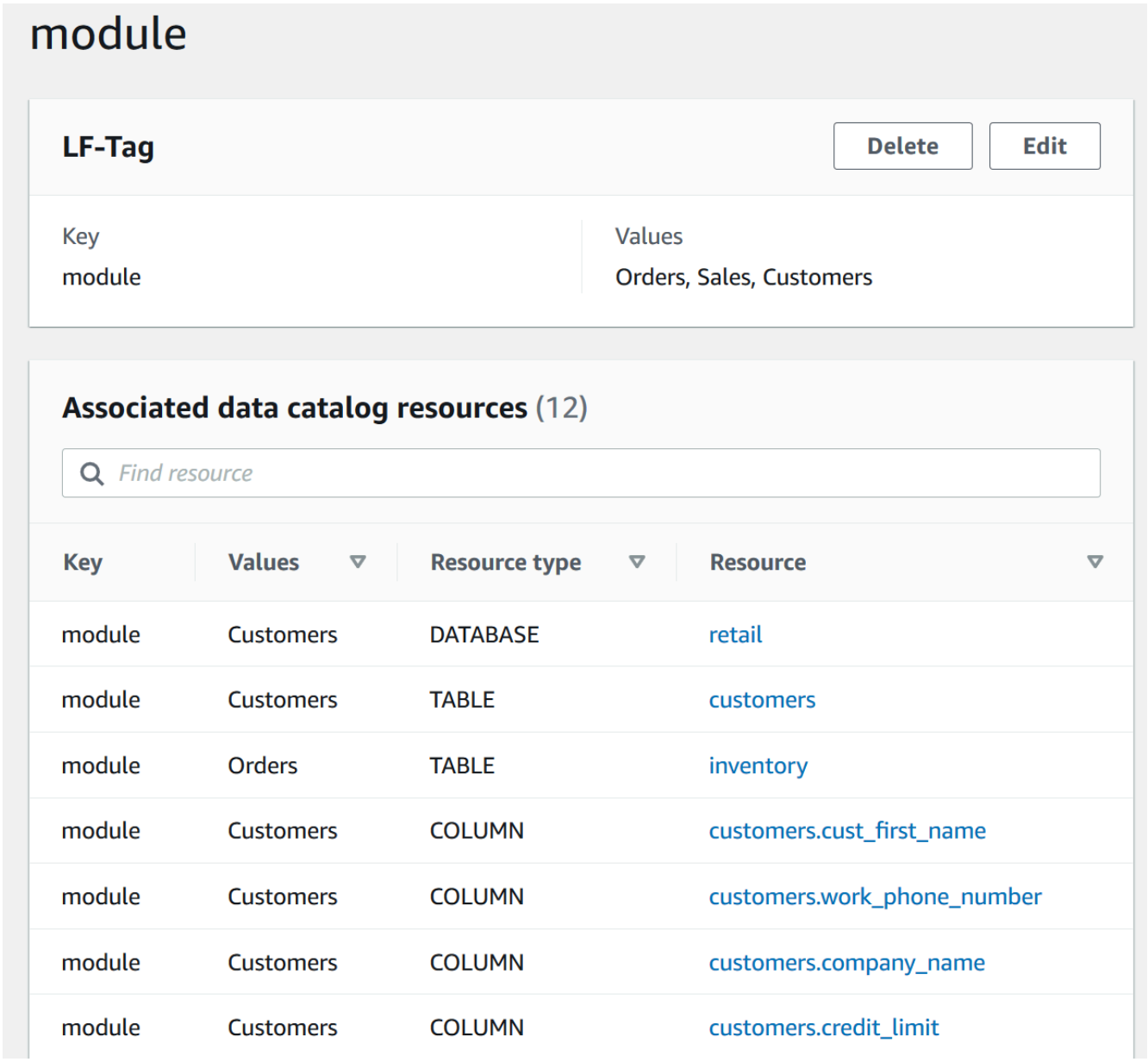
Console

To view the resources that a LF-Tag is assigned to (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator or as a user who meets the requirements listed earlier.
2. In the navigation pane, under **Permissions** and **LF-Tags and permissions**, choose **LF-Tags**.
3. Choose a LF-Tag key (not the option button next to the key name).

   The LF-Tag details page displays a list of resources that the LF-Tag has been assigned to.



AWS CLI

To view the resources that a LF-Tag is assigned to

- Run a `search-tables-by-lf-tags` or `search-databases-by-lf-tags` command.

Example

The following example lists tables and columns that have the `level=vp` LF-Tag assigned. For each table and column listed, all assigned LF-Tags for the table or column are output, not just the search expression.

```
aws lakeformation search-tables-by-lf-tags --expression
 TagKey=level,TagValues=vp
```

For more information about the required permissions, see Lake Formation personas and IAM permissions reference.

Life cycle of a LF-Tag

1. The LF-Tag creator Michael creates a LF-Tag `module=Customers`.
2. Michael grants `Associate` on the LF-Tag to the data engineer Eduardo. Granting `Associate` implicitly grants `Describe`.
3. Michael grants `Super` on the table `Custs` to Eduardo with the grant option, so that Eduardo can assign LF-Tags to the table. For more information, see Assigning LF-Tags to Data Catalog resources.
4. Eduardo assigns the LF-Tag `module=customers` to the table `Custs`.
5. Michael makes the following grant to data engineer Sandra (in pseudo-code).

```
GRANT (SELECT, INSERT ON TABLES) ON TAGS module=customers TO Sandra WITH GRANT OPTION
```

6. Sandra makes the following grant to data analyst Maria.

```
GRANT (SELECT ON TABLES) ON TAGS module=customers TO Maria
```

Maria can now run queries on the `Custs` table.

See also

- Metadata access control

Comparison of Lake Formation tag-based access control to IAM attribute-based access control

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM resources, including IAM entities (users or roles) and to AWS resources. You can create a single ABAC policy or small set of policies for your IAM principals. These ABAC policies can be designed to allow operations when the principal's tag matches the resource tag. ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

Cloud security and governance teams use IAM to define access policies and security permissions for all resources including Amazon S3 buckets, Amazon EC2 instances and any resources you can reference with an ARN. The IAM policies define broad (coarse-grained) permissions to your data lake resources, for example, to allow or deny access at Amazon S3 bucket or prefix level or database level. For more information about IAM ABAC, see What is ABAC for AWS? in the *IAM User Guide*.

For example, you can create three roles with the `project-access` tag key. Set the tag value of the first role to `Dev`, the second to `Marketing`, and the third to `Support`. Assign tags with the appropriate value to resources. You can then use a single policy that allows access when the role and the resource are tagged with the same value for `project-access`.

Data governance teams use Lake Formation to define fine-grained permissions to specific data lake resources. LF-Tags are assigned to Data Catalog resources (databases, tables, and columns) and are granted to principals. A principal with LF-Tags that match the LF-Tags of a resource can access that resource. Lake Formation permissions are secondary to IAM permissions. For example, if IAM permissions don't allow a user access to a data lake, Lake Formation doesn't grant access to any resource within that data lake to that user, even if the principal and resource have matching LF-Tags.

Lake Formation tag-based access control (LF-TBAC) works with IAM ABAC to provide additional levels of permissions for your Lake Formation data and resources.

- **Lake Formation TBAC permissions scale with innovation.** It's no longer necessary for an administrator to update existing policies to allow access to new resources. For example, assume that you use an IAM ABAC strategy with the `project-access` tag to provide access to specific databases within Lake Formation. Using LF-TBAC, the LF-Tag `Project=SuperApp` is assigned to specific tables or columns, and the same LF-Tag is granted to a developer for that project. Through IAM, the developer can access the database, and LF-TBAC permissions grant the developer further access to specific tables or columns within tables. If a new table is added to

the project, the Lake Formation administrator only needs to assign the tag to the new table for the developer to be given access to the table.

- **Lake Formation TBAC requires fewer IAM policies.** Because you use IAM policies to grant high level access to Lake Formation resources and Lake Formation TBAC for managing more precise data access, you create fewer IAM policies.
- **Using Lake Formation TBAC, teams can change and grow quickly.** This is because permissions for new resources are automatically granted based on attributes. For example, if a new developer joins the project, it's easy to grant this developer access by associating the IAM role to the user and then assigning the required LF-Tags to the user. You don't have to change the IAM policy to support a new project or to create new LF-Tags.
- **Finer-grained permissions are possible using Lake Formation TBAC.** IAM policies grant access to the top-level resources, such as Data Catalog databases or tables. Using **Lake Formation TBAC**, you can grant access to specific tables or columns that contain specific data values.

> **Note**
>
> IAM tags are not the same as LF-Tags. These tags are not interchangeable. LF-Tags are used to grant Lake Formation permissions and IAM tags are used to define IAM policies.

Managing LF-Tag expressions for metadata access control

LF-Tag expressions are logical expressions composed of one or more LF-Tags (key-value pairs) used to grant permissions on AWS Glue Data Catalog resources. LF-Tag expressions allow you to define rules that govern access to your data resources based on their metadata tags. You can save these expressions and reuse them across multiple permission grants, ensuring consistency and making it straight-forward to manage changes to the tag ontology over time.

Within a given LF-Tag expression, the tag keys are combined using the AND operation, while the values are combined using the OR operation. For example, the tag expression `content_type:Sales AND location:US` represents resources related to sales data in the US.

You can create up to 1000 LF-Tag expressions in an AWS account. These expressions provide a flexible and scalable way to manage permissions based on metadata tags, ensuring that only authorized users or applications can access specific data resources based on the defined tag rules.

LF-Tag expressions offer the following benefits:

- **Reusability** – By defining and saving LF-Tag expressions, you no longer need to manually replicate the same expressions when assigning permissions to other resources or principals.
- **Consistency** – Reusing LF-Tag expressions across multiple permission grants ensures consistency in how permissions are granted and managed.
- **Tag ontology management** – LF-Tag expressions help manage changes to the tag ontology over time, as you can update the saved expressions instead of modifying individual permission grants.

For more information about tag-based access control, please refer to the Lake Formation tag-based access control.

LF-Tag expression creators

LF-Tag expression creator is a principal who has permissions to create and manage LF-Tag expressions. Data lake administrators can add LF-Tag expression creators using the Lake Formation console, CLI, API, or SDK. LF-Tag expression creators have implicit Lake Formation permissions to create, update, and delete LF-Tag expressions, and to grant LF-Tag expression permissions to other principals.

LF-Tag expression creators that are not data lake administrators receive implicit `Alter`, `Drop`, `Describe`, and `Grant with LF-Tag expression` permissions only for expressions they created.

Data lake administrators can also grant LF-Tag expression creators grantable `Create LF-Tag expression` permissions. Then, the LF-Tag expression creator can grant the permission to create LF-Tag expressions to other principals.

Topics

- IAM permissions required to create LF-Tag expressions
- Add LF-Tag expression creators
- Creating LF-Tag expressions
- Updating LF-Tag expressions
- Deleting LF-Tag expressions
- Listing LF-Tag expressions

See also

- Managing LF-Tag value permissions
- Granting data lake permissions using the LF-TBAC method
- Lake Formation tag-based access control

IAM permissions required to create LF-Tag expressions

You must configure permissions to allow a Lake Formation principal to create LF-Tag expressions. Add the following statement to the permissions policy for the principal that needs to be an LF-Tag expression creator.

Note

Although data lake administrators have implicit Lake Formation permissions to create, update, and delete LF-Tags and LF-Tag expressions, to assign LF-Tags to resources, and to grant LF-Tags and LF-Tag expression permission to principals, data lake administrators also need the following IAM permissions.

For more information, see Lake Formation personas and IAM permissions reference.

```
{
"Sid": "Transformational",
"Effect": "Allow",
    "Action": [
        "lakeformation:AddLFTagsToResource",
        "lakeformation:RemoveLFTagsFromResource",
        "lakeformation:GetResourceLFTags",
        "lakeformation:ListLFTags",
        "lakeformation:CreateLFTag",
        "lakeformation:GetLFTag",
        "lakeformation:UpdateLFTag",
        "lakeformation:DeleteLFTag",
        "lakeformation:SearchTablesByLFTags",
        "lakeformation:SearchDatabasesByLFTags",
        "lakeformation:CreateLFTagExpression",
        "lakeformation:DeleteLFTagExpression",
```

```
        "lakeformation:UpdateLFTagExpression",
        "lakeformation:GetLFTagExpression",
        "lakeformation:ListLFTagExpressions",
        "lakeformation:GrantPermissions",
        "lakeformation:RevokePermissions",
        "lakeformation:BatchGrantPermissions",
        "lakeformation:BatchRevokePermissions"
    ]
}
```

Add LF-Tag expression creators

LF-Tag expression creators can create and save reusable LF-Tag expressions, update tag key and values, delete expressions, and grant permissions on Data Catalog resources to principals using LF-TBAC method. The LF-Tag expression creator can also grant these permissions to principals.

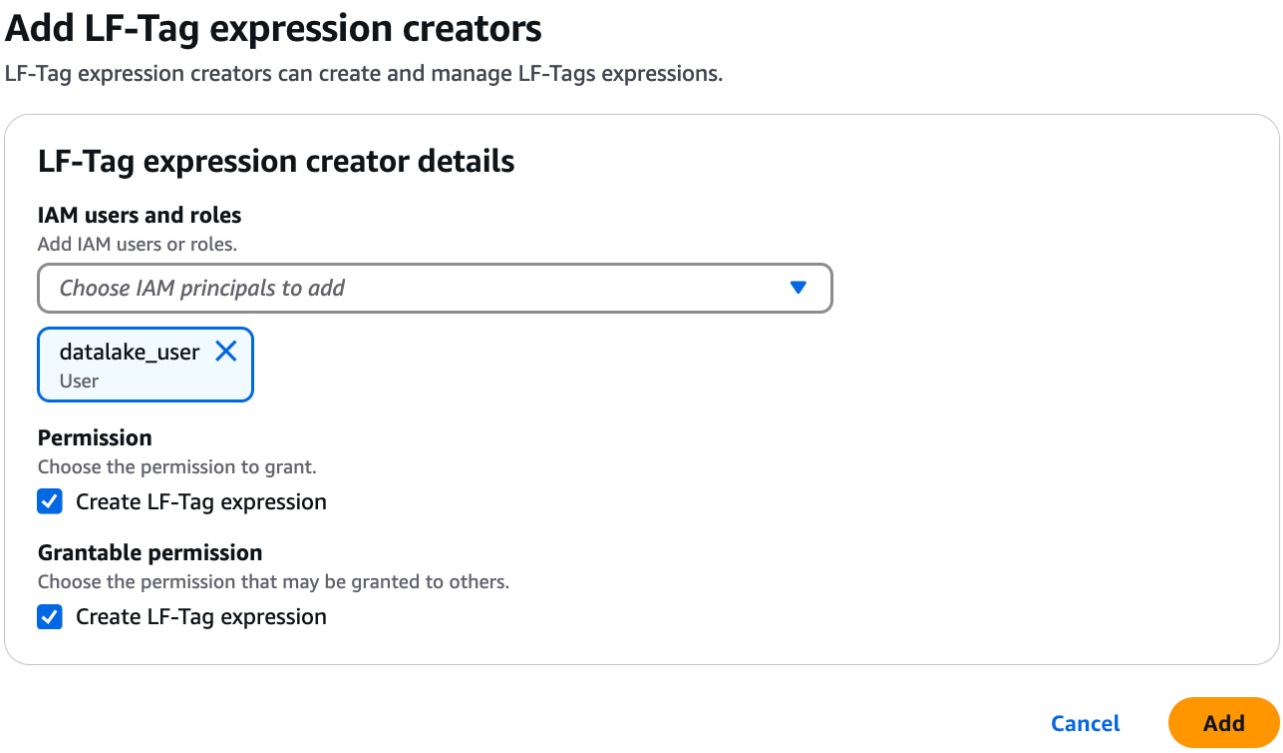
You can create LF-Tag expression creator roles by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

console

To add an LF-Tag expression creator

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator.
2. In the navigation pane, under **Permissions**, choose **LF-Tags and permissions**.
3. Choose the **LF-Tag expressions** tab.
4. In the **LF-Tag expression creators** section, choose **Add LF-Tag expression creators**.



5. On the **Add LF-Tag expression creators** page, choose an IAM role or user who has the required permissions to create LF-Tag expressions.
6. Select `Create LF-Tag expression` permission check box.
7. (Optional) To enable the selected principals to grant `Create LF-Tag expression` permission to principals, choose Grantable `Create LF-Tag expression` permission.
8. Choose **Add**.

AWS CLI

```
aws lakeformation grant-permissions --cli-input-json file://grantCreate
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::123456789012:user/tag-manager"
    },
    "Resource": {
        "Catalog": {}
    },
    "Permissions": [
        "CreateLFTagExpression"
    ],
```

```
    "PermissionsWithGrantOption": [
        "CreateLFTagExpression"
    ]
}
```

The LF-Tag expression creator role gets the ability to create, update, or delete LF-Tag expressions.

Permission	Description
`Create`	A principal with this permission can add LF-Tag expressions in the data lake.
`Drop`	A principal with this permission on an LF-Tag expression can delete an LF-Tag expression from the data lake.
`Alter`	A principal with this permission on an LF-Tag expression can update the expression body of an LF-Tag expression.
`Describe`	A principal with this permission on an LF-Tag expression can view the contents of an LF-Tag expression.
`Grant with LF-Tag expression`	This permission allows the recipient to use the tag expression as the resource when granting data or metadata access permissions. Granting `Grant with LF-Tag expression` implicitly grants `Describe`.
`Super`	For LF-Tag expressions, the `Super` permission grants the ability to `Describe`, `Alter`, `Drop`, and grant permissions on the tag expression to other principals.

These permissions are grantable. A principal who has been granted these permissions with the grant option can grant them to other principals.

Creating LF-Tag expressions

You need to define all LF-Tags in Lake Formation, and assign them to Data Catalog resources before they can be used to create expressions. An LF-Tag expression consists of one more keys and one or more possible values for each key.

After the data lake administrator has set up the required IAM permissions and Lake Formation permissions for the LF-Tag expression creator role, the principal can create reusable LF-Tag expressions. The LF-Tag expression creator gets implicit permissions to update the expression body, and delete the LF-Tag expression.

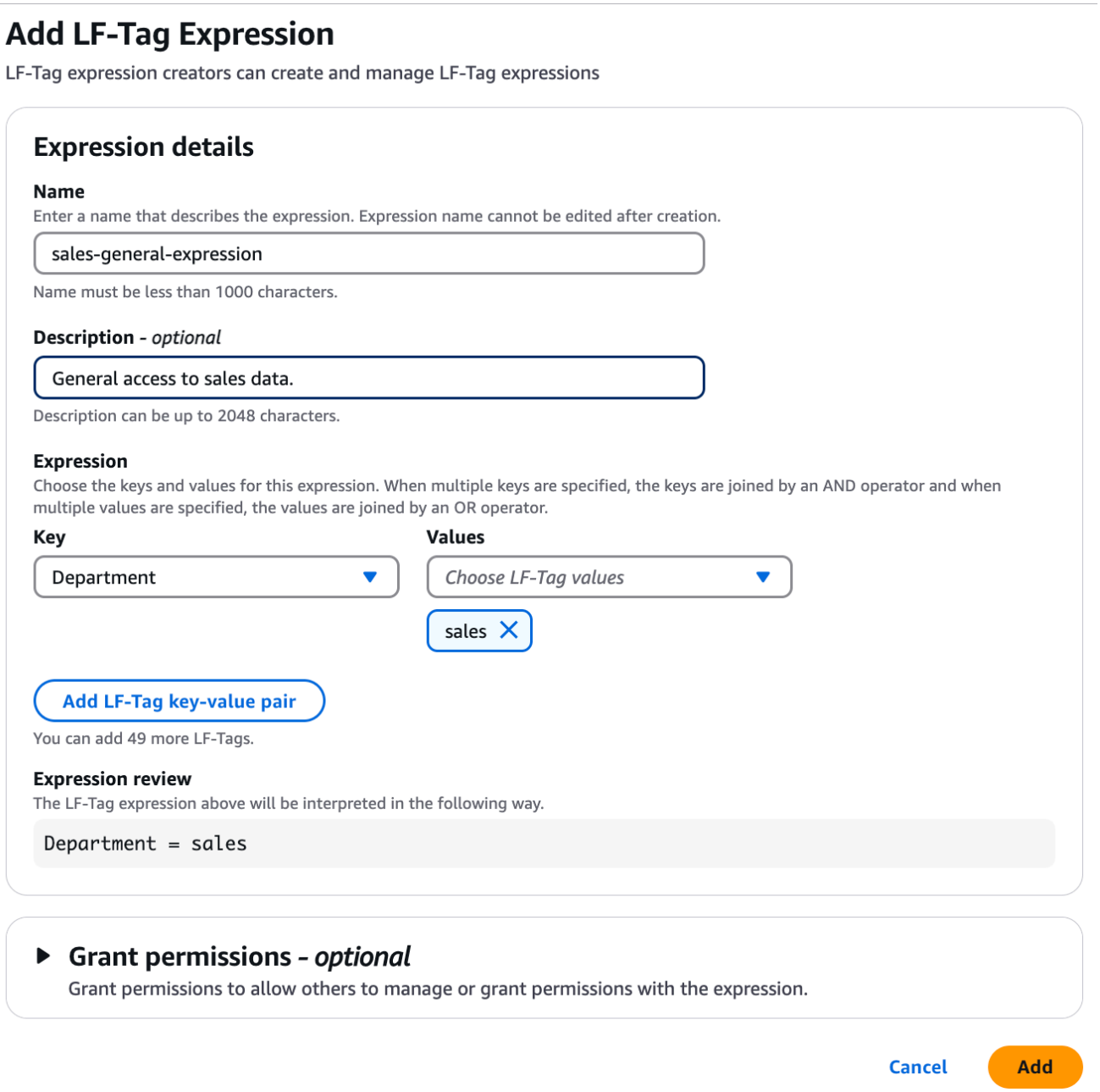
You can create LF-Tag expressions by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Console

To create an LF-Tag expression

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a principal with LF-Tag expression creator permissions or as data lake administrator.
2. In the navigation pane, under **Permissions, choose LF-Tags and permissions**.
3. Choose **LF-Tag expressions**. The **Add LF-Tag expressions** page appears.



4. Enter the following information:

   - Name – Enter a unique name for the expression. You can't update the expression name.
   - Description – Provide an optional description for the expression with the details of the expression.
   - Expression – Create the expression by specifying tag keys and their associated values. You can add up to 50 keys per expression. You must have `Grant with LF-Tags` Lake Formation permission on all tags in expression body.

Each key must have at least one value. To enter multiple values, either enter a comma-delimited list and then press **Enter**, or enter one value at a time and choose **Add** after each one. The maximum number of values permitted per key is 1000.

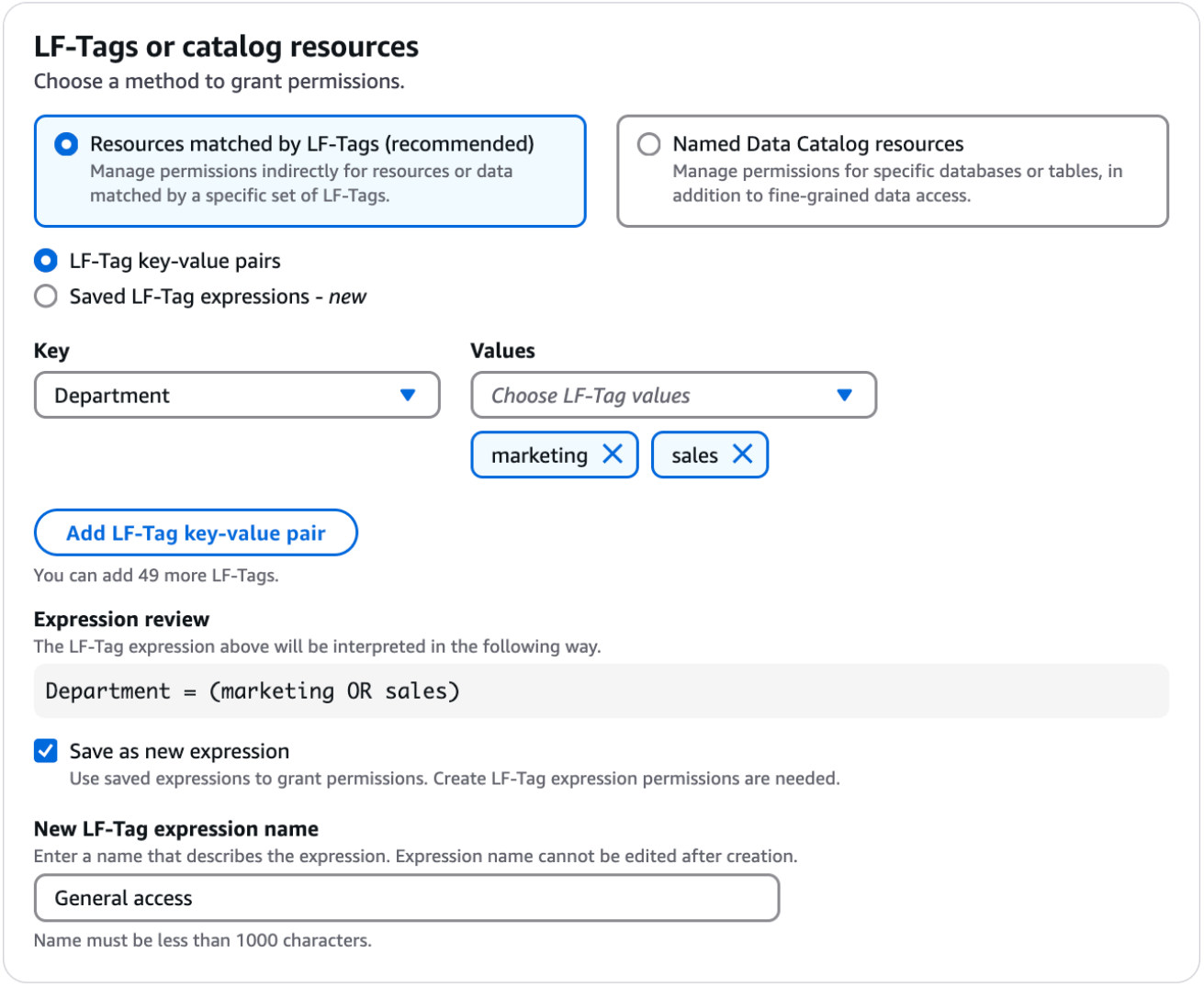
Lake Formation uses the AND/OR logic to combine multiple keys and values in an expression. Within a single (key : list of values) pair, the values are combined using the logical OR operator. For example, if the pair is (Department : [Sales, Marketing]), it means the tag matches if the resource has the Department tag with value Sales OR Marketing.

When you specify multiple keys, the keys are joined by an AND logical operator. So if the full expression is (Department : [Sales, Marketing]) AND (Location : [US, Canada]), it matches resources that have the Department tag with value Sales OR Marketing, AND also have the Location tag with value US OR Canada. The following is another example with multiple keys and values:

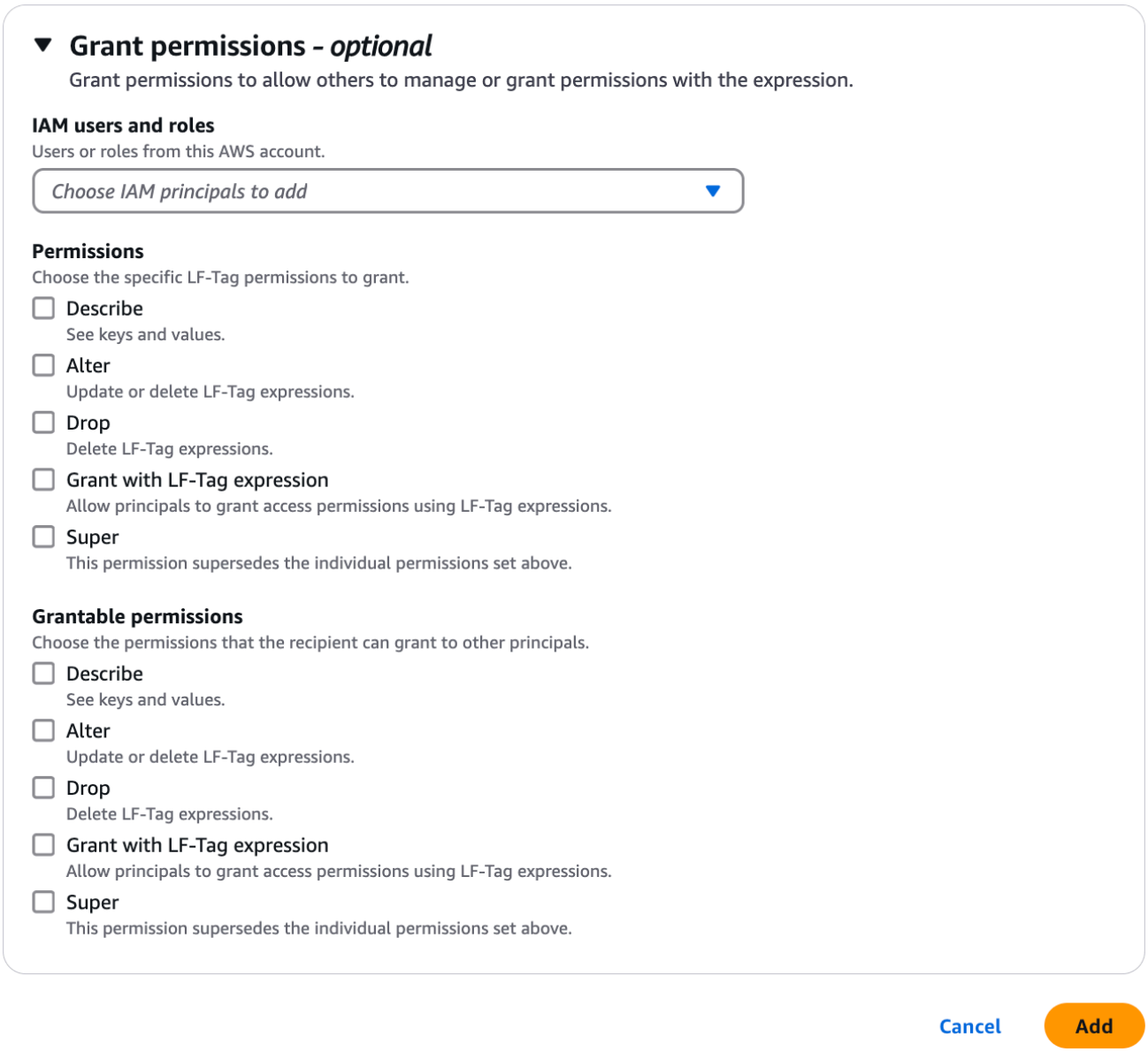
LF-Tag expression: (ContentType : [Video, Audio]) AND (Region : [Europe, Asia]) AND (Department : [Engineering, ProductManagement]).

This expression would match resources that have: - The ContentType tag with value Video OR Audio AND - The Region tag with value Europe OR Asia AND - The Department tag with value Engineering OR ProductManagement.

You can also save a tag expression when granting data lake permissions using LF-Tags. Choose the key and value pairs and choose the **Save as new expression** option. Enter a name that describes the expression.



5. (Optional) Next, choose the users/roles, and the permissions on the expression that you want to grant to them in the account. You can also choose grantable permissions that allows the users to grant these permissions to other users in the account. You can't grant cross account permissions on the tag expressions.



6. Choose **Add** .

AWS CLI

To create an LF-Tag expression

- Enter a `create-lf-tag-expression` command.

  The following example creates an LF-Tag expression with the tag `Department` with values `Sales` and `Marketing`, AND the tag `Location` with the value US.

```
aws lakeformation create-lf-tag-expression \
```

```
-- name "my-tag-expression" \
-- catalog-id "123456789012" \
-- expression '{"Expression":[{"TagKey":"Department","TagValues":
["Sales","Marketing"]},{"TagKey":"Location","TagValues":["US"]}]}'
```

This CLI command creates a new LF-Tag expression in the AWS Glue Data Catalog. The expression can be used grant permissions to Data Catalog resources such as databases, tables, views or columns based on their associated tags. In this example, the expression will match resources that have the `Department` key with values `Sales` or `Marketing`, and the `Location` key with the value US.

As a tag expression creator , the principal gets `Alter` permission on this LF-Tag expression and can update or remove the expression. The LF-Tag expression creator principal can also grant `Alter` permission to another principal to update and remove this expression.

Updating LF-Tag expressions

Only data lake administrators, the LF-Tag expression creator, and principals that have `Alter` or `Super` permission on the LF-Tag expression can update an LF-Tag expression. In addition to `Alter` permission, you also need the `lakeformation:UpdateLFTagExpression` IAM permission and `Grant with LF-Tag` permission on all underlying keys-values on the new expression body to update expressions.

You update an LF-Tag expression by updating the description, expression body and permissions granted on the expression. You can't change the name of the LF-Tag expression. To change the name, delete the LF-Tag expression and add one with the required parameters.

You can update an LF-Tag expression by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Console

To update an LF-Tag expression

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator, the LF-Tag creator or a principal with `Alter` permission on the LF-Tag.
2. In the navigation pane, under permissions, choose **LF-Tags and permissions**.

3. Choose **LF-Tag expressions** tab.
4. On the **LF-Tag expressions** section, select an LF-Tag expression, and then choose **Edit**.
5. In the **Edit LF-Tag expression** dialog box, update the description and update the expression body by adding or removing keys and values.

   To add multiple values, in the **Values** field, choose the values from the drop down.
6. Choose **Save**.

AWS CLI

The update-lf-tag-expression command in Lake Formation allows you to update an existing LF-Tag expression.

```
aws lakeformation update-lf-tag-expression \
-- name expression_name\
-- description new_description \
-- catalog-id catalog_id \
-- expression '{"Expression": [{"TagKey": "tag_key", "TagValues": ["tag_value1",
 "tag_value2", ...]}]}'
```

Here's what the parameters in the provided command mean:

- name – The name of the existing named tag expression that you want to update.
- description – A new description for the expression.

  catalog-id – The ID of the Data Catalog where the named tag expression resides.
- expression – The new tag expression string that you want to update the expression with.

Deleting LF-Tag expressions

You can delete LF-Tag expressions that are no longer in use. If you have granted permissions to principals on Data Catalog resources using the LF-Tag expression, they will no longer have the permissions.

Only data lake administrators, the LF-Tag expression creator, or a principal with `Drop` permission on the LF-Tag expression can delete an LF-Tag expression. In addition to the `Drop` permission, the

principal also needs `lakeformation:DeleteLFTagExpression` IAM permission to delete an LF-Tag expression.

You can delete an LF-Tag expression by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Console

To delete an LF-Tag expression (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator, the LF-Tag expression creator, or a principal that has permissions to delete the expression.
2. In the navigation pane, under **Permissions**, choose **LF-Tags and permissions**.
3. Choose the **LF-Tag expression** tab.
4. On the **LF-Tag expressions** section, select an LF-Tag expression, and then choose **Delete**.
5. In the **Delete LF-Tag expression?** dialog box, to confirm the deletion, enter the LF-Tag expression name in the designated field and then choose **Delete**.

AWS CLI

To delete an LF-Tag (AWS CLI)

- Enter a `delete-lf-tag-expression` command. Provide the expression name and catalog ID to delete.

  Example

  The following example deletes the LF-Tag expression with the name `my-tag-expression` from the Data Catalog with ID `123456789012`. The `catalog-id` parameter is optional if you're using the same account as your AWS CLI configuration. After deleting an LF-Tag expression, Lake Formation cleans up the associated permission records for that expression. This includes both individual permission records and aggregate permission records that contain the deleted expression.

  ```
  aws lakeformation delete-lf-tag-expression \
  --name "my-tag-expression" \
  ```

```
--catalog-id "123456789012"
```

Listing LF-Tag expressions

You can list the LF-Tag expressions that you have the Describe permissions on. Data lake administrators, LF-Tag expression creators, and Read-only administrators implicitly can see all tag expressions in their account.

You can list LF-Tag expressions by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Console

To list LF-Tag expressions (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as the LF-Tag expression creator, as a data lake administrator, or as a principal that has been granted permissions on LF-Tag expressions and that has the `lakeformation:ListLFTagExpressions` IAM permission.
2. In the navigation pane, under **Permissions**, **LF-Tags and permissions**.
3. Choose **LF-Tag expressions** tab to see the expressions. This section shows the information about the existing LF-Tag expressions, including the expression name, the expression itself with links to the included tags, and options to create, edit, or delete expressions.

AWS CLI

To list LF-Tags (AWS CLI)

- To list LF-Tag expressions using the AWS CLI, you can use the list-lf-tag-expressions command. The request syntax is:

  ```
  aws lakeformation list-lf-tag-expressions \
  -- catalog-id "123456789012" \
  -- max-items "100" \
  -- next-token "next-token"
  ```

  Where:

- `catalog-id` is the AWS account ID of the Data Catalog you want to list tag expressions for .
- `max-items` specifies the maximum number of tag expressions to return. If this parameter is not used, the default value is 100.
- `next-token` is a continuation token if the results were truncated in a previous request.

The response will include a list of LF-Tag expressions and a next token if applicable.

Managing LF-Tag value permissions

You can grant the `Drop`, `Alter` permissions on LF-Tags to principals to manage LF-Tag value expressions. You can also grant `Describe`, `Associate`, and `Grant with LF-Tag expressions` permissions on LF-Tags to principals to view the LF-Tags and assign them to Data Catalog resources (databases, tables, and columns). When LF-Tags are assigned to Data Catalog resources, you can use the Lake Formation tag-based access control (LF-TBAC) method to secure those resources. For more information, see Lake Formation tag-based access control.

You can grant these permissions with the grant option so that other principals can grant them. The `Grant with LF-Tag expressions`, `Describe`, and `Associate` permissions are explained in Add LF-Tag creators.

You can grant the `Describe` and `Associate` permissions on a LF-Tag to an external AWS account. A data lake administrator in that account can then grant those permissions to other principals in the account. Principals to whom the data lake administrator in the external account grants the `Associate` permission can then assign LF-Tags to Data Catalog resources that you shared with their account.

When granting to an external account, you must include the grant option.

You can grant permissions on LF-Tags by using the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Topics

- Listing LF-Tag permissions using the console
- Granting LF-Tag permissions using the console
- Managing LF-Tag permissions using the AWS CLI

For more information see Managing LF-Tags for metadata access control and Lake Formation tag-based access control.

Listing LF-Tag permissions using the console

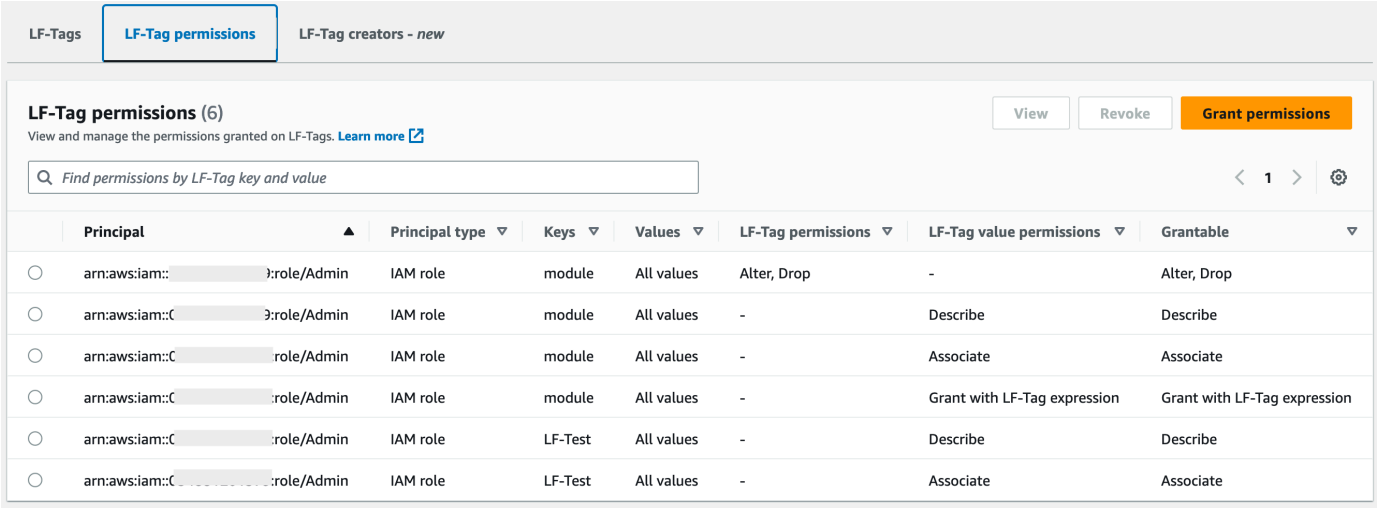
You can use the Lake Formation console to view the permissions granted on LF-Tags. You must be a LF-Tag creator, a data lake administrator, or have the `Describe` or `Associate` permission on a LF-Tag to see it.

To list LF-Tag permissions (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as the LF-Tag creator, a data lake administrator, or as a user to whom the `Drop`, `Alter`, `Associate`, or `Describe` permissions on LF-Tags have been granted.
2. In the navigation pane, under **Permissions**, choose **LF-Tags and permissions**, and choose **LF-Tag permissions** section.

   The **LF-Tag permissions** section shows a table that contains principal, tag keys, values, and permissions.



Granting LF-Tag permissions using the console

The following steps explain how to grant permissions on LF-Tags by using the **Grant LF-Tag permissions** page on the Lake Formation console. The page is divided into these sections:

- **Permission types** – The type of permission to grant.
- **Principals** – The IAM users or roles, or SAML users or roles to grant permissions to.

- **LF-Tag key-value pair permissions** permissions – The LF-Tag key-value pairs to grant permissions on.
- **LF-Tag permissions** – The LF-Tags to grant permissions on.
- **LF-Tag expression permissions** permissions – The LF-Tags to grant permissions on.
- **Permissions** – The permissions to grant.

Open the Grant LF-Tag permissions page

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as the LF-Tag creator, a data lake administrator, or as a user LF-Tag permissions or LF-Tag key-value pair permissions on LF-Tags have been granted with the `Grant` option.
2. In the navigation pane, choose **LF-Tags and permissions**, choose **LF-Tag permissions** section.
3. Choose **Grant permissions**.

Specify the permissions type

In the **Permissions type** section, choose a permissions type.

LF-Tag permissions

Choose the **LF-Tag permissions** to allow principals to update LF-Tag values or delete LF-Tags.

LF-Tag key-value pair permissions

Choose the **LF-Tag key-value pair permissions** to allow principals to assign LF-Tags to Data Catalog resources, view LF-Tags and values, and grant LF-Tags based permissions on Data Catalog resources to principals.

The options available in the following sections depend on the **Permissions type**.

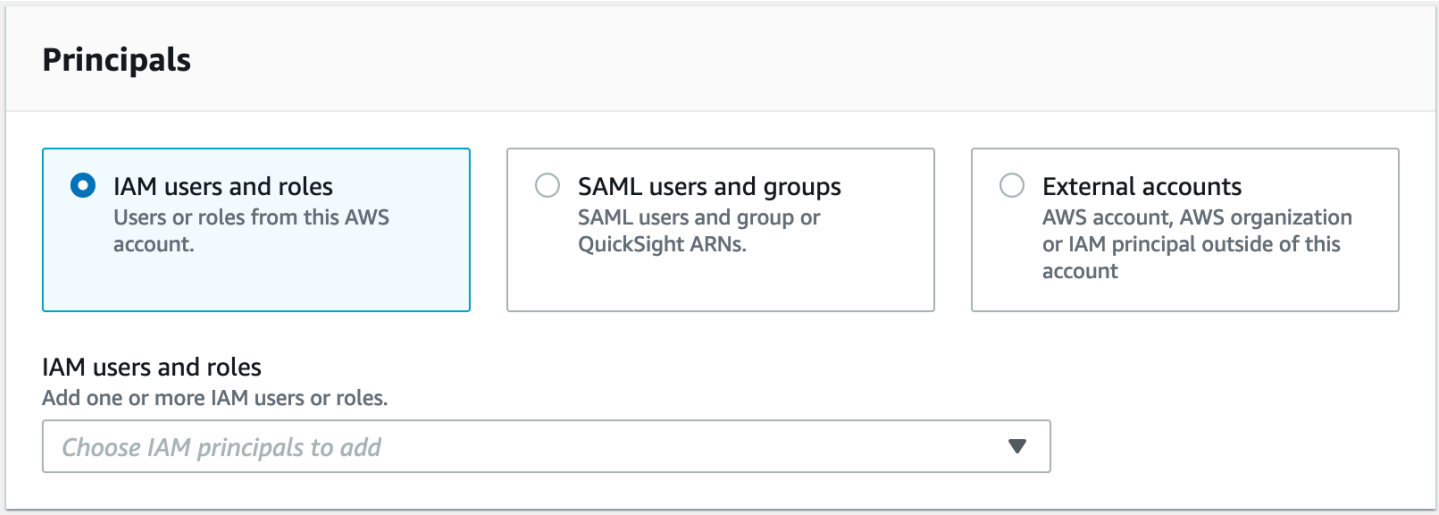
LF-Tag expression permissions

Choose the **LF-Tag expression permissions** to allow principals to update expressions or delete expressions.

Specify the principals

> **Note**
>
> You can't grant LF-Tag permissions (`Alter` and `Drop`) to external accounts or principals in another account.

In the **Principals** section, choose a principal type and specify principals to grant permissions to.



IAM users and roles

Choose one or more users or roles from the **IAM users and roles** list.

SAML users and groups

For **SAML and Quick users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Quick users or groups. Press **Enter** after each ARN.

For information about how to construct the ARNs, see Lake Formation grant and revoke AWS CLI commands.

> **Note**
>
> Lake Formation integration with Quick is supported for Quick Enterprise Edition only.

External accounts

For **AWS account**, enter one or more valid AWS account IDs. Press **Enter** after each ID.

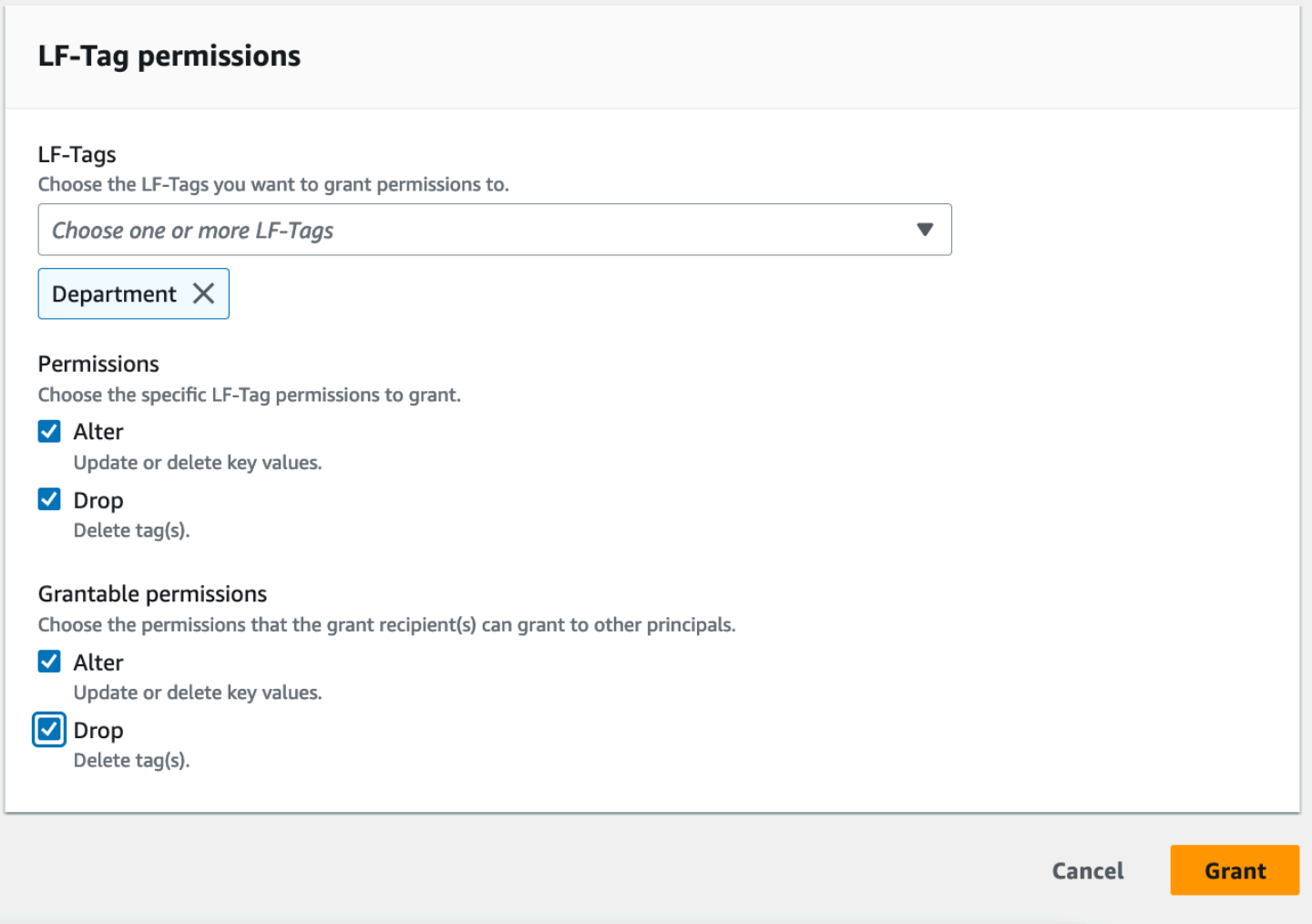
An organization ID consists of "o-" followed by 10 to 32 lower-case letters or digits.

An organizational unit ID starts with "ou-" followed by 4 to 32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

For IAM principal, enter the ARN for the IAM user or role.

Specify the LF-Tags

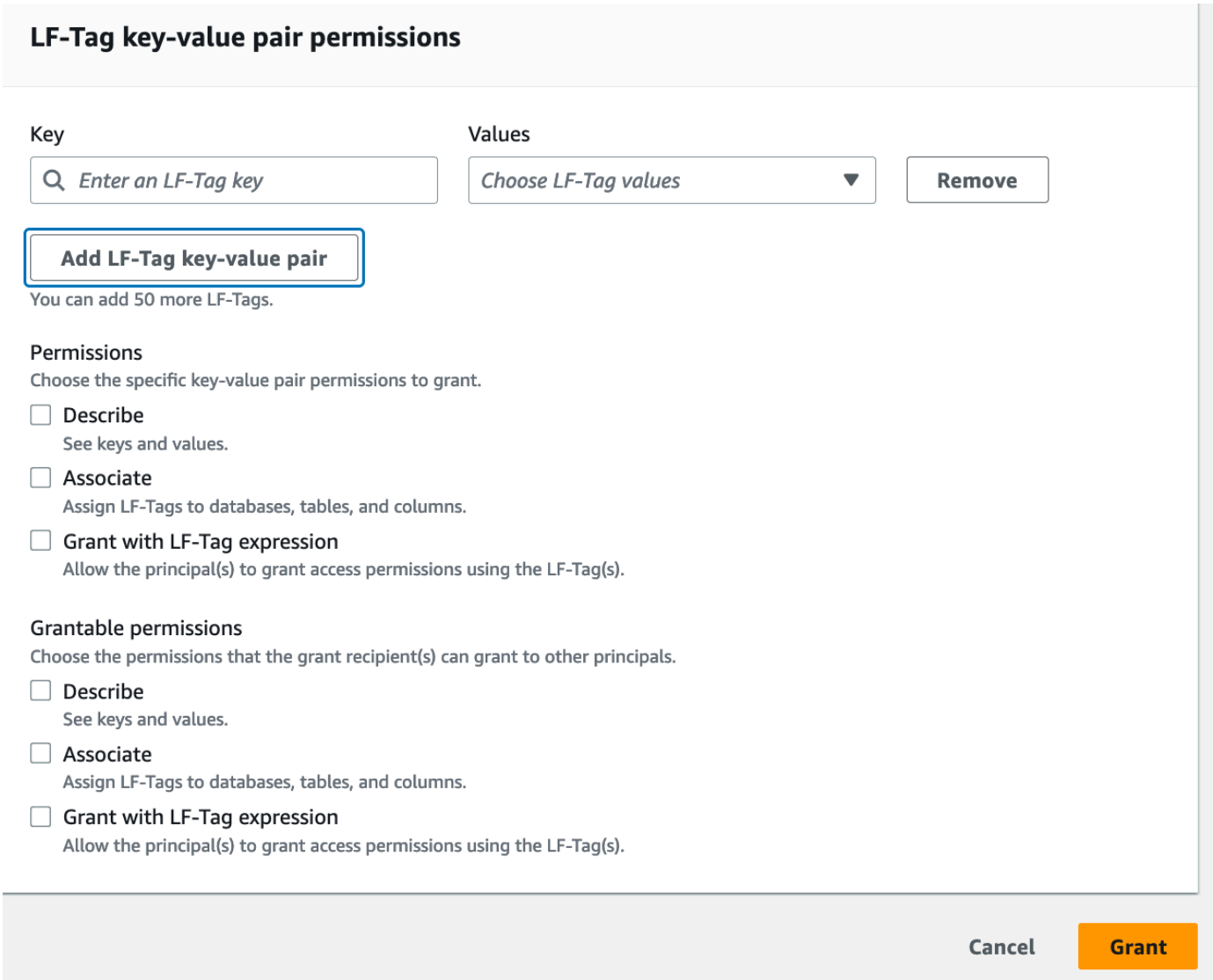
To grant permissions on LF-Tags, in the **LF-Tag permissions** section, specify the LF-Tags to grant permissions on.



- Choose one or more LF-Tag using the drop-down.

Specify the LF-Tag key-value pairs

1. To grant permissions on LF-Tag key-value pairs, (you need to first choose choose **LF-Tag key-value pair permissions** as the **Permission type**) choose **Add LF-Tag key-value pair** to reveal the first row of fields for specifying LF-Tag key and values.



2. Position the cursor in the **Key** field, optionally start typing to narrow down the selection list, and select a LF-Tag key.
3. In the **Values** list, select one or more values, and then press **Tab** or click or tap outside the field to save the selected values.

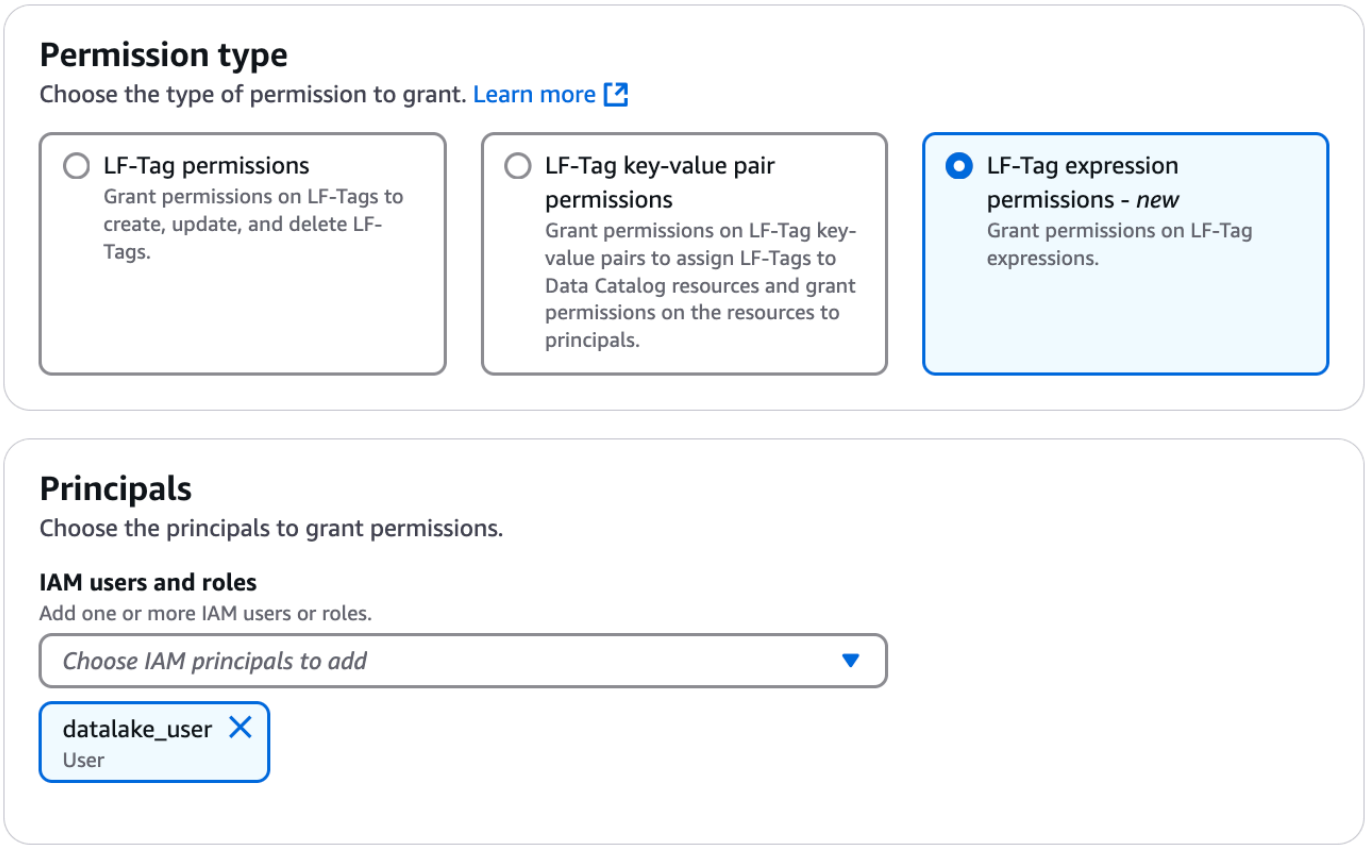
> **Note**
>
> If one of the rows in the **Values** list has focus, pressing **Enter** selects or clears the check box.

The selected values appear as tiles below the **Values** list. Choose the ✖ to remove a value. Choose **Remove** to remove the entire LF-Tag.

4. To add another LF-Tag, choose **Add LF-Tag** again, and repeat the previous two steps.

Specify the LF-Tag expressions

1. To grant permissions on LF-Tag expressions, (you need to first choose choose **LF-Tag expression permissions** as the **Permission type**).



2. Choose a LF-Tag expression.
3. The selected expressions appear as tiles below the **LF-Tag expressions** list. Choose the ✖ to remove an expression.

4. To add another LF-Tag expression, choose another expression.

Specify the permissions

This section shows either the **LF-Tag permissions** or the **LF-Tag value permissions** based on the **Permission type** you chose in the previous step.

Depending on the **Permission type** you chose to grant, select the **LF-Tag permissions** or **LF-Tag key-value pair permissions**, and grantable permissions.

1. Under **LF-Tag permissions**, select the permissions to grant.

   Granting **Drop** and **Alter** implicitly grants **Describe**.

   You need to grant **Alter** and **Drop** permissions on all tag values.
2. Under **LT-Tag key-value value permissions**, select the permissions to grant.

   Granting **Associate** implicitly grants **Describe**. Choose **Grant with LF-Tag expression** to allow the grant recipient to grant or revoke access permissions on Data Catalog resources using LF-TBAC method.
3. Under **LF-Tag expression permissions**, select the permissions to grant.

   Granting **Drop** and **Alter** implicitly grants **Describe**.

   Granting **Super** permission, grants all available permissions.
4. (Optional) Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account.
5. Choose **Grant**.

Managing LF-Tag permissions using the AWS CLI

You can grant, revoke, and list permissions on LF-Tags by using the AWS Command Line Interface (AWS CLI).

To list LF-Tag permissions (AWS CLI)

- Enter a `list-permissions` command. You must be the LF-Tag creator, a data lake administrator, or have the `Drop`, `Alter`, `Describe`, `Associate`, `Grant with LF-Tag permissions` permission on a LF-Tag to see it.

The following command requests all LF-Tags that you have permissions on.

```
aws lakeformation list-permissions --resource-type LF_TAG
```

The following is sample output for a data lake administrator, who sees all LF-Tags granted to all principals. Non-administrative users see only LF-Tags granted to them. LF-Tag permissions granted from an external account appear on a separate results page. To see them, repeat the command and supply the `--next-token` argument with the token returned from the previous command run.

```
{
    "PrincipalResourcePermissions": [
        {
            "Principal": {
                "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/
datalake_admin"
            },
            "Resource": {
                "LFTag": {
                    "CatalogId": "111122223333",
                    "TagKey": "environment",
                    "TagValues": [
                        "*"
                    ]
                }
            },
            "Permissions": [
                "ASSOCIATE"
            ],
            "PermissionsWithGrantOption": [
                "ASSOCIATE"
            ]
        },
        {
            "Principal": {
                "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/
datalake_user1"
            },
            "Resource": {
                "LFTag": {
                    "CatalogId": "111122223333",
```

```
                    "TagKey": "module",
                    "TagValues": [
                        "Orders",
                        "Sales"
                    ]
                }
            },
            "Permissions": [
                "DESCRIBE"
            ],
            "PermissionsWithGrantOption": []
        },
...
    ],
    "NextToken": "eyJzaG91bGRRdWVy...Wlzc2lvbnMiOnRydWV9"
}
```

You can list all grants for a specific LF-Tag key. The following command returns all permissions granted on the LF-Tag `module`.

```
aws lakeformation list-permissions --resource-type LF_TAG --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

You can also list LF-Tag values granted to a specific principal for a specific LF-Tag. When supplying the `--principal` argument, you must supply the `--resource` argument. Therefore, the command can only effectively request the values granted to a specific principal for a specific LF-Tag key. The following command shows how to do this for the principal `datalake_user1` and the LF-Tag key `module`.

```
aws lakeformation list-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/
datalake_user1 --resource-type LF_TAG --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

The following is sample output.

```
{
    "PrincipalResourcePermissions": [
        {
            "Principal": {
```

```
                "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/
datalake_user1"
            },
            "Resource": {
                "LFTag": {
                    "CatalogId": "111122223333",
                    "TagKey": "module",
                    "TagValues": [
                        "Orders",
                        "Sales"
                    ]
                }
            },
            "Permissions": [
                "ASSOCIATE"
            ],
            "PermissionsWithGrantOption": []
        }
    ]
}
```

To grant permissions on LF-Tags (AWS CLI)

1. Enter a command similar to the following. This example grants to user `datalake_user1` the `Associate` permission on the LF-Tag with the key `module`. It grants permissions to view and assign all values for that key, as indicated by the asterisk (*).

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/
datalake_user1 --permissions "ASSOCIATE" --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

   Granting the `Associate` permission implicitly grants the `Describe` permission.

   The next example grants `Associate` to the external AWS account 1234-5678-9012 on the LF-Tag with the key `module`, with the grant option. It grants permissions to view and assign only the values `sales` and `orders`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=123456789012 --permissions "ASSOCIATE"
```

```
 --permissions-with-grant-option "ASSOCIATE" --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["sales", "orders"]}}'
```

2. Granting the `GrantWithLFTagExpression` permission implicitly grants the `Describe` permission.

   The next example grants `GrantWithLFTagExpression` to a user on the LF-Tag with the key `module`, with the grant option. It grants permissions to view and grant permissions on Data Catalog resources using only the values `sales` and `orders`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333 --permissions "GrantWithLFTagExpression"
 --permissions-with-grant-option "GrantWithLFTagExpression" --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["sales", "orders"]}}'
```

3. The next example grants `Drop` permissions to a user on the LF-Tag with the key `module`, with the grant option. It grants permissions to delete the LF-Tag. To delete a LF-Tag, you need permissions on all values for that key.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333 --permissions "DROP"
 --permissions-with-grant-option "DROP" --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

4. The next example grants `Alter` permissions to the user on the LF-Tag with the key `module`, with the grant option. It grants permissions to delete the LF-Tag. To update a LF-Tag, you need permissions on all values for that key.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=111122223333 --permissions "ALTER"
 --permissions-with-grant-option "ALTER" --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

To revoke permissions on LF-Tags (AWS CLI)

- Enter a command similar to the following. This example revokes the `Associate` permission on the LF-Tag with the key `module` from user `datalake_user1`.

```
aws lakeformation revoke-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/
```

```
datalake_user1 --permissions "ASSOCIATE" --resource '{ "LFTag":
 {"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

Granting data lake permissions using the LF-TBAC method

You can grant the DESCRIBE and ASSOCIATE Lake Formation permissions on LF-Tags to principals so that they can view the LF-Tags and assign them to Data Catalog resources (databases, tables, views, and columns). When LF-Tags are assigned to Data Catalog resources, you can use the Lake Formation tag-based access control (LF-TBAC) method to secure those resources. For more information, see Lake Formation tag-based access control.

At first, only the data lake administrator can grant these permissions. If the data lake administrator grants these permissions with the grant option, other principals can grant them. The DESCRIBE and ASSOCIATE permissions are explained in Lake Formation tag-based access control best practices and considerations.

You can grant the DESCRIBE and ASSOCIATE permissions on a LF-Tag to an external AWS account. A data lake administrator in that account can then grant those permissions to other principals in the account. Principals to whom the data lake administrator in the external account grants the ASSOCIATE permission can then assign LF-Tags to Data Catalog resources that you shared with their account.

When granting to an external account, you must include the grant option.

You can grant permissions on LF-Tags by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

> **Note**
>
> The following steps are not needed for S3 Tables catalogs. You can use LF-Tags to grant permissions on existing S3 Tables catalogs without deleting and recreating them.

Enabling LF-Tags support for existing federated catalogs that uses Lake Formation permissions

Follow these steps, if you have existing federated catalogs that are using Lake Formation permissions, such as Amazon Redshift or Amazon DynamoDB catalogs that were created before LF-Tags support was available for federated catalogs.

1. Delete the existing catalog – Call the `deleteCatalog` API operation to remove the existing federated catalog that uses Lake Formation permissions.
2. Create a new federated catalog – Create a new catalog and point the new catalog to your existing namespace/datashare.

   Use a new name for the catalog – This process updates your pre-existing federated catalogs to support LF-Tag functionality. If you want to use the same catalog name, contact AWS support team for assistance.

Topics

- Granting Data Catalog permissions

See also

- Managing LF-Tag value permissions
- Managing LF-Tags for metadata access control
- Lake Formation tag-based access control

Granting Data Catalog permissions

Use the Lake Formation console or AWS CLI to grant Lake Formation permissions on Data Catalog databases, tables, views, and columns using the Lake Formation tag-based access control (LF-TBAC) method.

Console

The following steps explain how to grant permissions by using the Lake Formation tag-based access control (LF-TBAC) method and the **Grant data lake permissions** page on the Lake Formation console. The page is divided into the following sections:

- **Principals** – The users, roles, and AWS accounts to grant permissions to.
- **LF-Tags or catalog resources** – The databases, tables, or resource links to grant permissions on.
- **Permissions** – The Lake Formation permissions to grant.

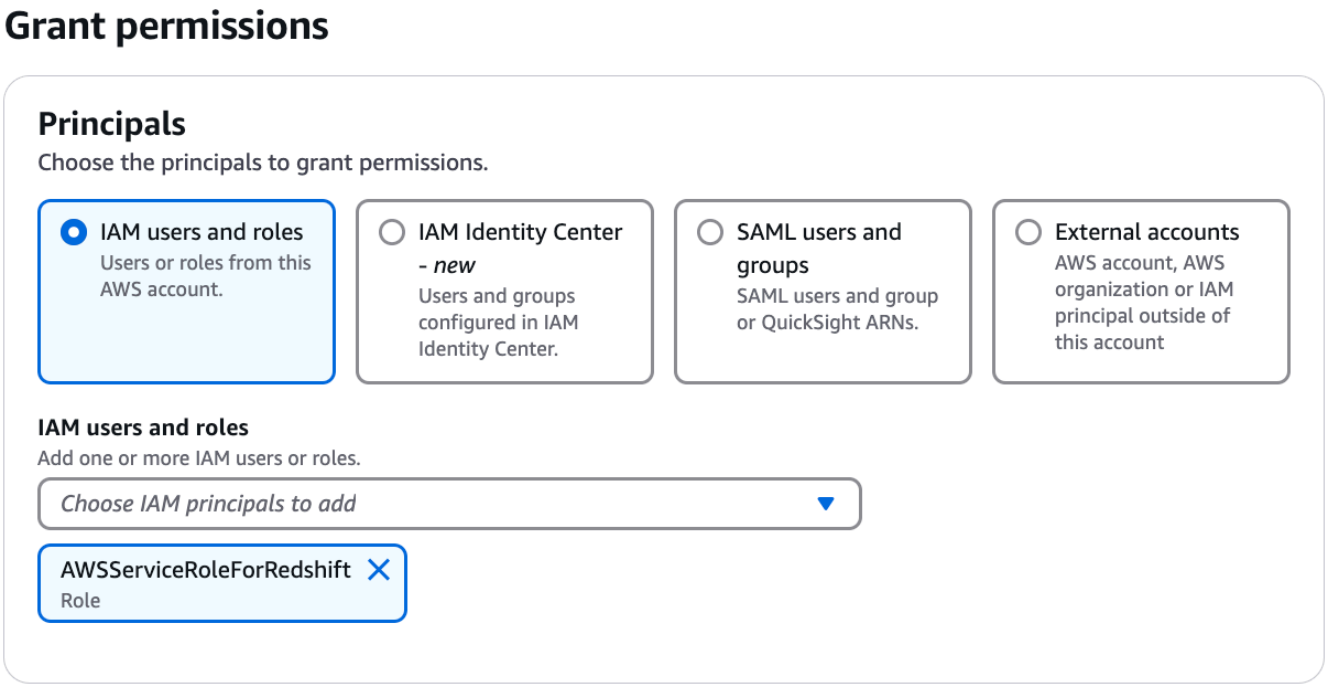
1. **Open the Grant data lake permissions page.**

   Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator or as a user who has been granted Lake Formation permissions on Data Catalog resources through LF-TBAC with the grant option.

   In the navigation pane, under **Permissions**, choose **Data lake permissions**. Then choose **Grant**.

2. **Specify the principals.**

   In the **Principals** section, choose a principal type and then specify principals to grant permissions to.



IAM users and roles

Choose one or more users or roles from the **IAM users and roles** list.

IAM Identity Center

Choose one or more users or from the **Users and groups** list.

SAML users and groups

For **SAML and Quick users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Quick users or groups. Press Enter after each ARN.

For information about how to construct the ARNs, see Lake Formation grant and revoke AWS CLI commands.

> **Note**
>
> Lake Formation integration with Quick is supported for Quick Enterprise Edition only.

External accounts

For **AWS accounts, AWS organization**, or **IAM principal** enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or ARN for the IAM user or role. Press **Enter** after each ID.

An organization ID consists of "o-" followed by 10 to 32 lower-case letters or digits.

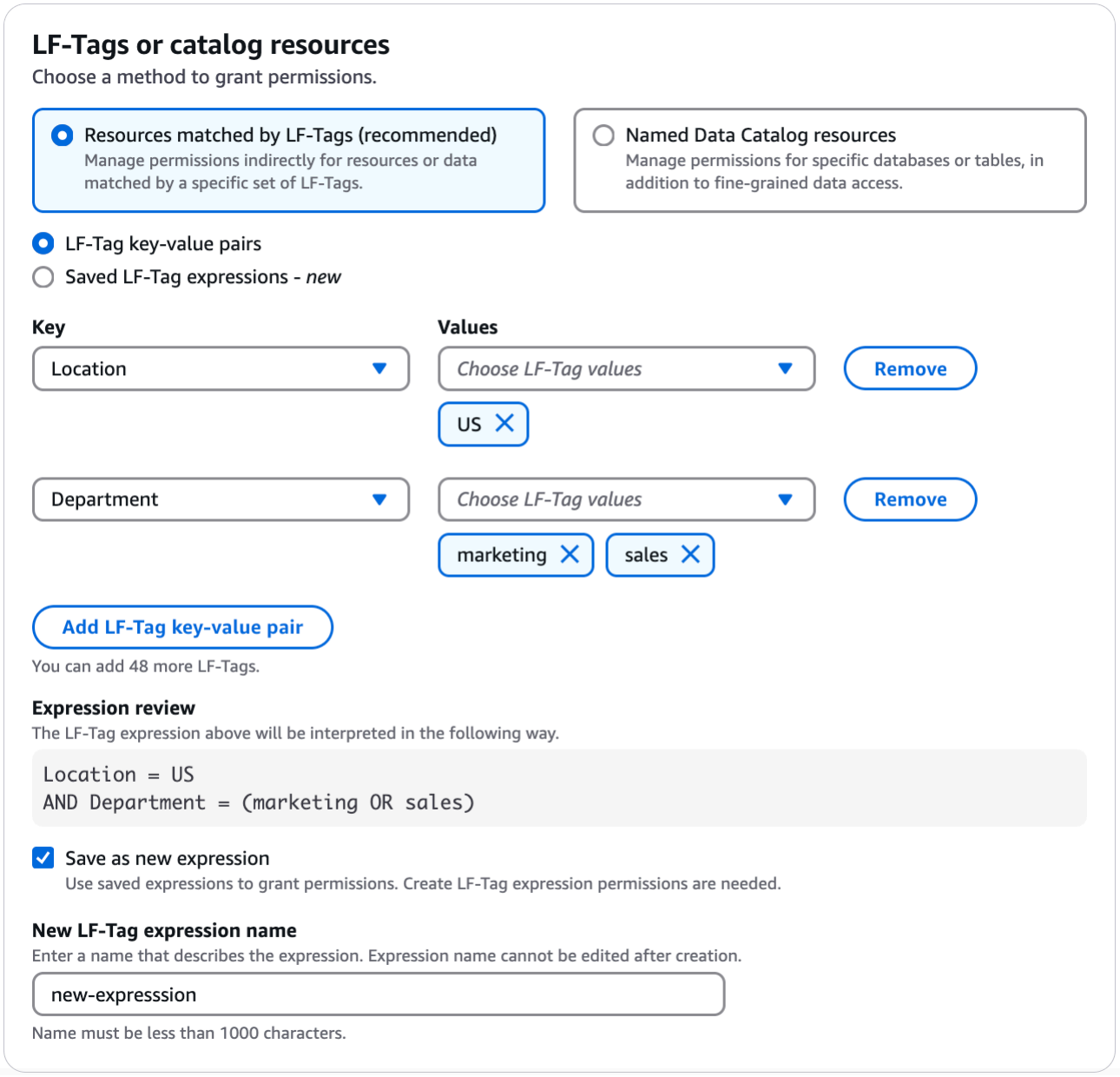
An organizational unit ID starts with "ou-" followed by 4 to 32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

3. **Specify the LF-Tags.**

Ensure that the **Resources matched by LF-Tags** option is chosen. Choose **LF-Tag key-value pairs** or **Saved LF-Tag expressions**.

1. If you choose the **LF-Tag key-value pairs** option, choose the keys and values.

   If you choose more than one value, you are creating a LF-Tag expression with an OR operator. This means that if any of the LF-Tag values match a LF-Tag assigned to a Data Catalog resource, you are granted permissions on the resource.



2. (Optional) Choose **Add LF-Tag key-value pair** again to specify another LF-Tag.

   If you specify more than one LF-Tag, you are creating a LF-Tag expression with an AND operator. The principal is granted permissions on a Data Catalog resource only if the resource was assigned a matching LF-Tag for each LF-Tag in the LF-Tag expression.

3. Choose **Save as a new expression** option to reuse the expression.

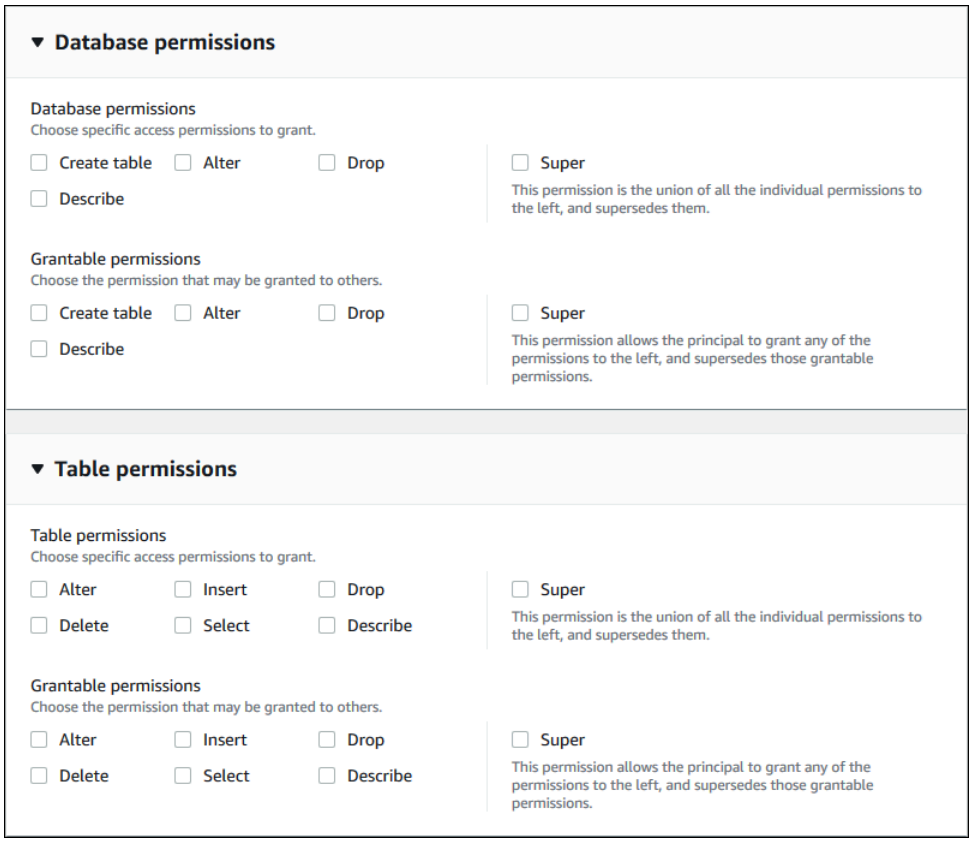
   You need `Create LF-Tag expression` to save expressions.

   For more information about LF-Tag expressions, see Managing LF-Tag expressions for metadata access control.

4. **Specify the permissions.**

   Specify the permissions that you want to grant the principal on matching Data Catalog resources. Matching resources are those resources that were assigned LF-Tags that match one of the LF-Tag expressions granted to the principal.

   You can specify the permissions to grant on matching databases, matching tables, and matching views.



   Under **Database permissions**, select the database permissions to grant to the principal on matching databases.

   Under **Table permissions**, select the table or view permissions to grant to the principal on matching tables and views.

   You can also choose `Select`, `Describe`, and `Drop` permissions from the **Table permissions** to apply on views.

5. Choose **Grant**.

AWS CLI

You can use the AWS Command Line Interface (AWS CLI) and the Lake Formation tag-based access control (LF-TBAC) method to grant Lake Formation permissions on Data Catalog databases, tables, and columns.

Granting data lake permissions using the AWS CLI and the LF-TBAC method

- Use the `grant-permissions` command.

  Example

  The following example grants the LF-Tag expression "`module=*`" (all values of the LF-Tag key `module`) to user `datalake_user1`. That user will have the `CREATE_TABLE` permission on all matching databases—databases that have been assigned the LF-Tag with the key `module`, with any value.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/
datalake_user1 --permissions "CREATE_TABLE" --resource '{ "LFTagPolicy":
 {"CatalogId":"111122223333","ResourceType":"DATABASE","Expression":
[{"TagKey":"module","TagValues":["*"]}]}}'
```

  Example

  The next example grants the LF-Tag expression "`(level=director) AND (region=west OR region=south)`" to user `datalake_user1`. That user will have the `SELECT`, `ALTER`, and `DROP` permissions with the grant option on matching tables—tables that have been assigned both `level=director` and (`region=west` or `region=south`).

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/
datalake_user1 --permissions "SELECT" "ALTER" "DROP" --permissions-
with-grant-option "SELECT" "ALTER" "DROP" --resource '{ "LFTagPolicy":
 {"CatalogId":"111122223333","ResourceType":"TABLE","Expression": [{"TagKey":
 "level","TagValues": ["director"]},{"TagKey": "region","TagValues": ["west",
 "south"]}]}}'
```

Example

This next example grants the LF-Tag expression "`module=orders`" to the AWS account 1234-5678-9012. The data lake administrator in that account can then grant the "`module=orders`" expression to principals in their account. Those principals will then have the `CREATE_TABLE` permission on matching databases owned by account 1111-2222-3333 and shared with account 1234-5678-9012 by using either the named resource method or the LF-TBAC method.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=123456789012 --permissions "CREATE_TABLE" --
permissions-with-grant-option "CREATE_TABLE" --resource '{ "LFTagPolicy":
 {"CatalogId":"111122223333","ResourceType":"DATABASE","Expression":
[{"TagKey":"module","TagValues":["orders"]}]}}'
```

Attribute-based access control

In AWS Lake Formation, you can grant access on AWS Glue Data Catalog objects such as catalogs, databases, tables, and data filters using attributes that are IAM tags and session tags associated with IAM entities such as roles and users.

For more information about using session tags, see assume-role in the AWS CLI user guide.

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. AWS calls these attributes *tags*. You can use ABAC to grant access to principals within the same account or in another account on the Data Catalog resources. Any IAM principal with matching IAM tag or session tag keys and values gains access to the resource. You must have grantable permissions on the resources to make these grants.

ABAC allows you to grant access to multiple users at the same time. When new users join the organization, their access to data can be automatically determined based on their attributes, such as their job function or department, without requiring administrators to manually assign specific roles or permissions. By using attributes instead of roles, ABAC provides a more streamlined and maintainable way to manage data access across diverse systems and environments, ultimately enhancing data governance and compliance.

For more information about defining attributes, see Define permissions based on attributes with ABAC authorization.

For information on limitations, considerations, and supported AWS Regions, see Attribute-based access control considerations, limitations, and supported regions.

Topics

- Prerequisites for granting permissions using attributes
- Granting permissions using attribute-based access control

Prerequisites for granting permissions using attributes

To grant permissions using attribute-based access control (ABAC), you must complete the following prerequisites:

- Update the **Data Catalog settings** to enable Lake Formation permissions for Data Catalog objects. For more information, see the Change the default permission model or use hybrid access mode section.
- Set the cross-account version settings to two or higher.
- Attach attributes to the IAM entities that require access.
- Only a data lake administrator or an IAM user with the required permissions can grant access on Data Catalog objects. For more information on required permissions, see IAM permissions.

Granting permissions using attribute-based access control

This topic describes the steps you need to follow to grant attribute-based access permissions on Data Catalog resources. You can use the Lake Formation console or the AWS Command Line Interface (AWS CLI).

Granting permissions using ABAC (AWS Management Console)

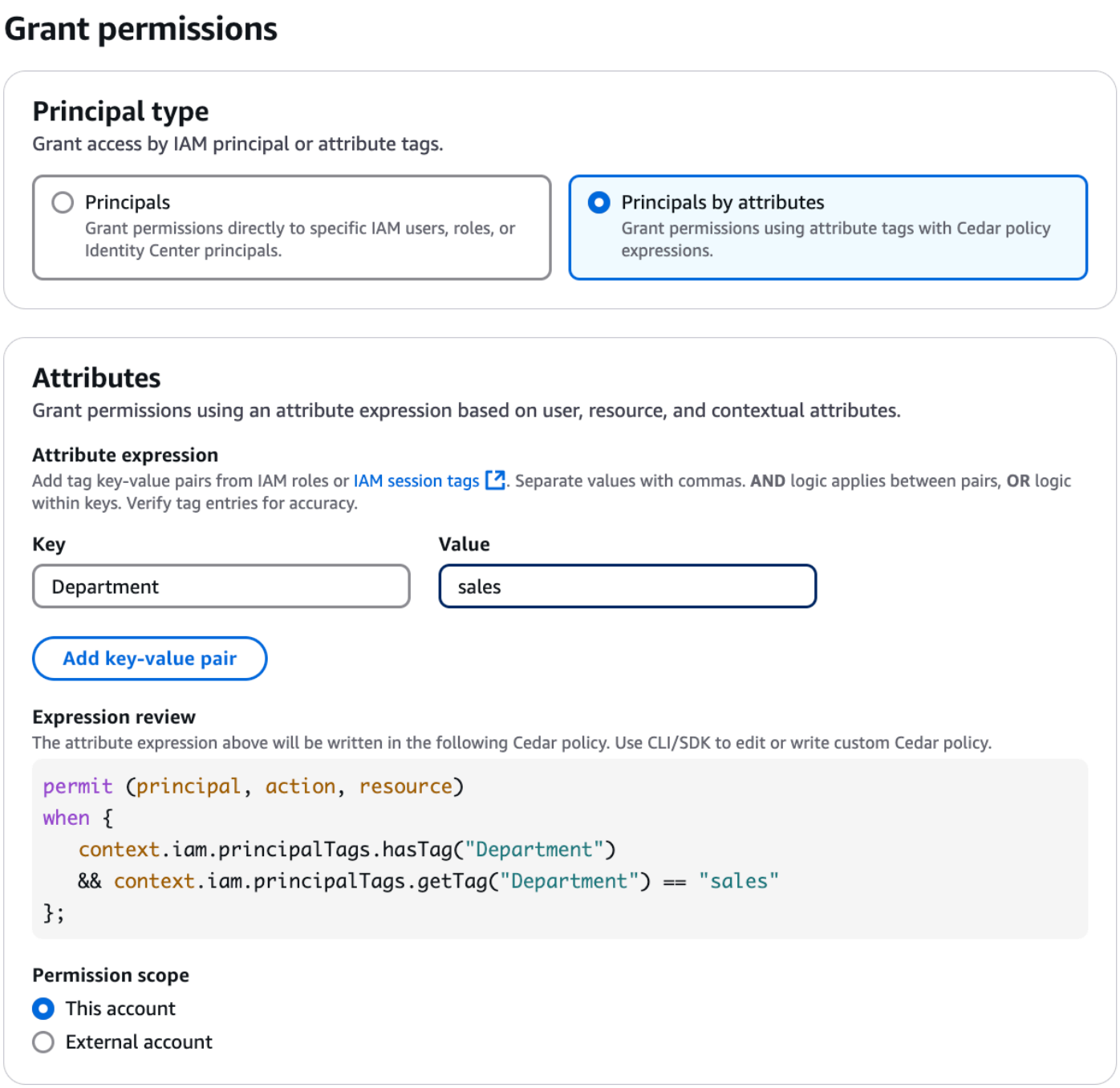
1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator, the resource creator, or an IAM user who has **Grantable permissions** on the resource.
2. Do one of the following:
   - In the navigation pane, under **Permissions**, choose **Data lake permissions**. Then choose **Grant**.

- In the navigation pane, choose **Catalogs** under **Data Catalog**. Then, choose a catalog object (catalogs, databases, tables, and data filters), and from the **Actions** menu under **Permissions**, and choose **Grant**.

3. On the **Grant permissions** page, choose **Principals by attribute**.
4. Specify the attribute key and value(s). If you choose more than one value, you are creating an attribute expression with an `OR` operator. This means that if any of the attribute tag values assigned to an IAM role or user match, the role/user gains access permissions on the resource.

   If you specify more than one attribute tag, you are creating an attribute expression with an `AND` operator. The principal is granted permissions on a Data Catalog resource only if the IAM role/user was assigned a matching tag for each attribute tag in the attribute expression.

   Review the resulting Cedar policy expression shown in the console.



5. Choose the permission scope. If the grantees belong to an external account, choose **External account** and enter the AWS account ID.
6. Next, choose the Data Catalog account or in external accounts. You must have corresponding grantable permissions on the resources to successfully complete the permission grants.
7. Specify which actions you want to allow for principals (users or roles) that have matching attributes perform. Access is granted to IAM entities that have been assigned tags and values

that match at least one of your specified attribute expressions. Review the Cedar policy expression in the console. For more information about Cedar, see What is Cedar? | Cedar Policy Language Reference GuideLink.

8. Next choose the Data Catalog resources to grant access. you can define these permissions for various Data Catalog resources, including catalogs, databases, tables, and data filters.
9. Choose **Grant**.

This approach allows you to control access based on attributes, ensuring that only users or roles with the appropriate tags can perform specific actions on the designated resources.

Granting permissions using ABAC (AWS CLI)

The following example shows an attribute expression that must be met for receiving all available permissions on the resource. You can alternatively specify individual permissions such as `Select`, `Describe`, or `Drop`. The expression uses Cedar policy expression. For more information about Cedar, see What is Cedar? | Cedar Policy Language Reference GuideLink.

This condition checks if the IAM principal has a `department` tag, and the `department` tag value equals `sales`.

```
aws lakeformation grant-permissions
--principal '{"DataLakePrincipalIdentifier": "111122223333:IAMPrincipals"}' \
--resource '{"Database": {"CatalogId": 111122223333, "Name": "abac-db"}}' \
--permissions ALL \
--condition '{"Expression": "context.iam.principalTags.hasTag(\"department\") \
   && context.iam.principalTags.getTag(\"department\") == \"sales\""}'
```

Permissions example scenario

The following scenario helps demonstrate how you can set up permissions to secure access to data in AWS Lake Formation.

Shirley is a data administrator. She wants to set up a data lake for her company, AnyCompany. Currently, all data is stored in Amazon S3. John is a marketing manager and needs write access to customer purchasing information (contained in `s3://customerPurchases`). A marketing analyst, Diego, joins John this summer. John needs the ability to grant Diego access to perform queries on the data without involving Shirley.

Mateo, from finance, needs access to query accounting data (for example, `s3://transactions`). He wants to query the transactions data in tables in a database (`Finance_DB`) that the finance team uses. His manager, Arnav, can give him access to the `Finance_DB`. Although he shouldn't be able to modify accounting data, he needs the ability to convert data into a format (schema) suitable for forecasting. This data will be stored in a separate bucket (`s3://financeForecasts`) that he can modify.

To summarize:

- Shirley is the data lake administrator.
- John requires `CREATE_DATABASE` and `CREATE_TABLE` permission to create new databases and tables in the Data Catalog.
- John also requires `SELECT`, `INSERT`, and `DELETE` permissions on tables he creates.
- Diego requires `SELECT` permission on the table to run queries.

The employees of AnyCompany perform the following actions to set up permissions. The API operations shown in this scenario show a simplified syntax for clarity.

1. Shirley registers the Amazon S3 path containing customer purchasing information with Lake Formation.

   ```
   RegisterResource(ResourcePath("s3://customerPurchases"), false, Role_ARN )
   ```

2. Shirley grants John access to the Amazon S3 path containing customer purchasing information.

   ```
   GrantPermissions(John, S3Location("s3://customerPurchases"),
    [DATA_LOCATION_ACCESS]) )
   ```

3. Shirley grants John permission to create databases.

   ```
   GrantPermissions(John, catalog, [CREATE_DATABASE])
   ```

4. John creates the database `John_DB`. John automatically has `CREATE_TABLE` permission on that database because he created it.

   ```
   CreateDatabase(John_DB)
   ```

5. John creates the table John_Table pointing to s3://customerPurchases. Because he created the table, he has all permissions on it, and can grant permissions on it.

```
CreateTable(John_DB, John_Table)
```

6. John allows his analyst, Diego, access to the table John_Table.

```
GrantPermissions(Diego, John_Table, [SELECT])
```

7. John allows his analyst, Diego, access to the s3://customerPurchases/London/. Because Shirley already registered s3://customerPurchases, its subfolders are registered with Lake Formation.

```
GrantDataLakePrivileges( 123456789012/datalake, Diego, [DATA_LOCATION_ACCESS], [],
S3Location("s3://customerPurchases/London/") )
```

8. John allows his analyst, Diego, to create tables in database John_DB.

```
GrantDataLakePrivileges( 123456789012/datalake, Diego, John_DB, [CREATE_TABLE],
[] )
```

9. Diego creates a table in John_DB at s3://customerPurchases/London/ and automatically gets ALTER, DROP, SELECT, INSERT, and DELETE permissions.

```
CreateTable( 123456789012/datalake, John_DB, Diego_Table )
```

Data filtering and cell-level security in Lake Formation

When you grant Lake Formation permissions on a Data Catalog table, you can include data filtering specifications to restrict access to certain data in query results and engines integrated with Lake Formation. Lake Formation uses data filtering to achieve column-level security, row-level security, and cell-level security. You can define and apply data filters on nested columns if your source data contains nested structures.

With the data filtering capabilities of Lake Formation, you can implement the following levels of data security.

Column-level security

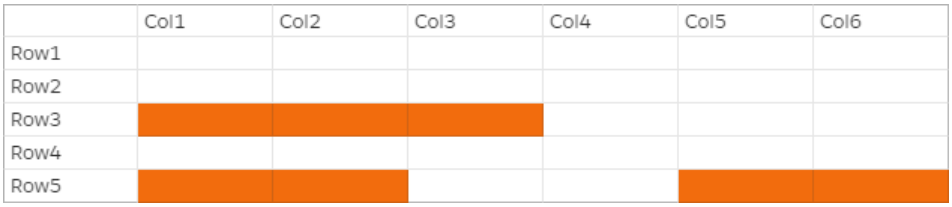
Granting permissions on a Data Catalog table with column-level security (column filtering) allows users to view only specific columns and nested columns that they have access to in the table. Consider a `persons` table that is used in multiple applications for a large multi-region communications company. Granting permissions on Data Catalog tables with column filtering can restrict users who don't work in the HR department from seeing personally identifiable information (PII) such as a social security number or birth date. You can also define security policies and grant access to only partial sub-structures of nested columns.

Row-level security

Granting permissions on a Data Catalog table with row-level security (row filtering) allows users to view only specific rows of data that they have access to in the table. The filtering is based on the values of one or more columns. You can include nested column structures when defining row-filter expressions. For example, if different regional offices of the communications company have their own HR departments, you can limit the person records that HR employees can see to only records for employees in their region.

Cell-level security

Cell-level security combines row filtering and column filtering for a highly flexible permissions model. If you view the rows and columns of a table as a grid, by using cell-level security, you can restrict access to individual elements (cells) of the grid anywhere in the two dimensions. That is, you can restrict access to different columns depending on the row. This is illustrated by the following diagram, in which restricted columns are shaded.



Continuing the example of the persons table, you can create a *data filter* at the cell-level that restricts access to the street address column if the row has the country column set to "UK", but allows access to the street address column if the row has the country column set to "US".

Filters apply only to read operations. Therefore, you can grant only the SELECT Lake Formation permission with filters.

Cell-level security on nested columns

Lake Formation allows you to define and apply data filters with cell-level security on nested columns. However, the integrated analytical engines such as Amazon Athena, Amazon EMR, and

Amazon Redshift Spectrum support executing queries against Lake Formation managed nested tables with row and column-level security.

For limitations, see Data filtering limitations.

Topics

- Data filters in Lake Formation
- PartiQL support in row filter expressions
- Permissions required for querying tables with cell-level filtering
- Managing data filters

Data filters in Lake Formation

You can implement column-level, row-level, and cell-level security by creating *data filters*. You select a data filter when you grant the `SELECT` Lake Formation permission on tables. If your table contains nested column structures, you can define a data filter by including or excluding the child columns and define row-level filter expressions on nested attributes.

Each data filter belongs to a specific table in your Data Catalog. A data filter includes the following information:

- Filter name
- The Catalog IDs of the table associated with the filter
- Table name
- Name of the database that contains the table
- Column specification – a list of columns and nested columns (with `struct` datatypes) to include or exclude in query results.
- Row filter expression – an expression that specifies the rows to include in query results. With some restrictions, the expression has the syntax of a `WHERE` clause in the PartiQL language. To specify all rows, choose **Access to all rows** under **Row-level access** in the console or use `AllRowsWildcard` in API calls.

  For more information about what is supported in row filter expressions, see PartiQL support in row filter expressions.

The level of filtering that you get depends on how you populate the data filter.

- When you specify the "all columns" wildcard and provide a row filter expression, you are establishing row-level security (row filtering) only.
- When you include or exclude specific columns and nested columns, and specify "all rows" using the all-rows wildcard, you are establishing column-level security (column filtering) only.
- When you include or exclude specific columns and also provide a row filter expression, you are establishing cell-level security (cell filtering).

The following screenshot from the Lake Formation console shows a data filter that performs cell-level filtering. For queries against the `orders` table, it restricts access to the `customer_name` column and the query results return only rows where the `product_type` column contains 'pharma'.

Create data filter

Data filter name
Enter a name that describes this data access filter.

restrict-pharma

Name may contain letters (A-Z), numbers (0-9), hyphens (-), or under-scores (_), and be less than 256 characters.

Target database
Select the database that contains the target table.

Choose databases Load more

sales
054881201579

Target table
Select the table for which the data filter will be created.

Choose tables Load more

orders
054881201579

Column-level access
Choose whether this filter should have column-level restrictions.

- Access to all columns
  Filter won't have any column restrictions.
- Include columns
  Filter will only allow access to specific columns.
- Exclude columns
  Filter will allow access to all but specific columns.

Select columns

Choose one or more columns

customer_name
string

Row filter expression
Enter the rest of the following query statement "SELECT * FROM orders WHERE..."
Please see the documentation for examples of filter expressions.

product_type='pharma'

Cancel Create filter

Note the use of single quotes to enclose the string literal, `'pharma'`.

You can use the Lake Formation console to create this data filter, or you can supply the following request object to the `CreateDataCellsFilter` API operation.

```
{
     "Name": "restrict-pharma",
     "DatabaseName": "sales",
     "TableName": "orders",
     "TableCatalogId": "111122223333",
     "RowFilter": {"FilterExpression": "product_type='pharma'"},
     "ColumnWildcard": {
         "ExcludedColumnNames": ["customer_name"]
     }
}
```

You can create as many data filters as you need for a table. In order to do so, you require SELECT permission with the grant option on a table. Data Lake Administrators by default have the permission to create *data filters* on all tables in that account. You typically only use a subset of the possible data filters when granting permissions on the table to a principal. For example, you could create a second data filter for the `orders` table that is a row-security-only data filter. Referring to the preceding screenshot, you could choose the **Access to all columns** option and include a row filter expression of `product_type<>pharma`. The name of this data filter could be `no-pharma`. It restricts access to all rows that have the `product_type` column set to 'pharma'.

The request object for the `CreateDataCellsFilter` API operation for this data filter is the following.

```
{
     "Name": "no-pharma",
     "DatabaseName": "sales",
     "TableName": "orders",
     "TableCatalogId": "111122223333",
     "RowFilter": {"FilterExpression": "product_type<>'pharma'"},
     "ColumnNames": ["customer_id", "customer_name", "order_num"
          "product_id", "purchase_date", "product_type",
          "product_manufacturer", "quantity", "price"]
}
```

You could then grant SELECT on the `orders` table with the `restrict-pharma` data filter to an administrative user, and SELECT on the `orders` table with the `no-pharma` data filter to non-

administrative users. For users in the healthcare sector, you would grant SELECT on the `orders` table with full access to all rows and columns (no data filter), or perhaps with yet another data filter that restricts access to pricing information.

You can include or exclude nested columns when specifying column-level and row-level security within a data filter. In the following example, access to the `product.offer` field is specified using qualified column names (wrapped in double quotes). This is important for nested fields in order to avoid errors occurring when column names contain special characters, and to maintain backward compatibility with top level column-level security definitions.

```
{
     "Name": "example_dcf",
     "DatabaseName": "example_db",
     "TableName": "example_table",
     "TableCatalogId": "111122223333",
     "RowFilter": { "FilterExpression": "customer.customerName <> 'John'" },
     "ColumnNames": ["customer", "\"product\".\"offer\""]
}
```

See also

- Managing data filters

PartiQL support in row filter expressions

You can construct row filter expressions using a subset of PartiQL data types, operators, and aggregations. Lake Formation does not allow any user defined or standard partiQL functions in the filter expression. You can use comparison operators to compare columns with constants (for example, `views >= 10000`), but you can't compare columns with other columns.

A Row filter expression may be a simple expression or a composite expression. Total length of the expression must be less than 2048 characters.

Simple expression

A simple expression will be of the format: `<column name > <comparison operator ><value >`

- **Column name**

  It can either a top level data column, a partition column, or a nested column present in the table schema and must belong to the Supported data types listed below.

- **Comparison operator**

  The following are the supported operators: `=, >, <, >=, <=, <>,!=, BETWEEN, IN, LIKE, NOT, IS [NOT] NULL`

- All string comparisons and `LIKE` pattern matches are case-sensitive. You can't use IS [NOT] NULL operator on partition columns.

- **Column value**

  The Column value must match the data type of the column name.

Composite expression

A composite expression will be of the format: `( <simple expression >) <AND/OR >(<simple expression >)`. Composite expressions can be further combined using logical operators `AND/OR`.

Supported data types

Row filters that refer to an AWS Glue Data Catalog table that contains an unsupported data types will result in an error. The following are the supported data types for table columns and constants, which are mapped to Amazon Redshift data types:

- `STRING, CHAR, VARCHAR`
- `INT, LONG, BIGINT, FLOAT, DECIMAL, DOUBLE`
- `BOOLEAN`
- `STRUCT`

For more information about data types in Amazon Redshift, see Data types in *Amazon Redshift Database Developer Guide*.

Row filter expressions

Example

The following are examples of valid row filter expressions for a table with columns: `country (String), id (Long), year (partition column of type Integer), month (partition column of type Integer)`

- `year > 2010 and country != 'US'`
- `(year > 2010 and country = 'US') or (month < 8 and id > 23)`
- `(country between 'Z' and 'U') and (year = 2018)`
- `(country like '%ited%') and (year > 2000)`

Example

The following is a valid examples of row filter expressions for a table with nested columns: `year > 2010 and customer.customerId <> 1`

Nested fields under partition columns should not be referenced when defining nested row-level expressions.

String constants must be enclosed in single-quotes.

Reserved keywords

If your row filter expression contains PartiQL keywords, you will receive a parsing error as column names may conflict with the keywords. When this happens, escape the column names by using double quotes. Some examples of reserved keywords are “first”, “last”, “asc”, “missing”. See PartiQL specification for a list of reserved keywords.

PartiQL reference

For more information about PartiQL, see https://partiql.org/.

Permissions required for querying tables with cell-level filtering

The following AWS Identity and Access Management (IAM) permissions are required to run queries against tables with cell-level filtering.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:StartQueryPlanning",
                "lakeformation:GetQueryState",
                "lakeformation:GetWorkUnits",
                "lakeformation:GetWorkUnitResults"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about Lake Formation permissions, see Lake Formation personas and IAM permissions reference.

Managing data filters

To implement column-level, row-level, and cell-level security, you can create and maintain data filters. Each data filter belongs to a Data Catalog table. You can create multiple data filters for a table, and then use one or more of them when granting permissions on the table. You can also define and apply data filters on nested columns that have `struct` datatypes allowing users to access only sub-structures of nested columns.

You require SELECT permission with the grant option to create or view a data filter. To allow principals in your account to view and use a data filter, you can grant the DESCRIBE permission on it.

> **Note**
>
> Lake Formation doesn't support granting `Describe` permission on a data filter, which is shared from another account.

You can manage data filters by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

For information about data filters, see Data filters in Lake Formation

Creating a data filter

You can create one or more data filters for each Data Catalog table.

To create a data filter for a Data Catalog table (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign as a data lake administrator, the target table owner, or a principal who has a Lake Formation permission on the target table.
2. In the navigation pane, under **Data catalog**, choose **Data filters**.
3. On the **Data filters** page, choose **Create new filter**.
4. In the **Create data filter** dialog box, enter the following information:

   - Data filter name
   - Target database – Specify the database that contains the table.
   - Target table
   - Column-level access – Leave this set to **Access to all columns** to specify row filtering only. Choose **Include columns** or **Exclude columns** to specify column or cell filtering, and then specify the columns to include or exclude.

     Nested columns – If you're applying the filter on a table that contains nested columns, you can explicitly specify sub-structures of the nested struct columns within a data filter.

     When you grant SELECT permission to a principal on this filer, the principal executing the following query, will only see the data for `customer.customerName` and not `customer.customerId`.

     ```
     SELECT "customer" FROM "example_db"."example_table";
     ```

Column-level access

Choose whether this filter should have column-level restrictions.

Column-level access

Choose whether this filter should have column-level restrictions.

- ○ Access to all columns
  Filter won't have any column restrictions.
- ◉ Include columns
  Filter will only allow access to specific columns.
- ○ Exclude columns
  Filter will allow access to all but specific columns.

Included columns (4/11)

Choose the columns for column-level access

Find column

1

	Name	Type
☐	customer	struct
☐	customerId	string
☑	customerName	string
☑	customerapplication	struct
☐	appId	string
☑	product	struct
☐	offer	struct
☐	listingId	string
☐	prodId	string
☐	type	string
☑	purchaseid	string

Row-level access

Choose whether this filter should have row-level restrictions.

- ○ Access to all rows
- ◉ Filter rows

Row filter expression

Enter the rest of the following query statement SELECT * FROM nested-table WHERE...
Please see the documentation for examples of filter expressions.

```
customer.customerName <> 'John'
```

When you grant permissions to the `customer` column, the principal receives the access to the column and the nested fields under the column (`customerName` and `customerID`).

- Row filter expression – Enter a filter expression to specify row or cell filtering. For supported data types and operators, see PartiQL support in row filter expressions. Choose **Access to all rows** to grant access to all .

  You can include partial column structs from nested columns in a row filter expression to filter rows that contain specific value.

  When a principal is granted permissions to a table with a row filter expression `Select * from example_nestedtable where customer.customerName <>'John'`, and **Column-level** access is set to **Access to all columns**, the query results shows only rows where `customerName <>'John'` evaluates to true.

The following screenshot shows a data filter that implements cell filtering. In queries against the `orders` table, it denies access to the `customer_name` column and shows only rows that have 'pharma' in the `product_type` column.

Create data filter

Data filter name
Enter a name that describes this data access filter.

restrict-pharma

Name may contain letters (A-Z), numbers (0-9), hyphens (-), or under-scores (_), and be less than 256 characters.

Target database
Select the database that contains the target table.

Choose databases | Load more

sales
054881201579

Target table
Select the table for which the data filter will be created.

Choose tables | Load more

orders
054881201579

Column-level access
Choose whether this filter should have column-level restrictions.

- Access to all columns
  Filter won't have any column restrictions.
- Include columns
  Filter will only allow access to specific columns.
- Exclude columns
  Filter will allow access to all but specific columns.

Select columns

Choose one or more columns

customer_name
string

5. Choose **Create filter**.

To create a data filter with cell-filter policies on a nested field

This section uses the following sample schema to show how to create a data cells filter:

```
[
    { name: "customer", type: "struct<customerId:string,customerName:string>" },
    { name: "customerApplication", type: "struct<appId:string>" },
    { name: "product", type:
 "struct<offer:struct<prodId:string,listingId:string>,type:string>" },
    { name: "purchaseId", type: "string" },
]
```

1. On the **Create a data filter**, page enter a name for the data filter.
2. Next, use the drop-down to choose a database name and table name.
3. In the **Column-level access** section, choose Included columns, and select a nested column (`customer.customerName`).
4. In the **Row-level access** section, choose the **Access to all rows** option.
5. Choose **Create filter**.

   When you grant SELECT permission on this filter, the principal gets access to all rows in the `customerName` column.
6. Next, define another data filter for the same database/table.
7. In the **Column-level access** section, choose Included columns, and select another nested column (`customer.customerid`).
8. In the **Row-level access** section, choose **Filter rows**, and enter a **Row filter expression** (`customer.customerid <> 5`).
9. Choose **Create filter**.

   When you grant SELECT permission on this filter, the principal receives access to all rows in the `customerName`, and `customerId` fields except the cell where the value is 5 in the `customerId` column.

Granting data filter permissions

You can grant the SELECT, DESCRIBE and DROP Lake Formation permissions on data filters to principals.

At first, only you can view the data filters that you create for a table. To enable another principal to view a data filter and grant Data Catalog permissions with the data filter, you must either:

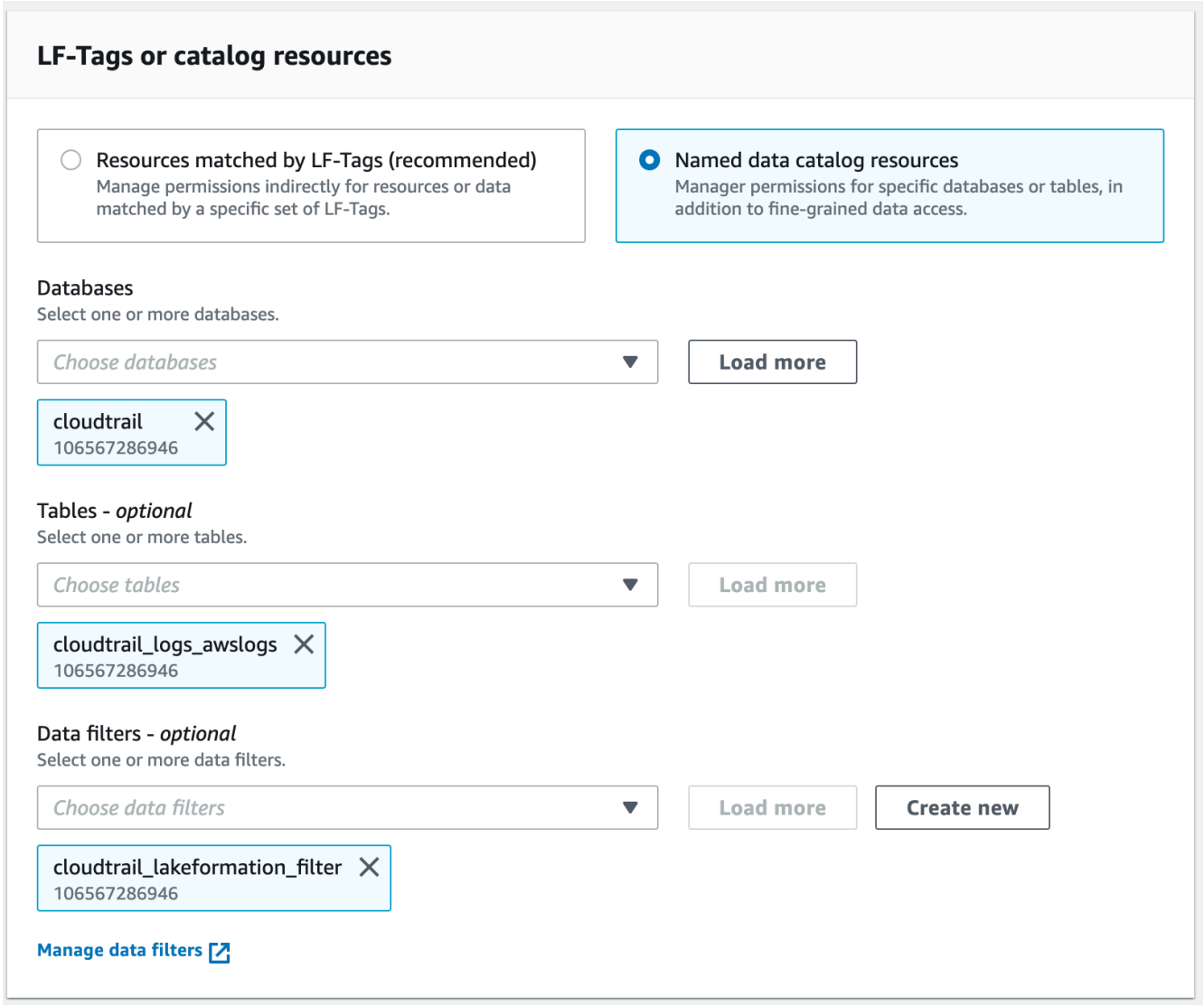
- Grant SELECT on a table to the principal with the grant option, and apply the data filter to the grant.
- Grant the DESCRIBE or DROP permission on the data filter to the principal.

You can grant the SELECT permission to an external AWS account. A data lake administrator in that account can then grant that permission to other principals in the account. When granting to an external account, you must include the grant option so that administrator of the external account can further cascade the permission to other users in his/her account. When granting to a principal in your account, granting with the grant option is optional.

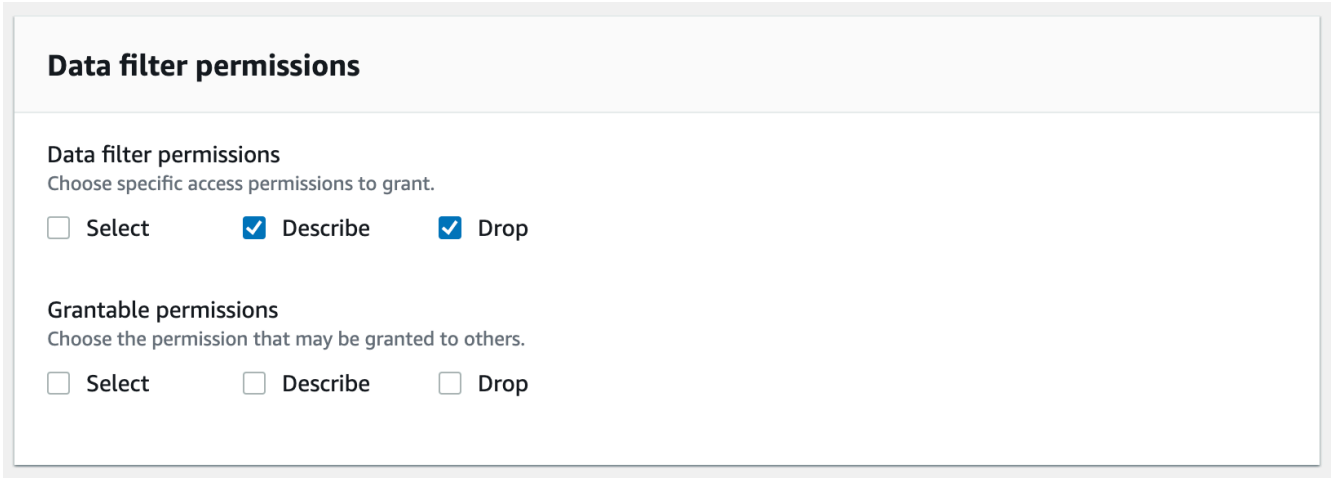
You can grant and revoke permissions on data filters by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Console

1. Sign in to the AWS Management Console and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, under **Permissions**, choose **Data lake permissions**.
3. On the **Permissions** page, in the **Data permissions** section, choose **Grant**.
4. On the **Grant data permissions** page, choose the principals to grant the permissions to.
5. In the LF-Tags or catalog resources section, choose **Named data catalog resources**. Then choose the database, table, and data filter for which you want to grant permissions.



6. In the **Data filter permissions** section, choose the permissions you want to grant to the selected principals.



AWS CLI

- Enter a `grant-permissions` command. Specify `DataCellsFilter` for the `resource` argument, and specify `DESCRIBE` or `DROP` for the `Permissions` argument and, optionally, for the `PermissionsWithGrantOption` argument.

  The following example grants `DESCRIBE` with the grant option to user `datalake_user1` on the data filter `restrict-pharma`, which belongs to the `orders` table in the `sales` database in AWS account 1111-2222-3333.

  ```
  aws lakeformation grant-permissions --cli-input-json file://grant-params.json
  ```

  The following are the contents of file `grant-params.json`.

  ```
  {
      "Principal": {"DataLakePrincipalIdentifier":
   "arn:aws:iam::111122223333:user/datalake_user1"},
      "Resource": {
          "DataCellsFilter": {
              "TableCatalogId": "111122223333",
              "DatabaseName": "sales",
              "TableName": "orders",
              "Name": "restrict-pharma"
          }
      },
      "Permissions": ["DESCRIBE"],
      "PermissionsWithGrantOption": ["DESCRIBE"]
  }
  ```

Granting data permissions provided by data filters

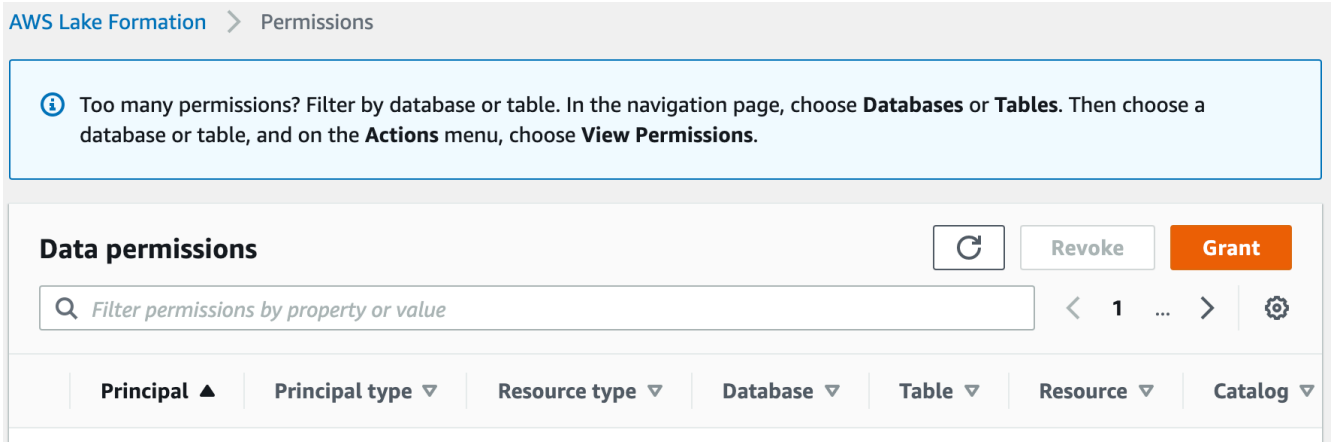
Data filters represent a subset of data within a table. To provide data access to principals, `SELECT` permissions need to be granted to those principals. With this permission the principals can:

- View the actual table name in list of tables shared with their account.
- Create data filters on the shared table and grant permissions to their users on those data filters.

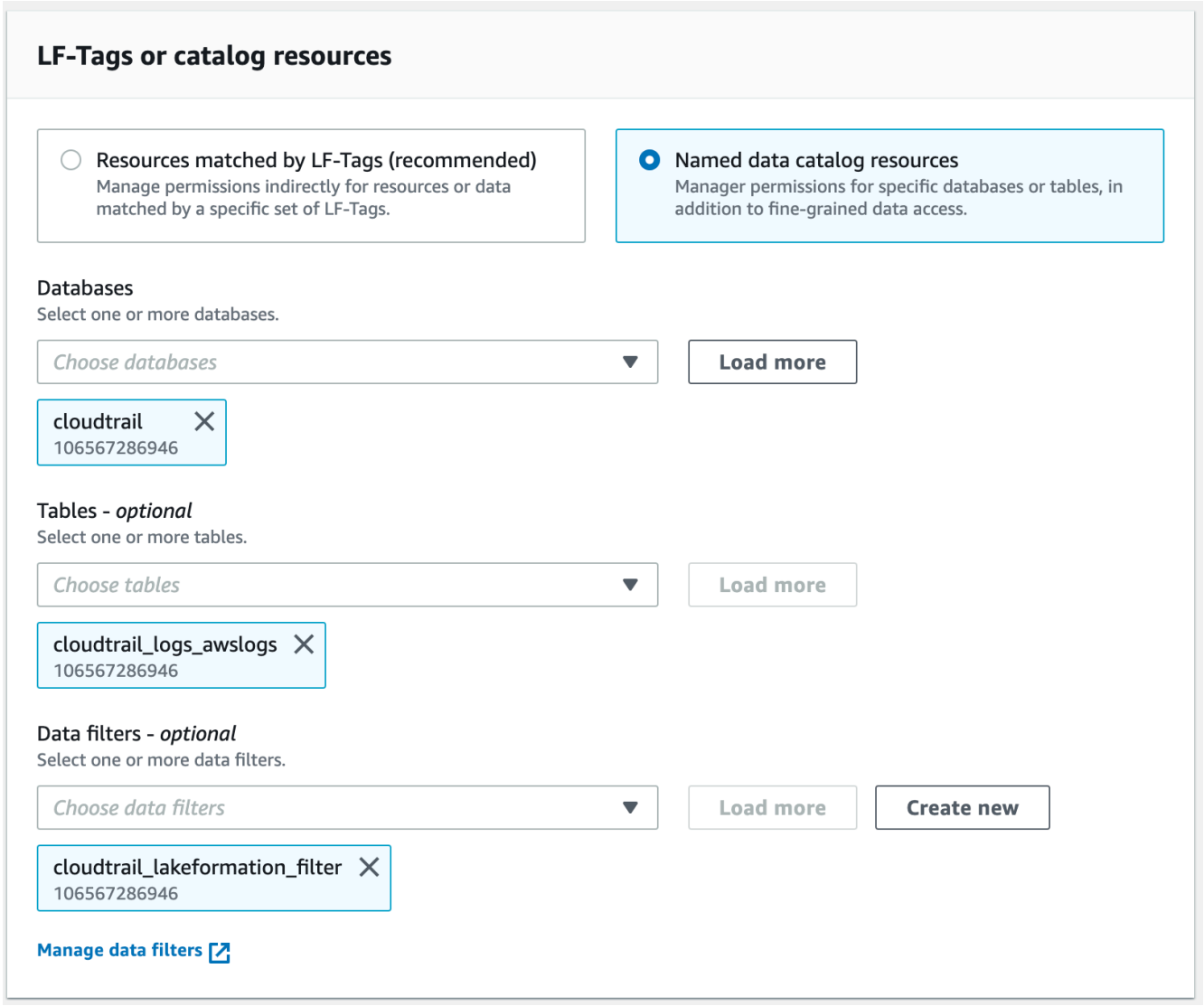
Console

To grant SELECT permissions

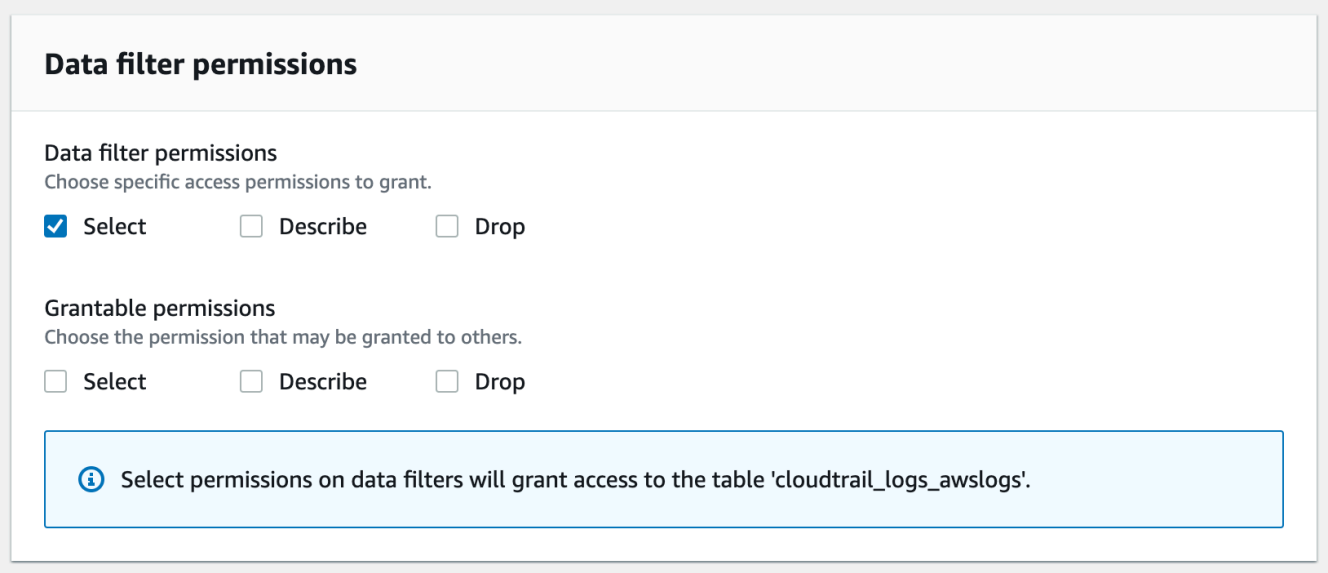
1. Go to the **Permissions** page in the Lake Formation console, and then choose **Grant**.



2. Select the principals you want to provide access to, and select **Named data catalog resources**.



3. To provide access to the data that the filter represents, choose **Select** under **Data filter permissions**.



CLI

Enter a `grant-permissions` command. Specify `DataCellsFilter` for the resource argument, and specify `SELECT` for the Permissions argument.

The following example grants `SELECT` with the grant option to user `datalake_user1` on the data filter `restrict-pharma`, which belongs to the `orders` table in the `sales` database in AWS account `1111-2222-3333`.

```
aws lakeformation grant-permissions --cli-input-json file://grant-params.json
```

The following are the contents of file `grant-params.json`.

```
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/
datalake_user1"
    },
    "Resource": {
        "DataCellsFilter": {
            "TableCatalogId": "111122223333",
            "DatabaseName": "sales",
            "TableName": "orders",
            "Name": "restrict-pharma"
        }
    },
```

```
    "Permissions": ["SELECT"]
}
```

Viewing data filters

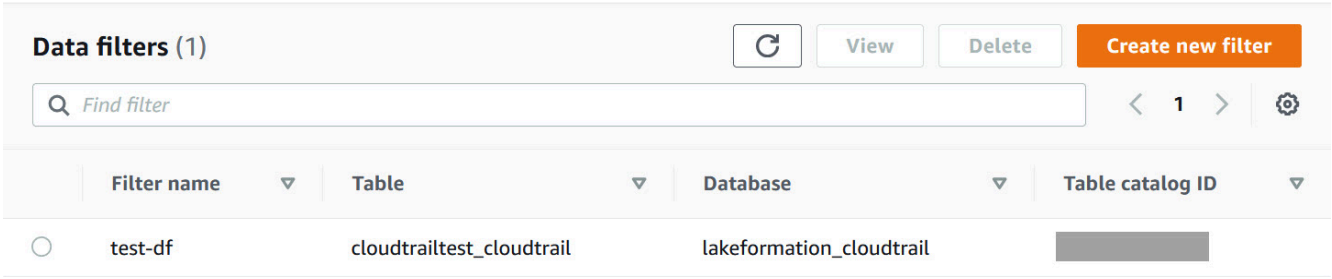
You can use the Lake Formation console, AWS CLI, or the Lake Formation API to view data filters.

To view data filters, you must be a Data Lake administrator or have the required permissions on the data filters.

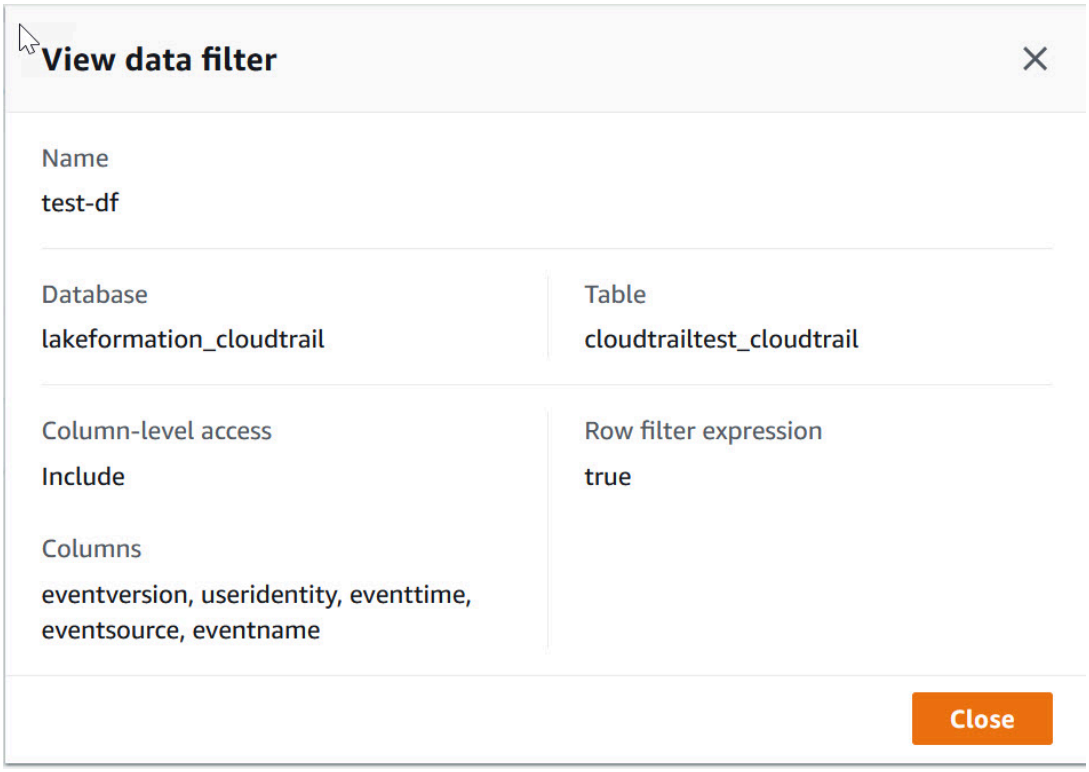
Console

1. Sign in to the AWS Management Console and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, under **Data catalog**, choose **Data filters**.

   The page displays the data filters you have access to.



3. To view the data filter details, choose the data filter, and then choose View. A new window appears with the data filter detailed information.



AWS CLI

Enter a `list-data-cells-filter` command and specify a table resource.

The following example lists the data filters for the `cloudtrailtest_cloudtrail` table.

```
aws lakeformation list-data-cells-filter --table '{ "CatalogId":"123456789012",
"DatabaseName":"lakeformation_cloudtrail", "Name":"cloudtrailtest_cloudtrail"}'
```

API/SDK

Use the `ListDataCellsFilter` API and specify a table resource.

The following example uses Python to list the first 20 data filters for the `myTable` table.

```
response = client.list_data_cells_filter(
    Table = {
        'CatalogId': '111122223333',
        'DatabaseName': 'mydb',
        'Name': 'myTable'
    },
```

```
    MaxResults=20
)
```

Listing data filter permissions

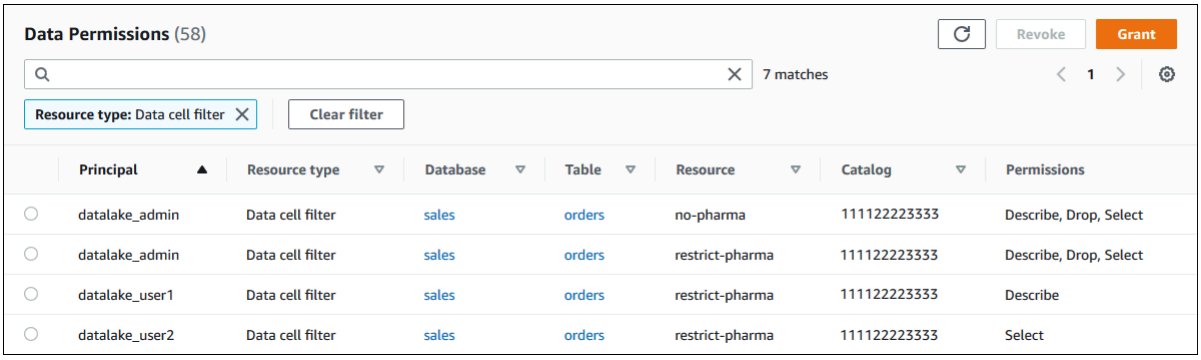
You can use the Lake Formation console to view the permissions granted on data filters.

To view permissions on a data filter, you must be a Data Lake administrator or have the required permissions on the data filter.

Console

1. Sign in to the AWS Management Console and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the navigation pane, under **Permissions**, choose **Data permissions**.
3. On the **Data Permissions** page, click or tap in the search field, and on the **Properties** menu, choose **Resource type**.
4. On the **Resource type** menu, choose **Resource type: Data cell filter**.

   The data filters that you have permissions on are listed. You might have to scroll horizontally to see the **Permissions** and **Grantable** columns.



AWS CLI

- Enter a `list-permissions` command. Specify `DataCellsFilter` for the `resource` argument, and specify `DESCRIBE` or `DROP` for the `Permissions` argument and, optionally, for the `PermissionsWithGrantOption` argument.

  The following example lists `DESCRIBE` permissions with the grant option on the data filter `restrict-pharma`. The results are limited to permissions granted for the principal

datalake_user1 and the orders table in the sales database in AWS account 1111-2222-3333.

```
aws lakeformation list-permissions --cli-input-json file://list-params.json
```

The following are the contents of file grant-params.json.

```
{
    "Principal": {"DataLakePrincipalIdentifier":
 "arn:aws:iam::111122223333:user/datalake_user1"},
    "Resource": {
        "DataCellsFilter": {
            "TableCatalogId": "111122223333",
            "DatabaseName": "sales",
            "TableName": "orders",
            "Name": "restrict-pharma"
        }
    },
    "Permissions": ["DESCRIBE"],
    "PermissionsWithGrantOption": ["DESCRIBE"]
}
```

Row-level security policies in AWS Lake Formation

Row-level security (RLS) policies in AWS Lake Formation let you grant access to specific rows in a table based on filter expressions. When a principal queries a table with an attached RLS policy, Lake Formation automatically applies the filter so that only matching rows are returned.

How RLS policies work

An RLS policy consists of a named filter expression and a set of columns the predicate applies to. You create a policy once, then attach it to one or more tables for specific principals. When a principal queries a table with an RLS policy attachment, Lake Formation evaluates the policy and returns only the rows the principal is allowed to see.

RLS policies are **additive**. If multiple policies are attached to a table for a principal, the principal sees the union of all matching rows. This means attaching additional policies only expands access, never restricts it.

With RLS policies in Lake Formation, you can implement fine-grained row-level data security that scales across thousands of tables.

Key concepts

Policy components

Each RLS policy has the following components:

- **Name** – A unique identifier for the policy within your catalog. Names are case-insensitive.
- **FilterExpression** – A PartiQL-subset expression that defines which rows are visible. For example: `department = 'Engineering'`. For the supported syntax, operators, and data types, see Filter expression syntax for RLS policies.
- **AttachmentColumns** – The columns referenced in the filter expression, each with a name and data type.

Attachment methods

You can attach RLS policies to tables in two ways:

- **Direct attachment** – Attach the policy to a specific table for a specific principal.
- **Tag-based attachment** – Attach the policy to an LF-Tag. Any table matching the tag expression inherits the policy.

Prerequisite: SELECT permission

RLS policies filter rows, but they do not grant `SELECT` permission on the table. Principals must already have `SELECT` permission (granted through Lake Formation permissions, Lake Formation tag-based access control (LF-TBAC), attribute-based access control (ABAC), or data cell filters), which can be granted by a data lake administrator.

Default policy

A default RLS policy applies to every principal with `SELECT` access on the table. If a principal also has an explicit policy attached, they see the union of the default policy's rows and their explicit policy's rows. Set a default policy by specifying `SetDefault: true` when attaching the policy. As a best practice, set a default policy before you grant any permissions.

RLS policies compared to data cell filters

Data cell filters (DCF) also provide row-level filtering. The following table compares the two approaches.

Feature	Data cell filters	RLS policies
Reusability	Bound to a single table	Attach one policy to many tables
Column filtering	Supported (include/exclude columns)	Not supported (use column-level permissions instead)
Tag-based attachment	Not supported	Supported via LF-Tag policy resources
Default policy	Not supported	Supported (applies to all principals without explicit attachment)
ABAC conditions	Supported	Supported
Update propagation	Must update each filter individually per table	Update once, change propagates to all attached tables

Important

DCF provides column filtering and per-table row filtering with a simpler setup. RLS policies provide scalable, reusable row filtering across many tables with stronger enforcement against SELECT statements. Evaluate your requirements before choosing one or both. For a detailed comparison and migration guidance, see Best practices for RLS policies.

Integration with other Lake Formation features

Column-level security

RLS policies work alongside column-level permissions. If a principal has access to only certain columns and also has an RLS policy attached, both restrictions apply. The principal sees only the permitted columns for the rows matching the filter.

Lake Formation tag-based access control (LF-TBAC)

You can use LF-Tags to attach RLS policies to multiple tables. When you attach an RLS policy using LF-Tags, the policy automatically applies to any table that matches the tag.

Attribute-based access control (ABAC)

You can attach an RLS policy using attributes that are IAM tags and session tags associated with IAM roles and principals. Lake Formation evaluates IAM session attributes at query time and determines whether the attachment applies to the caller. This lets you create attribute-driven row filtering without creating separate attachments for each principal.

Data cell filters

RLS policies and data cell filters (DCF) can apply to the same table at the same time. When a principal is subject to both a DCF row filter and an RLS policy on a table, Lake Formation combines the two row filters using union (OR) semantics. The principal sees every row that matches either the DCF row filter or the RLS policy filter.

Because the row filters are additive, adding an RLS policy to a table that already has a DCF row filter can only expand the rows a principal sees, never restrict them. Plan for this behavior when you use both features together.

RLS compatibility

Table types

RLS policies are supported on any table registered in the AWS Glue Data Catalog, including:

- Apache Iceberg tables
- Apache Hive tables
- Delta Lake tables
- Apache Hudi tables
- Amazon S3 Tables

- Tables registered through federated catalogs (Amazon Redshift, and third-party catalogs where supported)

Query engines

RLS policies are enforced by any analytics engine integrated with Lake Formation fine-grained access control, including:

- Amazon Athena
- Amazon Redshift Spectrum
- Amazon EMR (Spark in FGAC mode)
- AWS Glue ETL (Spark in FGAC mode)
- Third-party engines that support Lake Formation fine-grained access control via session tags

Identity types

RLS policies can be attached to any principal type supported by Lake Formation.

Unsupported features

You cannot attach RLS policies to multi-dialect views. For more information, see Row-level security policy limitations.

Getting started

To use RLS policies, create an RLS policy with a filter expression and attachment columns, attach the policy to the table for one or more principals (or set it as a default), and grant SELECT on the table to the principals who need access. Principals then query the table and automatically see only the filtered rows. For detailed steps, see Managing RLS policies.

How RLS policies work in detail

This section covers how row-level security (RLS) policies are structured, enforced, and used in practice, including a worked end-to-end example.

Policy components

An RLS policy consists of the following fields.

Field	Type	Description
Name	String	A unique, case-insensitive identifier for the policy within a catalog. Maximum 255 characters.
FilterExpression	String	A PartiQL-subset expressio n that defines the row filter condition. Maximum 2048 characters.
AttachmentColumns	List	The columns referenced in the filter expression. Each entry contains a `Name` (string) and a `Type` (string).

AttachmentColumns structure

Each entry in the AttachmentColumns list identifies a column by name and data type.

```
{
  "AttachmentColumns": [
    {
      "Name": "department",
      "Type": "string"
    },
    {
      "Name": "region_code",
      "Type": "int"
    }
  ]
}
```

Column names are case-insensitive, but type strings are case-sensitive and must be lowercase (`string`, `int`, `bigint`, `double`, `decimal`, `boolean`). Uppercase values such as `STRING` are rejected.

Lake Formation does not validate the type against the target table when you create or update a policy, because no table is involved at that point. For name-based table attachments, it is validated when you attach the policy. The attachment columns **must** match the table's columns by name (case-insensitive) and by equivalent type. If they do not match, the attach fails. For all-tables and tag-based attachments, the validation happens at query time.

Every column referenced in the filter expression must also appear in AttachmentColumns. Referencing a column that is not declared causes the create to fail.

Enforcement

Additive (union) semantics

When multiple RLS policies are attached to the same table for the same principal, Lake Formation combines them using union logic. The principal sees all rows that satisfy any of the attached policies.

Example Additive policies

A principal has two policies attached:

- Policy A: `department = 'Engineering'`
- Policy B: `department = 'Sales'`

The principal sees all rows where department is either Engineering or Sales.

SELECT-based enforcement

An RLS policy scopes down a principal's `SELECT` permission only. It scopes down the rows a principal can read. It does not add or remove any other permission, and it does not affect principals whose access comes from a broader grant.

How RLS interacts with a principal's existing permissions on the table:

- **SELECT permission, no RLS policy attached** – The principal sees all rows, limited to the columns their Lake Formation `SELECT` permission grants.
- **SELECT permission, RLS policy attached** – The principal sees all columns they are granted, with rows restricted to those matching the filter.

- **ALL or SUPER_USER permission** – The principal sees the full table. RLS scopes down SELECT, so a broader ALL or SUPER_USER grant is not affected by an RLS attachment.
- **No permission on the table** – The principal has no access. An RLS attachment alone does not grant SELECT.

To let a principal see all rows on a table that has RLS in effect, either grant ALL on the table or attach an RLS policy whose filter matches every row.

Default policy (Highly recommended)

A default RLS policy applies to every principal accessing the table. It serves as a baseline filter. If a principal also has an explicit policy attachment, the default policy is combined with the explicit policy using additive (union) semantics.

Example Default policy combined with an explicit policy

A table has:

- Default policy: `status = 'active'`
- Explicit policy for user A: `department = 'Engineering'`

User A sees rows where `status = 'active'` OR `department = 'Engineering'`. A user with no explicit attachment sees only rows where `status = 'active'`.

Policy lifecycle

The typical lifecycle of an RLS policy follows these stages.

1. Create

Define the policy with a name, filter expression, and attachment columns using the `CreateRLSPolicy` API or the Lake Formation console.

2. Attach

Attach the policy to one or more tables for specific principals using `AttachRLSPolicy`. You can attach using name-based attachment or tag-based attachment.

3. Enforce

Once attached, the policy is immediately enforced. Any query from the specified principal against the table has the filter applied transparently.

4. Update

Modify the filter expression using `UpdateRLSPolicy`. Changes take effect immediately on all existing attachments. Note that attachment columns cannot be changed after creation. To change columns, delete and recreate the policy and perform the attachment again.

5. Delete

Remove the policy using `DeleteRLSPolicy`. Deletion immediately removes the policy and all its attachments. If a principal has other policies (or a default policy) still attached to the table, they see the union of the remaining policies' rows. If no policies and no default remain, RLS stops filtering and the principal sees all rows their `SELECT` permission allows.

Worked example

The following example demonstrates the complete lifecycle of an RLS policy that restricts access to rows in a `transactions` table based on region.

Step 1: Create the policy

```
{
  "Name": "us-east-transactions-only",
  "FilterExpression": "region = 'us-east-1'",
  "AttachmentColumns": [
    {
      "Name": "region",
      "Type": "string"
    }
  ]
}
```

Step 2: Attach to a table for a principal

```
{
  "Name": "us-east-transactions-only",
```

```
  "Resource": {
    "Table": {
      "DatabaseName": "finance_db",
      "Name": "transactions"
    }
  },
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/AnalystRole"
  }
}
```

Step 3: Grant SELECT for the principal

```
{
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/AnalystRole"
  },
  "Resource": {
    "Table": {
      "DatabaseName": "finance_db",
      "Name": "transactions"
    }
  },
  "Permissions": [
    "SELECT"
  ]
}
```

Step 4: Query results

When the `AnalystRole` queries the `transactions` table:

```
SELECT * FROM finance_db.transactions;
```

Lake Formation appends the filter condition automatically. The role only sees rows where `region = 'us-east-1'`.

Step 5: Update the filter

To expand access to include a second region:

```
{
  "Name": "us-east-transactions-only",
  "FilterExpression": "region IN ('us-east-1', 'us-west-2')"
}
```

The change takes effect immediately on all existing attachments.

Step 6: Delete the policy

```
{
  "Name": "us-east-transactions-only"
}
```

After deletion, no RLS policy filters the `AnalystRole` on this table, so it sees all rows its `SELECT` permission allows (not only `us-east-1`).

Managing RLS policies

This section covers how to create, update, delete, list, and manage row-level security (RLS) policies and their attachments.

Permissions required

The following list shows the permissions required for each RLS policy operation.

Create

Data lake administrators have implicit access. Non-admin principals need `CREATE_RLS_POLICY` permission on the catalog.

Update

Data lake administrators have implicit access. Non-admin principals need `ALTER` permission on the RLS policy resource.

Delete

Data lake administrators have implicit access. Non-admin principals need `DROP` permission on the RLS policy resource.

Attach, detach

Data lake administrators have implicit access. Non-admin principals need `ASSOCIATE` permission on the RLS policy resource, plus `SELECT` with the grant option (or `ALL`) on the target table. For tag-based attachments, `MANAGE_TAG_POLICY` on the referenced LF-Tag keys and values is required instead of table permissions.

List policies, list attachments

Any principal with `DESCRIBE` permission on the RLS policy resource. Data lake administrators can list all policies.

Data lake administrators have implicit permission to perform all RLS policy operations without explicit grants.

Creating an RLS policy

AWS Management Console

To create an RLS policy (console)

1. Open the Lake Formation console.
2. In the navigation pane, choose **RLS policies**.
3. Choose **Create policy**.
4. For **Policy name**, enter a unique name. Names are case-insensitive.
5. For **Filter expression**, enter the PartiQL-subset expression (for example, `status = 'active'`).
6. Under **Attachment columns**, add each column referenced in the filter expression. Specify the column **Name** and **Type**.
7. Choose **Create**.

AWS CLI

```
aws lakeformation create-rls-policy \
  --name "active-orders-policy" \
  --filter-expression "status = 'active'" \
  --attachment-columns '[{"Name": "status", "Type": "string"}]'
```

Field descriptions

The following table describes each field you specify when creating an RLS policy.

Field	Required	Description
Name	Yes	A unique, case-insensitive name for the policy. Maximum 255 characters.
FilterExpression	Yes	The row filter expression. Maximum 2048 characters. Must use supported operators and reference only the declared attachment columns.
AttachmentColumns	Yes	List of columns referenced in the filter. Each entry has a `Name` (string) and `Type` (string).

Example Multi-column filter

```
aws lakeformation create-rls-policy \
  --name "engineering-us-policy" \
  --filter-expression "department = 'Engineering' AND country = 'US'" \
  --attachment-columns '[{"Name": "department", "Type": "string"}, {"Name": "country",
 "Type": "string"}]'
```

Retrieving an RLS policy

```
aws lakeformation get-rls-policy \
  --name "active-orders-policy"
```

Updating an RLS policy

You can update the filter expression of an existing policy. Attachment columns cannot be changed after creation. To change attachment columns, delete the policy and create a new one.

AWS Management Console

To update an RLS policy (console)

1. Open the Lake Formation console.
2. In the navigation pane, choose **RLS policies**.
3. Select the policy to update.
4. Choose **Edit**.
5. Modify the **Filter expression**.
6. Choose **Save**.

AWS CLI

```
aws lakeformation update-rls-policy \
  --name "active-orders-policy" \
  --filter-expression "status = 'active' OR status = 'pending'"
```

Important

Changes to the filter expression apply to all existing attachments. It can take a few seconds for principals with this policy attached to see the updated row set.

Deleting an RLS policy

Deleting a policy immediately removes all attachments and stops all filtering provided by the policy.

AWS Management Console

To delete an RLS policy (console)

1. Open the Lake Formation console.

2. In the navigation pane, choose **RLS policies**.
3. Select the policy to delete.
4. Choose **Delete**.
5. Confirm the deletion.

AWS CLI

```
aws lakeformation delete-rls-policy \
  --name "active-orders-policy"
```

> **Warning**
>
> Deleting a policy is irreversible. Any RLS attachments made against the deleted policy become inconsequential. Therefore, if no other policies are attached and no default policy exists on the table, RLS no longer filters those principals' queries, and they see all rows their Lake Formation SELECT permission allows.

Listing RLS policies and attachments

Example List RLS policies

```
aws lakeformation list-rls-policies \
  --max-results 50
```

Example List RLS policy attachments

```
aws lakeformation list-rls-policy-attachments \
  --name "active-orders-policy" \
  --max-results 50
```

Attaching a policy to a table

For detailed instructions on attaching policies (including direct attachment, tag-based attachment, ABAC conditions, and default policies), see Attaching and detaching RLS policies.

Example Quick reference: Direct attachment

```
aws lakeformation attach-rls-policy \
  --name "active-orders-policy" \
  --resource '{"Table": {"DatabaseName": "analytics_db", "Name": "orders"}}' \
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
DataAnalyst"}'
```

Example Quick reference: Tag-based attachment

```
aws lakeformation attach-rls-policy \
  --name "pii-filter-policy" \
  --resource '{"LFTagPolicy": {"ResourceType": "TABLE", "Expression": [{"TagKey":
 "data-sensitivity", "TagValues": ["high"]}]}}' \
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
RestrictedUser"}'
```

> **Note**
>
> Tag-based attachment depends on the tags assigned to a table. If those tags change or are removed, the attachment can stop applying and principals may see all rows. For the risk and how to mitigate it, see Best practices for RLS policies.

Attaching and detaching RLS policies

This section explains how to attach row-level security (RLS) policies to principals to grant row-level access, and how to detach them to revoke that access. It covers direct attachment, tag-based attachment, ABAC conditions, and default policy setup.

Overview

To grant a principal access to specific rows, you attach an RLS policy to that principal on a Data Catalog resource (a table, or a set of tables via LF-Tags). The attachment establishes the relationship between the policy, the resource, and the principal. Once attached, the policy's filter expression is enforced immediately when the principal queries those tables.

To revoke access, you detach the policy. The principal can no longer see rows granted by that policy (unless other attached policies still provide access through additive semantics).

Attaching a policy

You can attach an RLS policy using the Lake Formation console or the AWS CLI.

AWS Management Console

To attach an RLS policy (console)

1. Open the Lake Formation console.
2. In the navigation pane, choose **RLS policies**.
3. Select the policy you want to attach.
4. Choose **Attach**.
5. For **Database**, select the database containing the target table.
6. For **Table**, select the table.
7. For **Principal**, choose the principal type (IAM user, IAM role, or SAML identity) and provide the ARN or identifier.
8. (Optional) To set this as a default policy, select **Set as default policy**. When set as default, you do not specify a principal.
9. Choose **Attach**.

AWS CLI

Direct attachment:

```
aws lakeformation attach-rls-policy \
  --name "region-filter-policy" \
  --resource '{"Table": {"DatabaseName": "analytics_db", "Name": "orders"}}' \
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
DataAnalyst"}'
```

Default policy attachment:

```
aws lakeformation attach-rls-policy \
  --name "no-rows-access-policy" \
  --resource '{"Table": {"DatabaseName": "main_db", "Name": "customers"}}' \
  --set-default
```

Attaching a policy using the API or CLI

Direct attachment to a principal

Use the `AttachRLSPolicy` API to attach a policy to a specific table and principal.

API

```
{
  "Name": "region-filter-policy",
  "Resource": {
    "Table": {
      "DatabaseName": "analytics_db",
      "Name": "orders"
    }
  },
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/DataAnalyst"
  }
}
```

AWS CLI

```
aws lakeformation attach-rls-policy \
  --name "region-filter-policy" \
  --resource '{"Table": {"DatabaseName": "analytics_db", "Name": "orders"}}' \
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
DataAnalyst"}'
```

Tag-based attachment

To attach a policy to all tables matching an LF-Tag expression, use an `LFTagPolicy` resource instead of a direct table reference.

API

```
{
  "Name": "pii-filter-policy",
  "Resource": {
    "LFTagPolicy": {
```

```
      "ResourceType": "TABLE",
      "Expression": [
        {
          "TagKey": "classification",
          "TagValues": ["confidential"]
        }
      ]
    }
  },
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
RestrictedAnalyst"
  }
}
```

AWS CLI

```
aws lakeformation attach-rls-policy \
  --name "pii-filter-policy" \
  --resource '{"LFTagPolicy": {"ResourceType": "TABLE", "Expression": [{"TagKey":
 "classification", "TagValues": ["confidential"]}]}}' \
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
RestrictedAnalyst"}'
```

With tag-based attachment, the policy applies to any table that matches the tag expression. If new tables are tagged with `classification=confidential`, the policy automatically applies to them.

> **Note**
>
> Tag-based attachments are validated at query time, not at attach time. If the filter expression references columns that do not exist on a matched table, the query returns an empty result for that table rather than failing.

ABAC conditions on attachment

You can add a `Condition` field to the `AttachRLSPolicy` request to make the attachment conditional on IAM session attributes. The condition is evaluated at query time. For ABAC, you typically attach to all IAM principals in the account by setting the principal identifier to `account-`

`id:IAMPrincipals`; the condition then selects which principals the policy applies to at query time.

API

```
{
  "Name": "department-filter",
  "Resource": {
    "Table": {
      "DatabaseName": "hr_db",
      "Name": "employees"
    }
  },
  "Principal": {
    "DataLakePrincipalIdentifier": "111122223333:IAMPrincipals"
  },
  "Condition": {
    "Expression": "tag:department = 'HR'"
  }
}
```

AWS CLI

```
aws lakeformation attach-rls-policy \
  --name "department-filter" \
  --resource '{"Table": {"DatabaseName": "hr_db", "Name": "employees"}}' \
  --principal '{"DataLakePrincipalIdentifier": "111122223333:IAMPrincipals"}' \
  --condition '{"Expression": "tag:department = '\''HR'\''"}'
```

How ABAC conditions work:

- The condition evaluates IAM session tags or principal tags at query time.
- If the condition evaluates to true, the RLS policy attachment is active, and the filter is applied.
- If the condition evaluates to false, the attachment is ignored for that session.
- ABAC conditions only support IAM attributes. IAM Identity Center attributes are not supported.

Setting up a default policy

A default RLS policy applies to all principals accessing the table. Think of the default as a deny-by-default baseline: create a policy whose filter expression is `FALSE` so that principals see no rows by default, and then grant explicit policies to the principals who need access. To create the default policy attachment, use `SetDefault: true` and omit the `Principal` field.

First, create the no-rows policy, then attach it as the default.

API

Create the no-rows policy:

```
{
  "Name": "no-rows-access-policy",
  "FilterExpression": "FALSE",
  "AttachmentColumns": []
}
```

Attach it as the default:

```
{
  "Name": "no-rows-access-policy",
  "Resource": {
    "Table": {
      "DatabaseName": "main_db",
      "Name": "customers"
    }
  },
  "SetDefault": true
}
```

AWS CLI

Create the no-rows policy:

```
aws lakeformation create-rls-policy \
  --name "no-rows-access-policy" \
  --filter-expression "FALSE" \
  --attachment-columns '[]'
```

Attach it as the default:

```
aws lakeformation attach-rls-policy \
```

```
  --name "no-rows-access-policy" \
  --resource '{"Table": {"DatabaseName": "main_db", "Name": "customers"}}' \
  --set-default
```

Default policy behavior:

- All principals querying the table see only rows matching the default policy filter.
- If a principal also has an explicit policy attachment, the results are additive (union). The principal sees rows matching either the default policy or their explicit policy.
- Only one default policy can be set per table at a time.

Detaching a policy (revoking access)

AWS Management Console

To detach an RLS policy (console)

1. Open the Lake Formation console.
2. In the navigation pane, choose **RLS policies**.
3. Select the policy.
4. On the **Attachments** tab, select the attachment you want to remove.
5. Choose **Detach**.

AWS CLI

```
aws lakeformation detach-rls-policy \
  --name "region-filter-policy" \
  --resource '{"Table": {"DatabaseName": "analytics_db", "Name": "orders"}}' \
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
DataAnalyst"}'
```

Detaching a policy using the API or CLI

Use the `DetachRLSPolicy` API to remove an attachment.

API

```
{
  "Name": "region-filter-policy",
  "Resource": {
    "Table": {
      "DatabaseName": "analytics_db",
      "Name": "orders"
    }
  },
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/DataAnalyst"
  }
}
```

AWS CLI

```
aws lakeformation detach-rls-policy \
  --name "region-filter-policy" \
  --resource '{"Table": {"DatabaseName": "analytics_db", "Name": "orders"}}' \
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
DataAnalyst"}'
```

Removing a default policy

To remove a default policy, use `RemoveDefault: true` instead of specifying a principal.

API

```
{
  "Name": "no-rows-access-policy",
  "Resource": {
    "Table": {
      "DatabaseName": "main_db",
      "Name": "customers"
    }
  },
  "RemoveDefault": true
}
```

AWS CLI

```
aws lakeformation detach-rls-policy \
  --name "no-rows-access-policy" \
  --resource '{"Table": {"DatabaseName": "main_db", "Name": "customers"}}' \
  --remove-default
```

Preserving access for existing users

Because RLS policies are additive, attaching a policy to one principal never changes what other principals can see, and attaching an additional policy to a principal only expands the rows they can see. To narrow a principal's access, detach the policy granting the broader access.

If you detach all policies from a principal and no default policy exists on the table, RLS no longer filters that principal's queries. They see all rows their Lake Formation SELECT permission allows.

Cross-account RLS policies

When a table is shared across AWS accounts, row-level security (RLS) policies can be attached by the producer account, the recipient (consumer) account, or both. This section describes how RLS filtering behaves in each cross-account scenario. For general cross-account sharing setup, see Cross-account data sharing in Lake Formation.

Overview

Cross-account RLS builds on Lake Formation cross-account data sharing. The producer account owns the table and its RLS policies; the consumer account receives access to the shared table and can manage permissions and its own RLS attachments for its principals. An RLS attachment always scopes down a principal's SELECT permission and never grants access on its own.

Producer account attaches RLS for the consumer account

The producer account can attach an RLS policy that applies to principals in the consumer account. A consumer-account administrator can then grant those principals broad SELECT access on the shared table without being aware that the producer account has already attached an RLS policy to them.

In this scenario, the RLS policy attached by the producer account remains in effect. Consumer principals see only the rows the RLS filter allows, even though the consumer-account administrator

granted full SELECT on the table. The consumer-side grant provides the SELECT permission that RLS then scopes down; it does not override the producer account's attachment.

> **Important**
>
> A consumer-account administrator who grants full SELECT may not realize that rows are still being filtered by an RLS policy attached by the producer account. If a consumer principal appears to be missing rows, check for RLS attachments made by the producer account, not only for attachments in the consumer account.

RLS attached in both accounts

Both the producer account and the consumer account can attach RLS policies that apply to the same consumer principal on the shared table. When both attachments are in effect, the principal sees the intersection of the two filters – only the rows that satisfy both the producer account's RLS filter and the consumer account's RLS filter.

> **Note**
>
> Within a single account, multiple RLS policies attached to a principal combine additively (union), so the principal sees the union of all matching rows. Across accounts, the producer account's filter and the consumer account's filter combine with intersection (AND), so the principal sees only the rows that satisfy both. For details on how additive (union) behavior works within a single account, see How RLS policies work in detail.

Direct cross-account grant with a direct cross-account RLS attachment

The producer account can make a direct cross-account SELECT grant to a specific consumer-account principal and, in the same account, make a direct cross-account RLS attachment for that principal on the shared table.

In this scenario, the direct grant provides the consumer principal's SELECT permission and the direct RLS attachment scopes it down. The consumer principal sees only the rows matching the producer account's RLS filter, without the consumer account making any attachment of its own.

How producer and consumer filters combine

When RLS policies are attached in both the producer account and the consumer account for the same consumer principal, Lake Formation combines them as follows:

- **Between accounts (producer and consumer): AND.** The principal sees only rows that satisfy both the producer account's filter and the consumer account's filter (the intersection).
- **Within a single account: OR.** When one account attaches multiple RLS policies to the same principal on the table, that account's filters combine additively (the union of their matching rows).
- **A direct producer-to-principal attachment is authoritative.** When the producer account attaches an RLS policy directly to a specific consumer principal (rather than for the consumer account as a whole), the producer's filter alone determines the rows that principal can see. In this case, Lake Formation does not also apply a consumer-account attachment for that principal on the shared table.

Example: how cross-account filters change query results

The following examples use a shared `employees` table owned by the producer account. The producer account grants `SELECT` on the table to the consumer *account* (not directly to a principal), and the consumer-account administrator then grants `SELECT` on the full table to the `DataAnalyst` role. The following table shows the sample data in the `employees` table.

Row	region	dept
1	us	sales
2	us	hr
3	eu	sales
4	eu	hr
5	us	finance
6	eu	finance
7	asia	sales
8	asia	hr

The following table shows how different combinations of producer-account and consumer-account attachments change the effective filter and the rows that the `DataAnalyst` role's query returns.

Producer account attaches	Consumer account attaches (for the DataAnalyst role)	Effective filter	Rows the DataAnalyst role sees
For the consumer account, two policies: `region = 'us'`, `region = 'eu'` (combined with OR)	Two policies: `dept = 'sales'`, `dept = 'hr'` (combined with OR)	`(region = 'us' OR region = 'eu') AND (dept = 'sales' OR dept = 'hr')`	1, 2, 3, 4
For the consumer account, one policy: `region = 'us'`	One policy: `dept = 'sales'`	`region = 'us' AND dept = 'sales'`	1
Nothing (the cross-account `SELECT` grant is on the consumer account, not on the principal directly)	One policy: `dept = 'finance'`	`dept = 'finance'`	5, 6

Within each account, multiple policies attached to the principal combine with OR (the union of their rows). Across the producer and consumer accounts, the two filters combine with AND (the intersection). For how additive (union) behavior works within a single account, see How RLS policies work in detail.

Filter expression syntax for RLS policies

This section describes the syntax, operators, data types, and rules for filter expressions used in row-level security (RLS) policies. Filter expressions use the PartiQL subset supported by Lake Formation data cell filters.

Overview

A filter expression defines the condition that rows must satisfy for a principal to see them. The expression is specified in the `FilterExpression` field when creating or updating an RLS policy.

Filter expressions have the following constraints:

- Maximum length: 2048 characters
- Must reference only columns declared in `AttachmentColumns`
- Must use supported operators and data types
- Cannot contain subqueries, joins, user-defined functions, or column-to-column comparisons

Simple expressions

A simple expression compares a column to a literal value.

```
column_name operator value
```

Example Simple expressions

```
department = 'Engineering'
age >= 21
salary < 100000
country <> 'US'
```

Composite expressions

Combine multiple conditions using `AND`, `OR`, and `NOT`.

```
condition AND condition
condition OR condition
NOT condition
(condition) AND (condition)
```

Example Composite expressions

```
department = 'Engineering' AND country = 'US'
region = 'us-east-1' OR region = 'eu-west-1'
NOT (status = 'inactive')
```

```
(department = 'Sales' OR department = 'Marketing') AND country = 'US'
```

Use parentheses to control evaluation order. Without parentheses, AND has higher precedence than OR.

Supported operators

Operator	Description	Example
=	Equal to	status = 'active'
>	Greater than	amount > 1000
<	Less than	age < 65
>=	Greater than or equal to	score >= 80
<=	Less than or equal to	priority <= 3
<>	Not equal to	region <> 'internal'
!=	Not equal to (alternative)	type != 'draft'
BETWEEN	Within a range (inclusive)	age BETWEEN 18 AND 65
IN	Matches any value in a list	region IN ('us-east-1', 'us-west-2')
LIKE	Pattern matching with wildcards	email LIKE '%@example.com'
NOT	Negates a condition	NOT (status = 'deleted')
IS NULL	Checks for null value	manager_id IS NULL
IS NOT NULL	Checks for non-null value	email IS NOT NULL

BETWEEN operator

The `BETWEEN` operator tests whether a value falls within an inclusive range. `salary BETWEEN 50000 AND 100000` is equivalent to `salary >= 50000 AND salary <= 100000`.

IN operator

The `IN` operator tests whether a value matches any value in a list, for example `department IN ('Engineering', 'Sales', 'Marketing')`.

LIKE operator

The `LIKE` operator performs pattern matching. Use `%` as a wildcard for any sequence of characters and _ for a single character, for example `email LIKE '%@example.com'`.

IS NULL and IS NOT NULL

Test whether a column value is null or not null, for example `manager_id IS NULL` or `email IS NOT NULL`.

> **Note**
>
> `IS NULL` and `IS NOT NULL` cannot be used on partition columns. See Row-level security policy limitations for details.

Supported data types

The following data types are supported in the `Type` field of AttachmentColumns and determine what literal values are valid in the filter expression.

Data type	Description	Example literal
STRING	Variable-length character string	`'hello'`
CHAR	Fixed-length character string	`'A'`
VARCHAR	Variable-length character string	`'hello world'`
INT	32-bit signed integer	42

Data type	Description	Example literal
LONG	64-bit signed integer	`9223372036854775807`
BIGINT	64-bit signed integer (alias for LONG)	`1000000000000`
FLOAT	32-bit floating point	`3.14`
DECIMAL	Arbitrary precision decimal	`99.99`
DOUBLE	64-bit floating point	`2.718281828`
BOOLEAN	True or false value	`true`

Note

In the `AttachmentColumns` `Type` field, type strings are case-sensitive and must be lowercase (for example, `string`, `int`, `bigint`, `double`, `decimal`, `boolean`).

String literals

String literals must be enclosed in single quotes, for example `department = 'Engineering'`. To include a single quote within a string literal, use two consecutive single quotes (`''`), for example `name = 'O''Brien'`.

Numeric literals

Numeric literals are specified without quotes, for example `amount > 1000`, `ratio >= 0.75`, `is_active = true`.

Case sensitivity rules

- **Column names** in filter expressions are case-insensitive. `Department`, `department`, and `DEPARTMENT` all refer to the same column.
- **String literal values** in comparisons are case-sensitive. `'Engineering'` does not match `'engineering'`.

- **Keywords** (AND, OR, NOT, BETWEEN, IN, LIKE, IS, NULL) are case-insensitive.
- **Policy names** are case-insensitive.

Reserved keyword escaping

If a column name matches a reserved keyword (for example, `status`, `name`, `type`), enclose it in double quotes within the filter expression.

```
"status" = 'active'
"name" LIKE 'J%'
"type" IN ('order', 'return')
```

Using double quotes ensures the identifier is treated as a column name rather than a keyword.

Complete examples

Example Simple department filter

```
{
  "Name": "engineering-only",
  "FilterExpression": "department = 'Engineering'",
  "AttachmentColumns": [
    {
      "Name": "department",
      "Type": "string"
    }
  ]
}
```

Example Multi-condition filter with numeric comparison

```
{
  "Name": "high-value-us-orders",
  "FilterExpression": "country = 'US' AND amount >= 10000",
  "AttachmentColumns": [
    {
      "Name": "country",
      "Type": "string"
    },
```

```
    {
      "Name": "amount",
      "Type": "decimal"
    }
  ]
}
```

Example IN clause with multiple values

```
{
  "Name": "multi-region-access",
  "FilterExpression": "region IN ('us-east-1', 'us-west-2', 'eu-west-1')",
  "AttachmentColumns": [
    {
      "Name": "region",
      "Type": "string"
    }
  ]
}
```

Example BETWEEN with numeric range

```
{
  "Name": "recent-records",
  "FilterExpression": "year BETWEEN 2023 AND 2026",
  "AttachmentColumns": [
    {
      "Name": "year",
      "Type": "int"
    }
  ]
}
```

Example Pattern matching with LIKE

```
{
  "Name": "internal-emails-only",
  "FilterExpression": "email LIKE '%@example.com'",
  "AttachmentColumns": [
    {
      "Name": "email",
      "Type": "string"
```

```
    }
  ]
}
```

Example Composite filter with parentheses

```
{
  "Name": "sales-marketing-us",
  "FilterExpression": "(department = 'Sales' OR department = 'Marketing') AND country =
 'US'",
  "AttachmentColumns": [
    {
      "Name": "department",
      "Type": "string"
    },
    {
      "Name": "country",
      "Type": "string"
    }
  ]
}
```

Nested column references

If your table contains struct (nested) columns, you can reference nested fields in RLS policy filter expressions. The nested path must be specified using dot notation or double-quoted path notation in both the `AttachmentColumns` and the `FilterExpression`.

Dot notation (recommended)

```
{
  "Name": "a.city",
  "Type": "string"
}
```

Quoted path notation

```
{
```

```
  "Name": "\"a\".\"city\"",
  "Type": "string"
}
```

The following format is **not** supported:

```
{
  "Name": "a",
  "Type": "struct<city: string>"
}
```

You must reference the leaf-level nested field directly (for example, `a.city`), not the parent struct column with its full type definition.

Important rules for column names with dots

- A column name containing a dot (for example, `addr.zip`) is always interpreted as a nested column path (parent `addr`, child `zip`). If you have a top-level column literally named `addr.zip`, it cannot be used in an RLS policy because the dot is treated as a path separator.
- Column names containing literal double-quote characters (") are not supported in RLS policies.

When filtering on a nested column, use dot notation in the filter expression, for example `a.city = 'Seattle'`.

Limitations

Any expression or feature not described in this section is not supported. For the complete list of RLS policy limitations, see Row-level security policy limitations.

Best practices for RLS policies

This section provides best practices for implementing and managing row-level security (RLS) policies in Lake Formation, including guidance for customers migrating from data cell filters (DCF) to RLS policies.

Design best practices

Use tag-based attachment for scale

When attaching the same policy across many tables, attach RLS policies using Lake Formation tags rather than naming each table individually. Tag-based attachment offers the following advantages:

- New tables that are tagged inherit the policy automatically.
- You manage one attachment instead of hundreds of individual table attachments.
- You can combine tag-based RLS attachment with tag-based permission grants (LF-TBAC) for a tag-driven governance model.

When using tag-based attachment, ensure that all tables with the specified tags contain the columns referenced in the filter expression. Lake Formation validates columns at query time (not at attach time) for tag-based attachments. For a matched table that is missing a referenced column, the query returns an empty result for that table rather than failing.

> **Warning**
>
> Tag-based attachment carries the same risk as deleting a policy. If the LF-Tag that drives the attachment is removed from a table, or the table no longer matches the tag expression, the RLS policy stops applying to that table and principals with `SELECT` see all rows, with no row-level filtering. Treat changes to the tags that drive RLS attachments as access-control changes, and keep a default no-rows policy in place so that losing a tag-based attachment does not expose all rows. For how to attach by tags, see Attaching and detaching RLS policies.

Group tables by shared columns for tagging

Tag-based RLS attachment works best when tables share the same columns that you want to filter on. Design your tagging strategy around column commonality:

- Tag tables that share a `region` column with a common tag (for example, `data-scope=regional`).
- If some tables have a `region` column and others do not, do not apply a `region`-based RLS policy to a tag shared by both groups.
- You may need to use table-level tags rather than database-level tag inheritance to avoid applying RLS to tables that lack the filter columns.

Use default RLS policies for baseline security (Highly recommended)

This is the most important best practice we recommend. For tables containing sensitive data, set a default RLS policy that restricts all principals to a subset by default:

1. Create a policy (for example, `contains_pii = false`).
2. Attach it to the table with `SetDefault: true`.
3. Grant additional policies to principals who need broader access.

This ensures that new principals, both current and future, are automatically restricted to non-sensitive data without requiring policy attachment.

Combine RLS with column-level permissions for cell-level security

For additional data protection, add RLS policies along with column-level permissions:

- Use RLS policies to filter rows (for example, `region = 'US-WEST'`).
- Use column-level grants (via LF-TBAC or named grants) to restrict which columns a principal can see.
- The result is security applied at both the column and the row level. The principal sees only the intersection of allowed rows and allowed columns.

Plan for the additive (OR) permissions model

When a principal is attached to multiple RLS policies on the same table, they see the union of all matching rows. Plan your policies accordingly:

- If you want a principal to see only rows matching ALL conditions, use a single policy with an `AND` filter expression (for example, `region = 'US-WEST' AND department = 'engineering'`).
- If you want a principal to see rows matching ANY condition, attach separate policies for the union behavior.

Use ABAC for dynamic principal management

Instead of attaching each IAM role individually, use IAM principal attributes (ABAC) with the `Condition` field on `AttachRLSPolicy`. This allows an RLS policy to be applied to principals based on session tags such as department, cost center, or job role without listing each principal individually.

> **Note**
>
> Only IAM principal attributes are supported for ABAC. IAM Identity Center attributes are not currently supported.

Migrating from data cell filters to RLS policies

As a best practice, we recommend using RLS policies instead of data cell filters (DCF) for row-level security.

If you currently use data cell filters (DCF) for row-level filtering, you can migrate to RLS policies for improved scalability, centralized management, and stronger enforcement. This section explains how to achieve equivalent functionality with RLS.

Key differences between DCF and RLS

The following table compares data cell filters (DCF) and RLS policies across several dimensions.

Feature	Data cell filters	RLS policies
Scope	One filter per table	One policy can apply to thousands of tables
Attachment method	Named table only	Named table or Lake Formation tags
Row filtering behavior	TBAC grants can provide full table access, with DCF row filters restricting access because of the union of grants	TBAC grants can provide SELECT on the full table; RLS restricts access to the rows a principal is allowed to see
Column filtering	Supported directly as part of DCFs	Supported through direct column-level permissions
Updating filter criteria	Must update each data cell filter individually	Update the RLS policy once and it propagates to all attached tables

Feature	Data cell filters	RLS policies
Dynamic application	Manual per table	Automatic via tags (new tables inherit policies)

Migration workflow

Use the following workflow to migrate from DCF row filtering to RLS policies without disrupting existing access.

1. **Identify existing DCF row filters.** List your existing data cell filters that include row filter expressions. Note the `RowFilter.FilterExpression` value for each filter.

```
aws lakeformation list-data-cells-filter \
  --table '{"CatalogId":"111122223333","DatabaseName":"mydb","Name":"mytable"}'
```

2. **Create equivalent RLS policies.** For each distinct filter expression, create an RLS policy. If you had 50 DCFs with the same filter logic applied to 50 tables, you now need only one RLS policy.

```
aws lakeformation create-rls-policy \
  --name "restrict-to-us-west" \
  --filter-expression "region = 'US-WEST'" \
  --attachment-columns '[{"Name": "region", "Type": "string"}]'
```

3. **Attach the RLS policy to tables.** Attach the policy to individual tables or, for scale, to Lake Formation tags.

```
aws lakeformation attach-rls-policy \
  --name "restrict-to-us-west" \
  --resource '{"LFTagPolicy": {"ResourceType": "TABLE", "Expression": [{"TagKey":
 "department", "TagValues": ["hr"]}]}}' \
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
DataAnalyst"}'
```

4. **Grant the RLS policy to principals.** Attach the same principals that had the DCF row filter to the new RLS policy on the table.

```
aws lakeformation attach-rls-policy \
  --name "restrict-to-us-west" \
  --resource '{"Table": {"DatabaseName": "mydb", "Name": "mytable"}}' \
```

```
  --principal '{"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:role/
DataAnalyst"}'
```

5. **Remove the DCF row filter.** After verifying that the RLS policy is enforced correctly, revoke the DCF grant that included the row filter. Then re-grant SELECT without the DCF.

```
aws lakeformation revoke-permissions \
  --principal '{"DataLakePrincipalIdentifier":"arn:aws:iam::111122223333:role/
DataAnalyst"}' \
  --resource '{"Table":
{"CatalogId":"111122223333","DatabaseName":"mydb","Name":"mytable"}}' \
  --permissions SELECT
```

> **Important**
>
> If both a DCF row filter and an RLS policy apply to the same table for a principal, Lake Formation returns the union (OR) of the rows matched by each filter. During migration this means a principal can see rows from both sources until you remove the DCF row filter. Complete the migration by removing the DCF row filter so that only the RLS policy governs row access.

Handling column filtering during migration

RLS policies do not support column filtering as part of the filter criteria. If your DCF included both row filtering and column filtering:

1. Use RLS policies for the row filtering portion.
2. Use column-level permissions (via LF-TBAC or named grants) for the column filtering portion.

Combining RLS row filtering with column-level permissions provides cell-level protection equivalent to what a DCF provides, while scaling to many tables through a single reusable policy.

DCF-to-RLS equivalence examples

Example 1: Simple row filter

DCF:

```
{
  "Name": "us-only",
  "DatabaseName": "sales",
  "TableName": "orders",
  "TableCatalogId": "111122223333",
  "RowFilter": {"FilterExpression": "region='US'"},
  "ColumnWildcard": {}
}
```

RLS equivalent:

```
aws lakeformation create-rls-policy \
  --name "us-only" \
  --filter-expression "region = 'US'" \
  --attachment-columns '[{"Name": "region", "Type": "string"}]'
```

Example 2: Cell-level filter (row + column)

DCF:

```
{
  "Name": "us-no-pii",
  "DatabaseName": "sales",
  "TableName": "orders",
  "TableCatalogId": "111122223333",
  "RowFilter": {"FilterExpression": "region='US'"},
  "ColumnWildcard": {"ExcludedColumnNames": ["customer_ssn", "customer_email"]}
}
```

RLS with column permissions equivalent:

1. Create an RLS policy with `FilterExpression: "region = 'US'"`.
2. Grant SELECT on the table excluding `customer_ssn` and `customer_email` columns (via column-level permissions).
3. Attach the RLS policy to the principal on the table.

Example 3: Scaling from 50 DCFs to 1 RLS policy

Before (DCF): 50 tables, each with an identical filter `department = 'finance'`, requiring 50 separate DCF definitions.

After (RLS):

1. Create one policy: `FilterExpression: "department = 'finance'"`.
2. Tag all 50 tables with `access-scope=finance`.
3. Attach the policy via tags.
4. Grant the policy to the finance analyst role.

New tables tagged with `access-scope=finance` automatically inherit the policy.

Operational best practices

Monitor with CloudTrail

All RLS policy operations are logged in CloudTrail: policy creation, updates, and deletions; policy attachments and detachments; and policy evaluations during query execution. Use CloudTrail events to audit who changed policies and when.

Adopt RLS incrementally

RLS policies are additive. Attaching a principal to a policy restricts only that principal. Other users of the table are unaffected. This lets you adopt row-level security incrementally:

1. Start with a small set of principals.
2. Verify correct enforcement via test queries.
3. Expand to additional principals.

Validate tag-based attachments

When attaching policies via tags, verify that the target tables contain the expected columns:

- There is no console view showing which tagged tables received an RLS policy.
- Use `ListRLSPolicyAttachments` with a `Resource` filter to verify attachments.
- Test queries as an attached principal to confirm enforcement.

Use descriptive policy names

Policy names are case-insensitive and must be unique within a catalog. Use descriptive names that indicate the filtering logic, such as `restrict-to-us-west` or `finance-department-only`, rather than unclear names such as `policy-1`.

Viewing database and table permissions in Lake Formation

You can view the Lake Formation permissions that are granted on a Data Catalog database or table. You can do so by using the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

Using the console, you can view permissions starting from the **Databases** or **Tables** pages, or from the **Data permissions** page.

> **Note**
>
> If you're not a database administrator or resource owner, you can view permissions that other principals have on the resource only if you have a Lake Formation permission on the resource with the grant option.
> In addition to the required Lake Formation permissions, you need the AWS Identity and Access Management (IAM) permissions `glue:GetDatabases`, `glue:GetDatabase`, `glue:GetTables`, `glue:GetTable`, and `lakeformation:ListPermissions`.

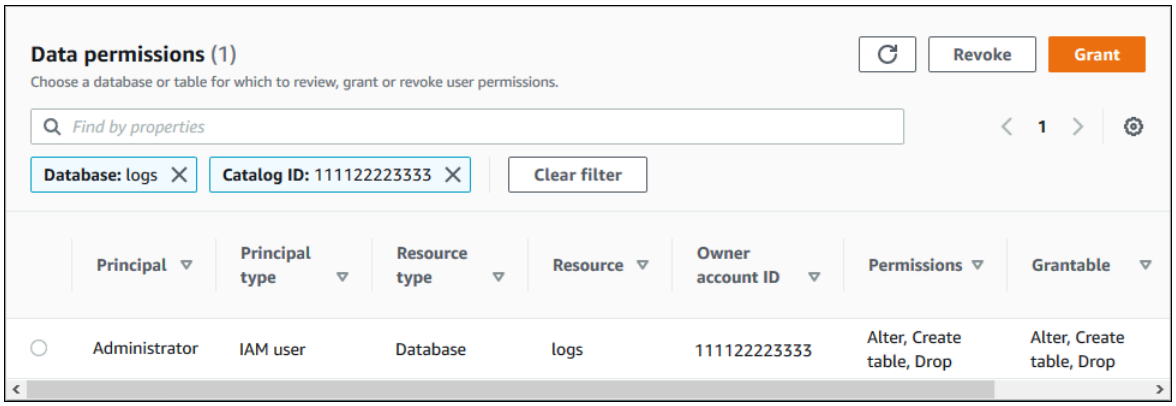
To view permissions on a database (console, starting from the Databases page)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator, the database creator, or as a user who has any Lake Formation permission on the database with the grant option.
2. In the navigation pane, choose **Databases**.
3. Choose a database, and on the **Actions** menu, choose **View permissions**.

   > **Note**
   >
   > If you choose a database resource link, Lake Formation displays the permissions on the resource link, not on the target database of the resource link.

The **Data permissions** page lists all Lake Formation permissions for the database. The database name and catalog ID (AWS account ID) of the database owner appear as labels under the search box. The tiles indicate that a filter has been applied to list permissions only for that database. You can adjust the filter by closing a tile or choosing **Clear filter**.



To view permissions on a database (console, starting from the Data permissions page)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator, the database creator, or as a user who has any Lake Formation permission on the database with the grant option.
2. In the navigation pane, choose **Data permissions**.
3. Position the cursor in the search box, and on the **Properties** menu that appears, choose **Database**.
4. On the **Databases** menu that appears, choose a database.

> **Note**
>
> If you choose a database resource link, Lake Formation displays the permissions on the resource link, not on the target database of the resource link.

The **Data permissions** page lists all Lake Formation permissions for the database. The database name appears as a tile under the search box. The tile indicates that a filter has been applied to list permissions only for that database. You can remove the filter by closing the tile or choosing **Clear filter**.

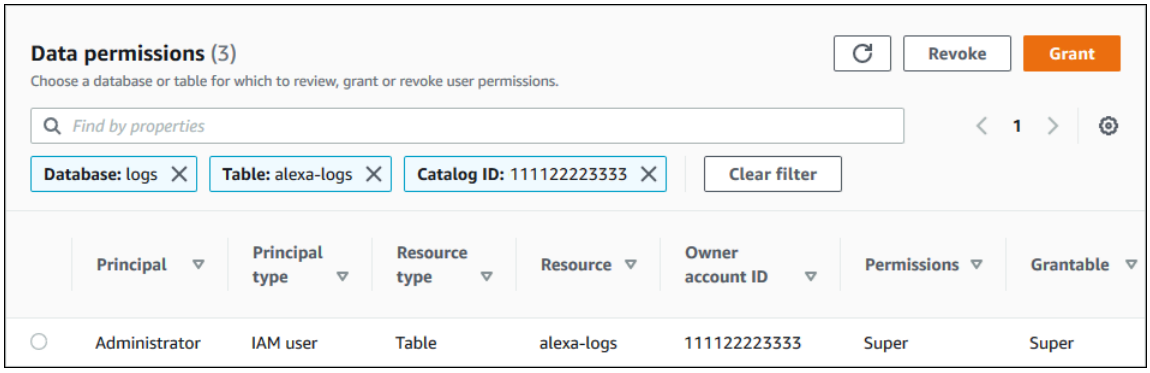
To view permissions on a table (console, starting from the Tables page)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator, the table creator, or as a user who has any Lake Formation permission on the table with the grant option.
2. In the navigation pane, choose **Tables**.
3. Choose a table, and on the **Actions** menu, choose **View permissions**.

   > **Note**
   >
   > If you choose a table resource link, Lake Formation displays the permissions on the resource link, not on the target table of the resource link.

   The **Data permissions** page lists all Lake Formation permissions for the table. The table name, the database name of the database that contains the table, and the catalog ID (AWS account ID) of the table owner appear as labels under the search box. The labels indicate that a filter has been applied to list permissions only for that table. You can adjust the filter by closing a label or choosing **Clear filter**.



To view permissions on a table (console, starting from the Data permissions page)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator, the table creator, or as a user who has any Lake Formation permission on the table with the grant option.
2. In the navigation pane, choose **Data permissions**.

3. Position the cursor in the search box, and on the **Properties** menu that appears, choose **Database**.
4. On the **Databases** menu that appears, choose a database.

   > **Important**
   >
   > If you want to view permissions on a table that was shared with your AWS account from an external account, you must choose the database in the external account that contains the table, not a resource link to the database.

   The **Data permissions** page lists all Lake Formation permissions for the database.
5. Position the cursor in the search box again, and on the **Properties** menu that appears, choose **Table**.
6. On the **Tables** menu that appears, choose a table.

   The **Data permissions** page lists all Lake Formation permissions for the table. The table name and the database name of the database that contains the table appear as tiles under the search box. The tiles indicate that a filter has been applied to list permissions only for that table. You can adjust the filter by closing a tile or choosing **Clear filter**.

To view permissions on a table (AWS CLI)

- Enter a `list-permissions` command.

  The following example lists permissions on a table shared from an external account. The `CatalogId` property is the AWS account ID of the external account, and the database name refers to the database in the external account that contains the table.

```
aws lakeformation list-permissions  --resource-type TABLE --resource '{ "Table":
 {"DatabaseName":"logs", "Name":"alexa-logs", "CatalogId":"123456789012"}}'
```

Revoking permission using the Lake Formation console

You can use the console to revoke all types of Lake Formation permissions—Data Catalog permissions, policy tag permissions, data filter permissions, and location permissions.

To revoke Lake Formation permissions on a resource (console)

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator or as a user who has been granted permissions with the grant option on the resource.
2. In the navigation pane, under **Permissions**, choose **Data lake permissions**, **LF-Tags and permissions**, or **Data locations**.
3. Select the permission or location, and then choose **Revoke**.
4. In the dialog box that opens, choose **Revoke**.

Cross-account data sharing in Lake Formation

Lake Formation cross-account capabilities allow users to securely share distributed data lakes across multiple AWS accounts, AWS organizations or directly with IAM principals in another account providing fine-grained access to the Data Catalog metadata and underlying data. Large enterprises typically use multiple AWS accounts, and many of those accounts might need access to a data lake managed by a single AWS account. Users and AWS Glue extract, transform, and load (ETL) jobs can query and join tables across multiple accounts and still take advantage of Lake Formation table-level and column-level data protections.

When you grant Lake Formation permissions on a Data Catalog resource to an external account or directly to an IAM principal in another account, Lake Formation uses the AWS Resource Access Manager (AWS RAM) service to share the resource. If the grantee account is in the same organization as the grantor account, the shared resource is available immediately to the grantee. If the grantee account is not in the same organization, AWS RAM sends an invitation to the grantee account to accept or reject the resource grant. Then, to make the shared resource available, the data lake administrator in the grantee account must use the AWS RAM console or AWS CLI to accept the invitation.

Lake Formation supports sharing Data Catalog resources with external accounts in hybrid access mode. Hybrid access mode provides the flexibility to selectively enable Lake Formation permissions for databases and tables in your AWS Glue Data Catalog.
With the Hybrid access mode, you now have an incremental path that allows you to set Lake Formation permissions for a specific set of users without interrupting the permission policies of other existing users or workloads.

For more information, see Hybrid access mode.

Direct cross-account share

Authorized principals can share resources explicitly with an IAM principal in an external account. This feature is useful when an account owner wants to have control over who in the external account can access the resources. The permissions the IAM principal receives will be a union of direct grants and the account level grants that is cascaded down to the principals. Only the recipient of the permission grant can view the direct cross-account grants. The principal who receives the resource share cannot share the resource with other principals.

Methods for sharing Data Catalog resources

With a single Lake Formation grant operation, you can grant cross-account permissions on the following Data Catalog resources.

- A database
- An individual table (with optional column filtering)
- A few selected tables
- All tables in a database (by using the All Tables wildcard)

There are two options for sharing your databases and tables with another AWS account or IAM principals in another account.

- Lake Formation tag-based access control (LF-TBAC) (recommended)

  Lake Formation tag-based access control is an authorization strategy that defines permissions based on attributes. You can use tag-based access control to share Data Catalog resources (databases, tables, and columns) with external IAM principals, AWS accounts, Organizations and organizational units (OUs). In Lake Formation, these attributes are called LF-tags. For more information, see Managing a data lake using Lake Formation tag-based access control.

  > **Note**
  >
  > The LF-TBAC method of granting Data Catalog permissions use AWS Resource Access Manager for cross-account grants.
  > Lake Formation now supports granting cross-account permissions to Organizations and organizational units using LF-TBAC method.
  > To enable this capability, you need to update the **Cross account version settings** to **Version 3** or above.

For more information, see Updating cross-account data sharing version settings.

- Lake Formation named resources

  The Lake Formation cross-account data sharing using named resource method allows you to grant Lake Formation permissions with a grant option on Data Catalog tables and databases to external AWS accounts, IAM principals, organizations, or organizational units. The grant operation automatically shares those resources.

> **Note**
>
> You can also allow the AWS Glue crawler to access a data store in a different account using Lake Formation credentials. For more information, see Cross-account crawling in AWS Glue Developer Guide.

Integrated services such as Athena and Amazon Redshift Spectrum require resource links to be able to include shared resources in queries. For more information about resource links, see How resource links work in Lake Formation.

For considerations and limitation, see Cross-account data sharing best practices and considerations.

Topics

- Prerequisites
- Updating cross-account data sharing version settings
- Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts
- Granting permissions on a database or table shared with your account
- Granting resource link permissions
- Accessing the underlying data of a shared table
- Cross-account CloudTrail logging
- Managing cross-account permissions using both AWS Glue and Lake Formation
- Viewing all cross-account grants using the GetResourceShares API operation

Related topics

- Overview of Lake Formation permissions
- Accessing and viewing shared Data Catalog tables and databases
- Creating resource links
- Troubleshooting cross-account access

Prerequisites

Before your AWS account can share Data Catalog resources (catalogs, databases and tables) with another account or principals in another account, and before you can access the resources shared with your account, the following prerequisites must be met.

General cross-account data sharing requirements

- To share Data Catalog databases and tables in hybrid access mode and share objects in the federated catalogs, you need to update the **Cross account version settings** to **Version 4**.
- Before granting cross-account permissions on a Data Catalog resource, you must revoke all Lake Formation permissions from the `IAMAllowedPrincipals` group for the resource. If the calling principal has cross account permissions to access a resource and the `IAMAllowedPrincipals` permission exists on the resource, then Lake Formation throws `AccessDeniedException`.

  This requirement is applicable only when you register the underlying data location in Lake Formation mode. If you register the data location in hybrid mode, the `IAMAllowedPrincipals` group permissions can exist on the shared database or table.
- For databases that contain tables that you intend to share, you must prevent new tables from having a default grant of `Super` to `IAMAllowedPrincipals`. On the Lake Formation console, edit the database and turn off **Use only IAM access control for new tables in this database** or enter the following AWS CLI command, replacing `database` with the name of the database. If the underlying data location is registered in hybrid access mode, you don't need to change this default setting. In hybrid access mode, Lake Formation allows you to selectively enforce Lake Formation permissions and IAM permissions policies for Amazon S3 and AWS Glue on the same resource.

```
aws glue update-database --name database --database-input
 '{"Name":"database","CreateTableDefaultPermissions":[]}'
```

- To grant cross-account permissions, the grantor must have the required AWS Identity and Access Management (IAM) permissions on AWS Glue and AWS RAM service. The AWS managed policy `AWSLakeFormationCrossAccountManager` grants the required permissions.

  Data lake administrators in accounts that receive resource shares using AWS RAM must have the following additional policy. It allows the administrator to accept AWS RAM resource share invitations. It also allows the administrator to enable resource sharing with organizations.

  JSON

  ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
                  "ram:AcceptResourceShareInvitation",
                  "ram:RejectResourceShareInvitation",
                  "ec2:DescribeAvailabilityZones",
                  "ram:EnableSharingWithAwsOrganization"
              ],
              "Resource": "*"
          }
      ]
  }
  ```

- If you want to share Data Catalog resources with AWS Organizations or organizational units, sharing with organizations must be enabled in AWS RAM.

  For information on how to enable sharing with organizations, see Enable sharing with AWS organizations in the *AWS RAM User Guide*.

  You must have the `ram:EnableSharingWithAwsOrganization` permission to enable sharing with organizations.

- To share resources directly with an IAM principal in another account, you need to update the **Cross account version settings** to **Version 3**. This setting is available on the **Data catalog settings** page. If you are using **Version 1**, see instructions to update the setting Updating cross-account data sharing version settings.

- You cannot share Data Catalog resources encrypted with AWS Glue service managed key with another account. You can share only Data Catalog resources encrypted with customer's encryption key, and the account receiving the resource share must have permissions on the Data Catalog encryption key to decrypt the objects.

Cross-account data sharing using LF-TBAC requirements

- To share Data Catalog resources with AWS Organizations and organizational units (OUs), you need to update the **Cross account version settings** to **Version 3**or above.
- To share Data Catalog resources with version 3 of the **Cross account version settings**, the grantor requires to have the IAM permissions defined in the AWS managed policy `AWSLakeFormationCrossAccountManager` in your account.
- If you are using version 1 or version 2 of the **Cross account version settings**, you must have a Data Catalog resource policy (`glue:PutResourcePolicy`) that enables LF-TBAC. For more information, see Managing cross-account permissions using both AWS Glue and Lake Formation.
- If you're currently using an AWS Glue Data Catalog resource policy to share resources, and you want to grant cross-account permissions using version 3 or higher of the **Cross account version settings**, you must add the `glue:ShareResource` permission in the Data Catalog Settings using the `glue:PutResourcePolicy` API operation as shown in the Managing cross-account permissions using both AWS Glue and Lake Formation section. This policy is not required if your account has made no cross-account grants using the AWS Glue Data Catalog resource policy (version 1 and version 2 use `glue:PutResourcePolicy` permission) to grant cross-account access.

```
{
      "Effect": "Allow",
      "Action": [
        "glue:ShareResource"
      ],
      "Principal": {"Service": [
        "ram.amazonaws.com"
      ]},
      "Resource": [
        "arn:aws:glue:<region>:<account-id>:table/*/*",
        "arn:aws:glue:<region>:<account-id>:database/*",
        "arn:aws:glue:<region>:<account-id>:catalog"
      ]
    }
```

- If your account has made cross-account shares using AWS Glue Data Catalog resource policy, and you are currently using named resource method or LF-TBAC with **Cross account settings** version 3 to share resources, which uses AWS RAM to share resources, you must set the `EnableHybrid` argument to `'true'` when you invoke the `glue:PutResourcePolicy` API operation. For more information, see Managing cross-account permissions using both AWS Glue and Lake Formation.

Setup required in each account that accesses the shared resource

- If you are sharing resources with AWS accounts, at least one user in the consumer account must be a data lake administrator to view shared resources. For information on how to create a data lake administrator, see Create a data lake administrator.

  The data lake administrator can grant Lake Formation permissions on the shared resources to other principals in the account. Other principals can't access shared resources until the data lake administrator grants them permissions on the resources.
- Integrated services such as Athena and Redshift Spectrum require resource links to be able to include shared resources in queries. Principals need to create a resource link in their Data Catalog to a shared resource from another AWS account. For more information about resource links, see How resource links work in Lake Formation.
- When a resource is shared directly with an IAM principal, to query the table using Athena, the principal needs to create a resource link. To create a resource link, the principal needs the Lake Formation `CREATE_TABLE` or `CREATE_DATABASE` permission, and the `glue:CreateTable` or `glue:CreateDatabase` IAM permission.

  If the producer account shares a different table under the same database with the same or another principal, that principal can immediately query the table.

> **Note**
>
> For the data lake administrator and for principals whom the data lake administrator has granted permissions to, shared resources appear in the Data Catalog as if they are local (owned) resources. Extract, transform, and load (ETL) jobs can access the underlying data of shared resources.
> For shared resources, the **Tables** and **Databases** pages on the Lake Formation console display the owner's account ID.

When the underlying data of a shared resource is accessed, CloudTrail log events are generated in both the shared resource recipient's account and the resource owner's account. The CloudTrail events can contain the ARN of the principal that accessed the data, but only if the recipient account opts in to include the principal ARN in the logs. For more information, see Cross-account CloudTrail logging.

Updating cross-account data sharing version settings

From time to time, AWS Lake Formation updates the cross-account data sharing settings to distinguish the changes made to the AWS RAM usage and to support updates made to the cross-account data sharing feature. When Lake Formation does this, it creates a new version of the **Cross account version settings**.

Main differences between cross-account version settings

For more information about how cross-account data sharing works under different **Cross account version settings**, see the following sections.

Note

To share data with another account, the grantor must have `AWSLakeFormationCrossAccountManager` managed IAM policy permissions. This is a prerequisite for all versions.
Updating the **Cross account version settings** does not impact the permissions the recipient has on shared resources. This is applicable when updating from version 1 to version 2, version 2 to version 3, and version 1 to version 3. See the considerations listed below when updating versions.

Version 1

Named resource method: Maps each cross-account Lake Formation permission grant to one AWS RAM resource share. User (grantor role or principal) does not require additional permissions.

LF-TBAC method: Cross-account Lake Formation permission grants don't use AWS RAM to share data. User must have `glue:PutResourcePolicy` permission.

Benefits from updating versions: Initial version - not applicable.

Considerations when updating versions: Initial version - not applicable

Version 2

Named resource method: Optimizes the number of AWS RAM resource shares by mapping multiple cross-account permission grants with one AWS RAM resource share. User does not require additional permissions.

LF-TBAC method: Cross-account Lake Formation permission grants don't use AWS RAM to share data. User must have `glue:PutResourcePolicy` permission.

Benefits from updating versions: Scalable cross-account setup by optimal utilization of AWS RAM capacity.

Considerations when updating versions: Users who want to grant cross-account Lake Formation permissions must have the permissions in the `AWSLakeFormationCrossAccountManager` AWS managed policy. Otherwise, you need to have `ram:AssociateResourceShare` and `ram:DisassociateResourceShare` permissions to successfully share resources with another account.

Version 3

Named resource method: Optimizes the number of AWS RAM resource shares by mapping multiple cross-account permission grants with one AWS RAM resource share. User does not require additional permissions.

LF-TBAC method: Lake Formation uses AWS RAM for cross-account grants. User must add glue:ShareResource statement to the `glue:PutResourcePolicy` permission. The recipient must accept resource share invitations from AWS RAM.

Benefits from updating versions: Supports the following capabilities:

- Allows sharing resources explicitly with an IAM principal in an external account.

  For more information, see Granting permissions on Data Catalog resources.
- Enables cross-account shares using LF-TBAC method to Organizations or organizational units (OUs).
- Removes the overhead of maintaining additional AWS Glue policies for cross-account grants.

Considerations when updating versions: When you use LF-TBAC method to share resources, if the grantor uses a version lower than version 3, and the recipient is using version 3 or higher, the

grantor receives the following error message: "Invalid cross account grant request. Consumer account has opt-in to cross account version: v3. Please update `CrossAccountVersion` in `DataLakeSetting` to minimal version v3 (Service: AmazonDataCatalog; Status Code: 400; Error Code: InvalidInputException)". However, if the grantor uses version 3 and the recipient is using version 1 or version 2, the cross-account grants using LF-Tags go through successfully.

Cross-account grants made using the named resource method are compatible across different versions. Even if the grantor account is using an older version (version 1 or 2) and the recipient account is using a newer version (version 3 or higher), the cross-account access functionality operates seamlessly without any compatibility issues or errors.

To share resources directly with IAM principals in another account, only the grantor needs to use version 3.

Cross-account grants made using LF-TBAC method require users to have an AWS Glue Data Catalog resource policy in the account. When you update to version 3, LF-TBAC grants uses AWS RAM. To allow AWS RAM based cross-account grants to succeed, you must add the `glue:ShareResource` statement to your existing Data Catalog resource policies as shown in the Managing cross-account permissions using both AWS Glue and Lake Formation section.

Version 4

The grantor needs version 4 or higher to share Data Catalog resources in hybrid access mode or share objects in a federated catalog.

Version 5

Cross Account Version 5 enhances cross-account resource sharing enabling you to share unlimited number of tables to another account, eliminating previous resource association limits per resource type. To get started, upgrade to cross-account version 5 through the Lake Formation console or API. Any new cross-account permission grants will automatically use wildcard patterns in the resource share instead of individual resource associations. All existing cross-account shares continue to function, and all existing Lake Formation APIs remain compatible.

Benefits from updating versions: Cross-account v5 enhances cross-account sharing, allowing you to share hundreds of thousands of tables across accounts.

Considerations when updating versions: New grants after version 5 upgrade will add wildcard resource patterns to existing AWS Resource Manager resource shares or create new shares with wildcard patterns. Once upgraded to version 5, downgrade is not supported.

Optimize AWS RAM resource shares

New versions (version 2 and above) of cross-account grants optimally utilize AWS RAM capacity to maximize cross account usage. When you share a resource with an external AWS account or an IAM principal, Lake Formation may create a new resource share or associate the resource with an existing share. By associating with existing shares, Lake Formation reduces the number of resource share invitations a consumer needs to accept. Version 5 further optimizes RAM usage by using wildcard-based resource patterns instead of individual resource associations, thereby significantly reducing resource associations per resource share.

Enable AWS RAM shares via TBAC or share resources directly to principals

To share resources directly with IAM principals in another account or to enable TBAC cross-account shares to Organizations or organizational units, you need to update the **Cross account version settings** to version 3. For more information about AWS RAM resource limits, see Cross-account data sharing best practices and considerations.

Required permissions for updating cross-account version settings

If a cross-account permission grantor has `AWSLakeFormationCrossAccountManager` managed IAM policy permissions, then there is no extra permission setup required for the cross-account permission grantor role or principal. However, if the cross-account grantor is not using the managed policy, then the grantor role or principal should have following IAM permissions granted for the new version of the cross-account grant to be successful.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
         "ram:AssociateResourceShare",
         "ram:DisassociateResourceShare",
         "ram:GetResourceShares"
       ],
     "Resource": "*",
```

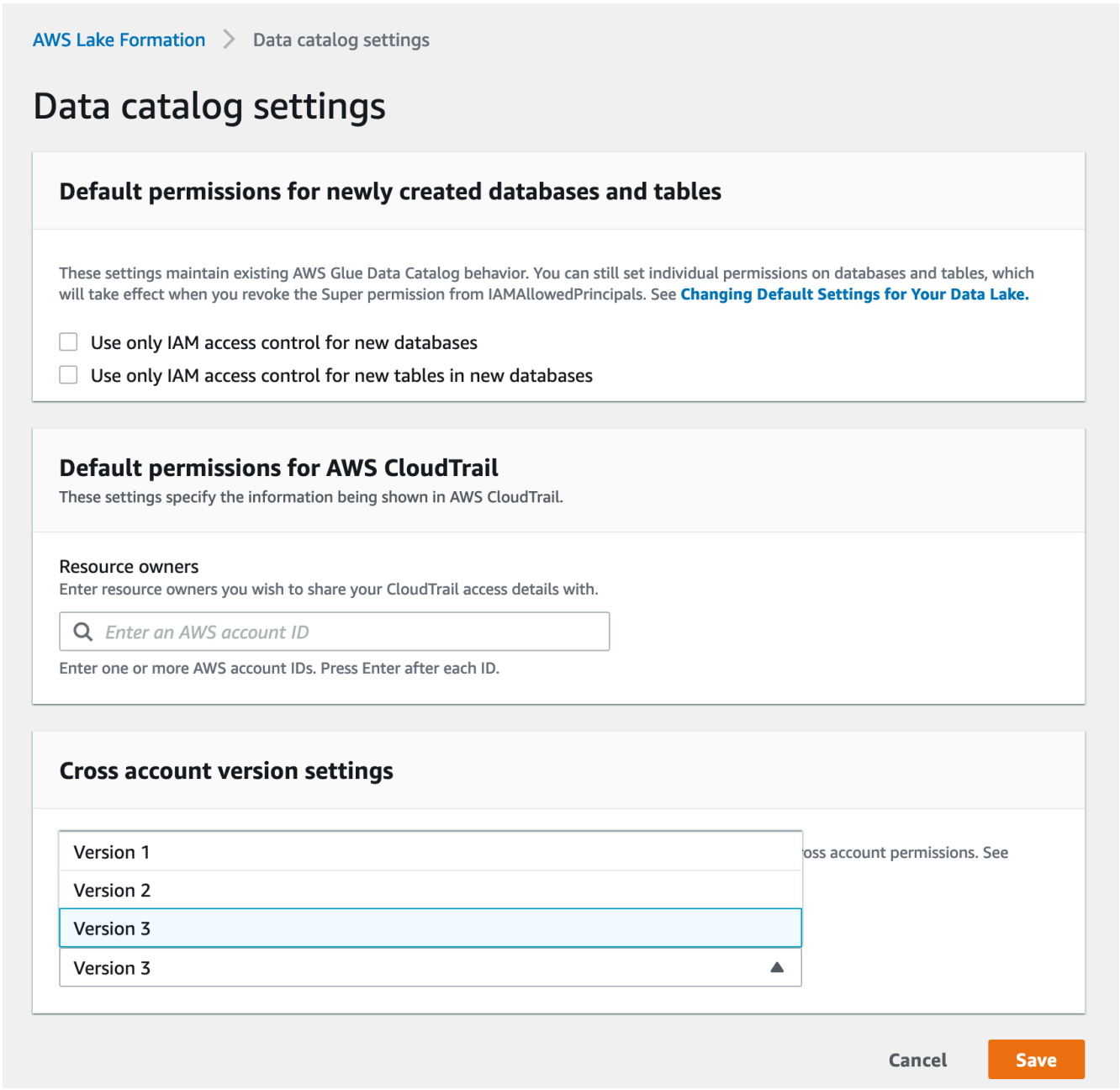
```
      "Condition": {
        "StringLike": {
          "ram:ResourceShareName": "LakeFormation*"
        }
      }
    }
  ]
}
```

To enable the new version

Follow these steps to update **Cross account version settings** through the AWS Lake Formation console or the AWS CLI.

Console

1. Choose **Version 2**, **Version 3**, **Version 4**, or **Version 5** under **Cross account version settings** on the **Data catalog settings** page. If you select **Version 1**, Lake Formation will use the default resource sharing mode.



2. Choose **Save**.

AWS Command Line Interface (AWS CLI)

Use the `put-data-lake-settings` AWS CLI command to set the `CROSS_ACCOUNT_VERSION` parameter. Accepted values are 1, 2, 3, 4, and 5.

```
aws lakeformation put-data-lake-settings --region us-east-1 --data-lake-settings
 file://settings
{
```

```
    "DataLakeAdmins": [
        {
            "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/test"
        }
    ],
    "CreateDatabaseDefaultPermissions": [],
    "CreateTableDefaultPermissions": [],
    "Parameters": {
        "CROSS_ACCOUNT_VERSION": "3"
    }
}
```

Important

Once you choose **Version 2** or **Version 3**, all new **named resource** grants will go through the new cross-account grant mode. To optimally use AWS RAM capacity for your existing cross-account shares, we recommend you to revoke the grants that were made with the older version, and re-grant in the new mode.

Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts

This section includes instructions on how to grant cross-account permissions on Data Catalog resources to an external AWS account, IAM principal, AWS organization, or organizational unit. The grant operation automatically shares those resources.

Topics

- Data sharing using tag-based access control
- Cross-account data sharing using the named resource method

Data sharing using tag-based access control

AWS Lake Formation tag-based access control (LF-TBAC) is an authorization strategy that defines permissions based on attributes. The following steps explain how to grant cross-account permissions by using LF-Tags.

Set up required on the producer/grantor account

1. Add LF-Tags.

   1. Sign in to Lake Formation console as a data lake administrator or a LF-Tag creator.
   2. In the left navigation bar, choose **Permissions**, and **LF-Tags and permissions**.
   3. Choose **Add LF-Tag**.

      For detailed instructions to create LF-Tags, see Creating LF-Tags.

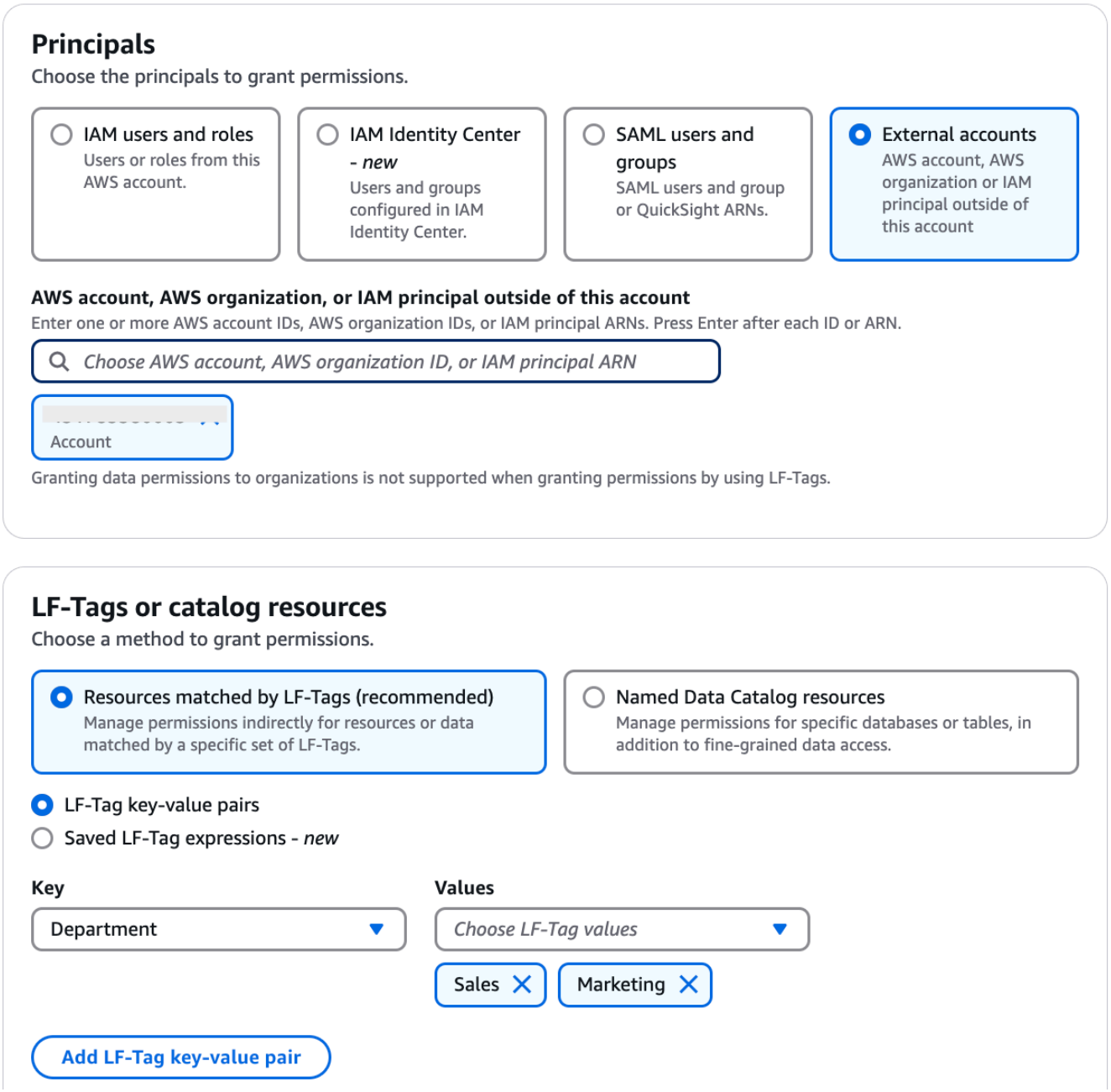
2. Grant **Describe** and/or **Associate** permissions **LF-Tag key-value** pairs to IAM principals in your account or external accounts.

   Granting permissions on **LF-Tag key-value** pairs enables the principals to view the LF-Tags, and assign them to Data Catalog resources (databases, tables, and columns).

3. Next, the data lake administrator or an IAM principal with **Associate** permission can assign the LF-Tag to databases, tables, or columns. For more information, see Assigning LF-Tags to Data Catalog resources.

4. Next, grant data permission to external accounts using LF-Tag expressions. This enables the grantee or recipient of the permissions to access the Data Catalog resource(s) that are tagged with the same key(s) and value(s).

   1. In the navigation pane, choose **Permissions** and **Data permissions**.
   2. Choose **Grant**.
   3. On the **Grant permissions** page, for **Principals**, choose **External accounts**, and enter the grantee AWS account ID or the IAM role of the principal or the Amazon Resource Name (ARN) for the principal (principal ARN) if making a direct cross-account grant to an external principal. You need to press **Enter** after entering the account ID.



4. For **LF-Tags or catalog resources**, choose **Resources matched by LF-Tags (recommended)**.
   a. Choose the option **LF-Tag key-value pairs** or **Saved LF-Tag expressions** .
   b. **If you choose, LF-Tag key-value pairs, enter the key and value(s)** of the **LF-Tag** that is associated with the Data Catalog resource being shared with the grantee account.

      The grantee is granted permissions on the Data Catalog resources that were assigned a matching LF-Tag in the LF-Tag expression. If the LF-Tag expression specifies multiple values per tag key, any one of the tag values can be a match.

5. Choose the database-level or table-level permissions to grant on resources that match the LF-Tag expression.

> **Important**
>
> Because the data lake administrator must grant permissions on shared resources to the principals in the grantee account, you must always grant cross-account permissions with the grant option.

For more information, see Granting LF-Tag permissions using the console.

> **Note**
>
> Principals who receive direct cross-account grants will not have the **Grantable permissions** option.

Set up required on the receiving/grantee account

1. Sign in to Lake Formation console as a data lake administrator of the consumer account.
2. Next, receive the resource share in the consumer account.
    1. Open the AWS RAM console.
    2. In the navigation pane, under **Shared with me**, choose **Resource shares**.
    3. Select the resource shares, choose **Accept resource share**.
3. When you share a resource with another account, the resource still belongs to the producer account and is not visible within the Athena console. To make the resource visible in the Athena console, you need to create a resource link pointing to the shared resource. For instructions on creating a resource link, see Creating a resource link to a shared Data Catalog table and Creating a resource link to a shared Data Catalog database
    1. Choose **Databases** or **Tables** under the Data Catalog.
    2. On the Databases/Tables page, choose **Create**, **Resource link** .
    3. Enter the following information for a database resource link:
        - **Resource link name** – A unique name for the resource link.

- **Destination catalog** – The catalog where you're creating the resource link.
- **Shared database Region** – The Region of the database shared with you if you are creating the resource link in a different Region.
- **Shared database** – Choose the shared database.
- **Shared database's catalog ID** – Enter the catalog ID for the shared database.

4. Choose **Create**. You can see the newly created resource link in the databases list.

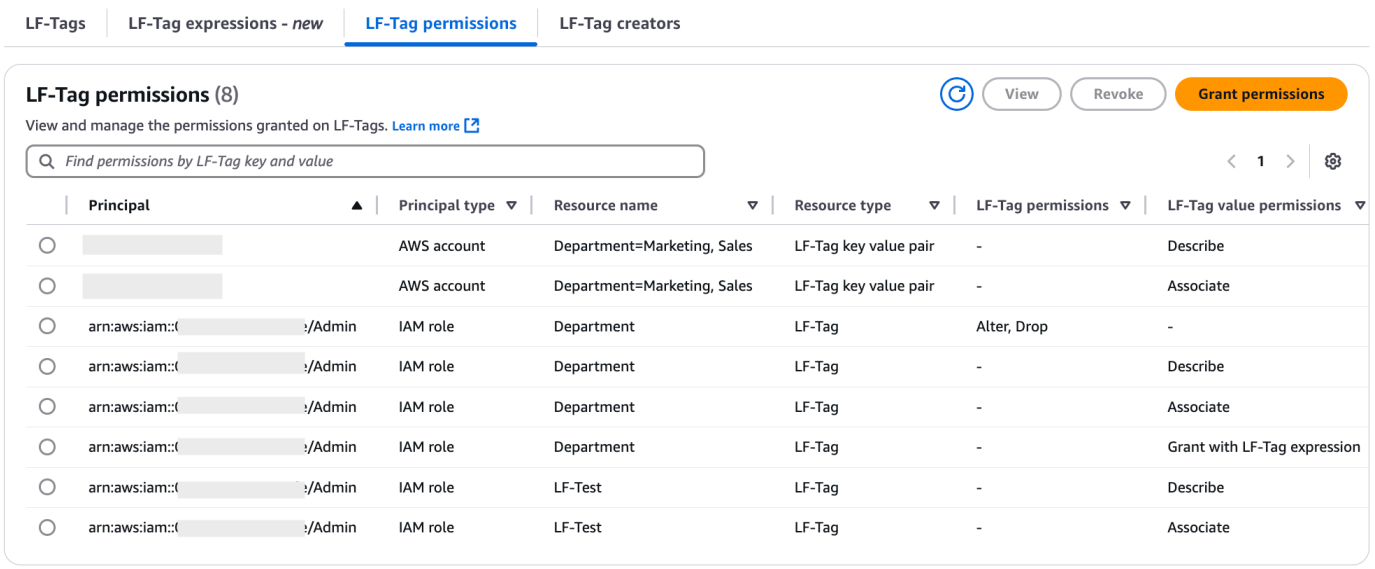
Similarly, you can create a resource link to a shared table.

4. Now grant **Describe** permission on the resource link to the IAM principals that you are sharing the resource.

   1. On the **Databases/Tables** page, select the resource link, and on the **Actions** menu, choose **Grant**.
   2. In the **Grant permissions** section, select **IAM users and roles**.
   3. Choose the IAM role that you want to grant access to the resource link.
   4. In the **Resource link** permissions section, select **Describe**.
   5. Choose **Grant**.

5. Next, grant **LF-Tag key-value permissions** to the principals in the consumer account.

   You should be able to find the LF-Tags that are shared with you in the consumer account on the Lake Formation console, under **Permissions**, **LF-Tags and permissions**. You can associate tags shared from grantor on resources shared from grantor account that includes: databases, tables, and columns. You can further grant permissions on the resources to other principals.



1. In the navigation pane, under **Permissions**, **Data permissions**, choose **Grant**.
2. On the **Grant permissions** page, choose **IAM users and roles**.
3. Next, choose the IAM users and roles in your account to grant access to the shared databases/tables.
4. Next, for **LF-Tags or catalog resources**, choose **Resources matched by LF-Tags**.
5. Next, choose the key and values of the LF-Tag that is shared with you.
6. Next, choose the database and table permissions that you want to grant to the IAM users and roles. You can also choose **Grantable permissions** that enables the IAM users and roles to grant permissions to other users/roles.
7. Choose **Grant**.
8. You can view the permission grants under **Data permissions** on the Lake Formation console.

Cross-account data sharing using the named resource method

You can grant permissions to directly to principals in the another AWS account, or to external AWS accounts or AWS Organizations. Granting Lake Formation permissions to Organizations or organizational units is equivalent to granting the permission to every AWS account in that organization or organizational unit.

When you grant permissions to external accounts or organizations, you must include the **Grantable permissions** option. Only the data lake administrator in the external account can access the shared

resources until the administrator grants permissions on the shared resources to other principals in the external account.

> **Note**
>
> **Grantable permissions** option is not supported when granting permissions directly to IAM principals from external accounts.

Follow instructions in Granting database permissions using the named resource method to grant cross-account permissions using the named resource method.

Granting permissions on a database or table shared with your account

After a Data Catalog resource belonging to another AWS account is shared with your AWS account, as a data lake administrator, you can grant permissions on the shared resource to other principals in your account. You can't, however, grant permissions on the resource to other AWS accounts or organizations.

You can use the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI) to grant the permissions.

To grant permissions on a shared database (named resource method, console)

- Follow the instructions in Granting database permissions using the named resource method. In the **Database** list under **LF-Tags or catalog resources**, ensure that you select the database in the external account, not a resource link for the database.

  If you don't see the database in the list of databases, ensure that you have accepted the AWS Resource Access Manager (AWS RAM) resource share invitation for the database. For more information, see Accepting a resource share invitation from AWS RAM.

  Also, for the `CREATE_TABLE` and `ALTER` permissions, follow the instructions in Granting data location permissions (same account), and be sure to enter the owning account ID in the **Registered account location** field.

To grant permissions on a shared table (named resource method, console)

- Follow the instructions in Granting table permissions using the named resource method. In the **Database** list under **LF-Tags or catalog resources**, ensure that you select the database in the external account, not a resource link for the database.

  If you don't see the table in the list of tables, ensure that you have accepted the AWS RAM resource share invitation for the table. For more information, see Accepting a resource share invitation from AWS RAM.

  Also, for the `ALTER` permission, follow the instructions in Granting data location permissions (same account), and be sure to enter the owning account ID in the **Registered account location** field.

To grant permissions on shared resources (LF-TBAC method, console)

- Follow the instructions in Granting Data Catalog permissions. In the **LF-Tags or catalog resources** section, grant the exact LF-Tag expression that the external account granted to your account, or a subset of that expression.

  For example, if an external account granted the LF-Tag expression `module=customers AND environment=production` to your account with the grant option, as a data lake administrator, you can grant that same expression, or `module=customers` or `environment=production` to a principal in your account. You can grant only the same or a subset of the Lake Formation permissions (for example, `SELECT`, `ALTER`, and so on) that were granted on resources through the LF-Tag expression.

To grant permissions on a shared table (named resource method, AWS CLI)

- Enter a command similar to the following. In this example:
  - Your AWS account ID is 1111-2222-3333.
  - The account that owns the table and that granted it to your account is 1234-5678-9012.
  - The `SELECT` permission is being granted on the shared table `pageviews` to user `datalake_user1`. That user is a principal in your account.
  - The `pageviews` table is in the `analytics` database, which is owned by account 1234-5678-9012.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "SELECT" --resource '{ "Table": {"CatalogId":"123456789012",
 "DatabaseName":"analytics", "Name":"pageviews"}}'
```

Note that the owning account must be specified in the `CatalogId` property in the `resource` argument.

Granting resource link permissions

Follow these steps to grant AWS Lake Formation permissions on one or more resource links to a principal in your AWS account.

After you create a resource link, only you can view and access it. (This assumes that **Use only IAM access control for new tables in this database** is not enabled for the database.) To permit other principals in your account to access the resource link, grant at least the `DESCRIBE` permission.

> **Important**
>
> Granting permissions on a resource link doesn't grant permissions on the target (linked) database or table. You must grant permissions on the target separately.

You can grant permissions by using the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

console

To grant resource link permissions using the Lake Formation console

1. Do one of the following:

   - For database resource links, follow the steps in Granting database permissions using the named resource method. to do the following:

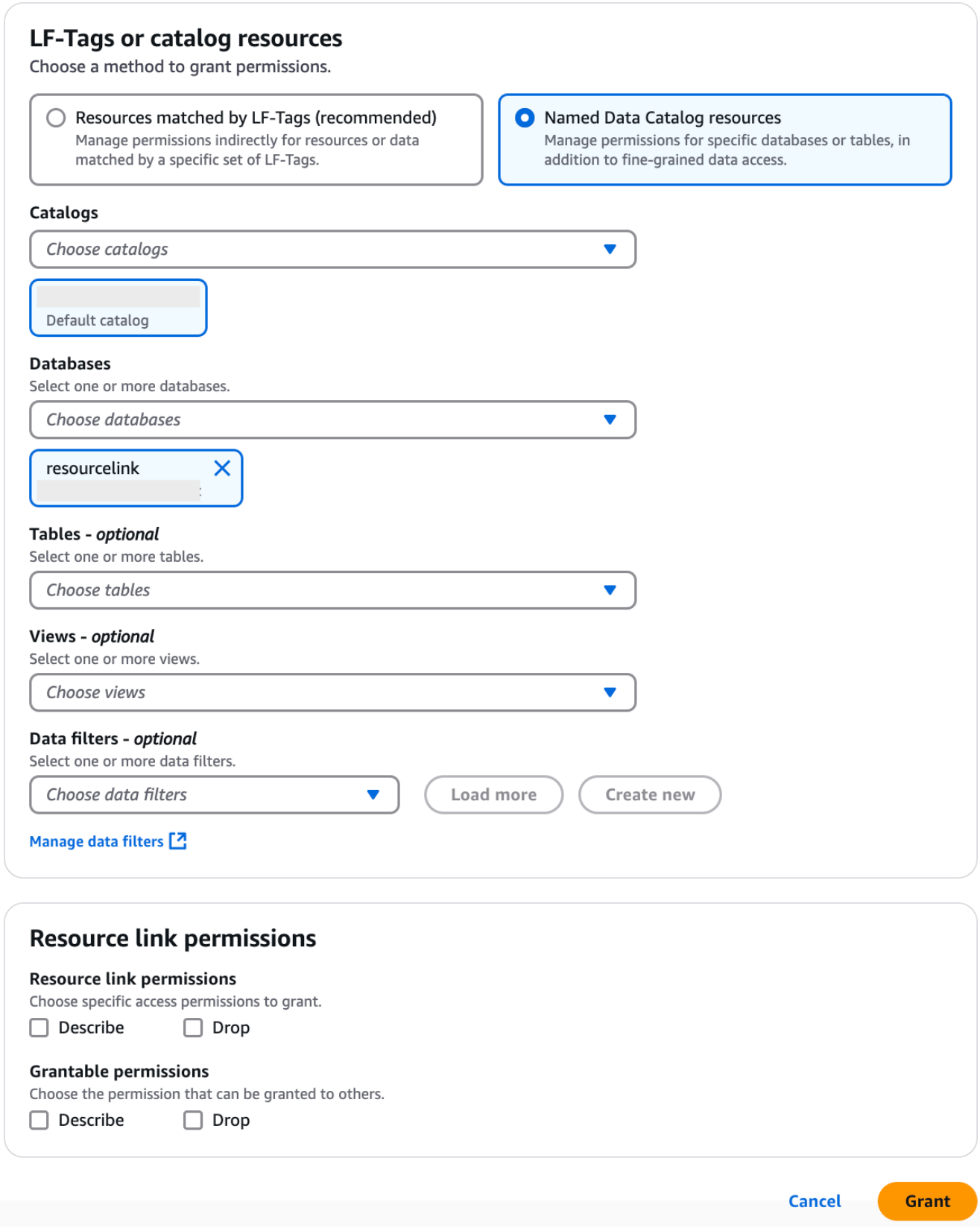
     1. Select the resource link from the databases list under Data Catalog, **Databases**.
     2. Choose **Grant** to open the **Grant permissions** page.
     3. Specify principals to grant permissions.

4. The **Catalogs** and **Databases** fields are populated.

- For table resource links, follow the steps in Granting table permissions using the named resource method to do the following:

  1. Select the resource link from the tables list under Data Catalog, **Tables**.
  2. Open the **Grant permissions** page.
  3. Specify principals.
  4. The **Catalogs**, **Databases** , **Tables** fields are populated.
  5. Specify principals.

2. Under **Permissions**, select the permissions to grant. Optionally, select grantable permissions.



3. Choose **Grant**.

AWS CLI

To grant resource link permissions using AWS CLI

- Run the `grant-permissions` command, specifying a resource link as the resource.

  Example

  This example grants `DESCRIBE` to user `datalake_user1` on the table resource link `incidents-link` in the database `issues` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
 --permissions "DESCRIBE" --resource '{ "Table": {"DatabaseName":"issues",
 "Name":"incidents-link"}}'
```

See Also:

- Creating resource links
- Lake Formation permissions reference

Accessing the underlying data of a shared table

Assume that AWS account A shares a Data Catalog table with account B—for example, by granting `SELECT` with the grant option on the table to account B. For a principal in account B to be able to read the shared table's underlying data, the following conditions must be met:

- The data lake administrator in account B must accept the share. (This isn't necessary if accounts A and B are in the same organization or if the grant was made with the Lake Formation tag-based access control method.)
- The data lake administrator must re-grant to the principal the Lake Formation `SELECT` permission that account A granted on the shared table.
- The principal must have the following IAM permissions on the table, the database that contains it, and the account A Data Catalog.

Note

In the following IAM policy:

- Replace *<account-id-A>* with the AWS account ID of account A.
- Replace *<region>* with a valid Region.
- Replace *<database>* with the name of the database in account A that contains the shared table.
- Replace *<table>* with the name of the shared table.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "glue:GetTable",
            "glue:GetTables",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:GetDatabase",
            "glue:GetDatabases"
           ],
           "Resource": [
            "arn:aws:glue:us-east-1:111122223333:table/<database>/<table>",
            "arn:aws:glue:us-east-1:111122223333:database/<database>",
            "arn:aws:glue:us-east-1:111122223333:catalog"
           ]
        },
        {
          "Effect": "Allow",
          "Action": [
            "lakeformation:GetDataAccess"
           ],
          "Resource": [
            "*"
```

```
            ]
        }
    ]
}
```

See Also:

- Accepting a resource share invitation from AWS RAM

Cross-account CloudTrail logging

Lake Formation provides a centralized audit trail of all cross-account access to data in your data lake. When a recipient AWS account accesses data in a shared table, Lake Formation copies the CloudTrail event to the owning account's CloudTrail logs. Copied events include queries against the data by integrated services such as Amazon Athena and Amazon Redshift Spectrum, and data accesses by AWS Glue jobs.

CloudTrail events for cross-account operations on Data Catalog resources are similarly copied.

As a resource owner, if you enable object-level logging in Amazon S3, you can run queries that join S3 CloudTrail events with Lake Formation CloudTrail events to determine the accounts that have accessed your S3 buckets.

Topics

- Including principal identities in cross-account CloudTrail logs
- Querying CloudTrail logs for Amazon S3 cross-account access

Including principal identities in cross-account CloudTrail logs

By default, cross-account CloudTrail events added to the shared resource recipient's logs and copied to resource owner's logs contain only the AWS principal ID of the external account principal —not the human-readable Amazon Resource Name (ARN) of the principal (principal ARN). When sharing resources within trusted boundaries, such as within the same organization or team, you can opt in to include the principal ARN in the CloudTrail events. Resource owner accounts can then track the principals in recipient accounts that access their owned resources.

> **Important**
>
> As a shared resource recipient, to see the principal ARN in events in your own CloudTrail logs, you must opt in to share the principal ARN with the owner account.
> If the data access occurs through a resource link, two events are logged in the shared resource recipient account: one for the resource link access and one for the target resource access. The event for the resource link access *does* include the principal ARN. The event for the target resource access does not include the principal ARN without the opt-in. The resource link access event is not copied to the owner account.

The following is an excerpt from a default cross-account CloudTrail event (without opt-in). The account performing the data access is 1111-2222-3333. This is the log that is shown in both the calling account and the resource owner account. Lake Formation populates logs in both accounts in the cross-account case.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AWSAccount",
        "principalId": "AROAQGFTBBBGOBWV2EMZA:GlueJobRunnerSession",
        "accountId": "111122223333"
    },
    "eventSource": "lakeformation.amazonaws.com",
    "eventName": "GetDataAccess",
...
...
    "additionalEventData": {
        "requesterService": "GLUE_JOB",
        "lakeFormationRoleSessionName": "AWSLF-00-GL-111122223333-G13T0Rmng2"
    },
...
}
```

As a shared resource consumer, when you opt in to include the principal ARN, the excerpt becomes the following. The `lakeFormationPrincipal` field represents the end role or user performing the query through Amazon Athena, Amazon Redshift Spectrum, or AWS Glue jobs.

```
{
    "eventVersion": "1.05",
```

```
    "userIdentity": {
        "type": "AWSAccount",
        "principalId": "AROAQGFTBBBGOBWV2EMZA:GlueJobRunnerSession",
        "accountId": "111122223333"
    },
    "eventSource": "lakeformation.amazonaws.com",
    "eventName": "GetDataAccess",
...
...
    "additionalEventData": {
        "requesterService": "GLUE_JOB",
        "lakeFormationPrincipal": "arn:aws:iam::111122223333:role/ETL-Glue-Role",
        "lakeFormationRoleSessionName": "AWSLF-00-GL-111122223333-G13T0Rmng2"
    },
...
}
```

To opt in to include principal ARNs in cross-account CloudTrail logs

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as the `Administrator` user, or a user with the `Administrator Access` IAM policy.
2. In the navigation pane, choose **Settings**.
3. On the **Data catalog settings** page, in the **Default permissions for AWS CloudTrail** section, for **Resource owners**, enter one or more AWS resource owner account IDs.

   Press **Enter** after each account ID.
4. Choose **Save**.

   Now cross-account CloudTrail events stored in the logs for both the shared resource recipient and the resource owner contain the principal ARN.

Querying CloudTrail logs for Amazon S3 cross-account access

As a shared resource owner, you can query S3 CloudTrail logs to determine the accounts that have accessed your Amazon S3 buckets (provided that you enabled object-level logging in Amazon S3). This applies only to S3 locations that you registered with Lake Formation. If shared resource consumers opt in to include principal ARNs in Lake Formation CloudTrail logs, you can determine the roles or users that accessed the buckets.

When running queries with Amazon Athena, you can join Lake Formation CloudTrail events and S3 CloudTrail events on the session name property. Queries can also filter Lake Formation events on `eventName="GetDataAccess"`, and S3 events on `eventName="Get Object"` or `eventName="Put Object"`.

The following is an excerpt from a Lake Formation cross-account CloudTrail event where data in a registered S3 location was accessed.

```
{
  "eventSource": "lakeformation.amazonaws.com",
  "eventName": "GetDataAccess",
  ..............
  ..............
  "additionalEventData": {
    "requesterService": "GLUE_JOB",
    "lakeFormationPrincipal": "arn:aws:iam::111122223333:role/ETL-Glue-Role",
    "lakeFormationRoleSessionName": "AWSLF-00-GL-111122223333-B8JSAjo5QA"
   }
}
```

The `lakeFormationRoleSessionName` key value, `AWSLF-00-GL-111122223333-B8JSAjo5QA`, can be joined with the session name in the `principalId` key of the S3 CloudTrail event. The following is an excerpt from the S3 CloudTrail event. It shows the location of the session name.

```
{
   "eventSource": "s3.amazonaws.com",
   "eventName": "Get Object"
   ..............
   ..............
   "principalId": "AROAQSOX5XXUR7D6RMYLR:AWSLF-00-GL-111122223333-B8JSAjo5QA",
   "arn": "arn:aws:sets::111122223333:assumed-role/Deformationally/AWSLF-00-
GL-111122223333-B8JSAjo5QA",
   "session Context": {
     "session Issuer": {
       "type": "Role",
       "principalId": "AROAQSOX5XXUR7D6RMYLR",
       "arn": "arn:aws:iam::111122223333:role/aws-service-role/
lakeformation.amazonaws.com/Deformationally",
       "accountId": "111122223333",
       "user Name": "Deformationally"
```

```
    },
   ..............
   ..............
}
```

The session name is formatted as follows:

```
AWSLF-<version-number>-<query-engine-code>-<account-id->-<suffix>
```

`version-number`

The version of this format, currently `00`. If the session name format changes, the next version will be `01`.

`query-engine-code`

Indicates the entity that accessed the data. Current values are:

GL	AWS Glue ETL job
AT	Athena
RE	Amazon Redshift Spectrum

`account-id`

The AWS account ID that requested credentials from Lake Formation.

`suffix`

A randomly generated string.

Managing cross-account permissions using both AWS Glue and Lake Formation

It's possible to grant cross-account access to Data Catalog resources and underlying data by using either AWS Glue or AWS Lake Formation.

In AWS Glue, you grant cross-account permissions by creating or updating a Data Catalog resource policy. In Lake Formation, you grant cross-account permissions by using the Lake Formation GRANT/REVOKE permissions model and the `Grant Permissions` API operation.

> **Tip**
>
> We recommend that rely solely on Lake Formation permissions to secure your data lake.

You can view Lake Formation cross-account grants by using the Lake Formation console or the AWS Resource Access Manager (AWS RAM) console. However, those console pages don't show cross-account permissions granted by the AWS Glue Data Catalog resource policy. Similarly, you can view the cross-account grants in the Data Catalog resource policy using the **Settings** page of the AWS Glue console, but that page doesn't show the cross-account permissions granted using Lake Formation.

To ensure that you don't miss any grants when viewing and managing cross-account permissions, Lake Formation and AWS Glue require you to perform the following actions to indicate that you are aware of and are permitting cross-account grants by both Lake Formation and AWS Glue.

When granting cross-account permissions using the AWS Glue Data Catalog resource policy

If your account (grantor account or producer account) has made no cross-account grants that uses AWS RAM to share the resources, you can save a Data Catalog resource policy as usual in AWS Glue. However, if grants that involve AWS RAM resource shares have already been made, you must do one of the following to ensure that saving the resource policy succeeds:

- When you save the resource policy on the **Settings** page of the AWS Glue console, the console issues an alert stating that the permissions in the policy will be in addition to any permissions granted using the Lake Formation console. You must choose **Proceed** to save the policy.
- When you save the resource policy using the `glue:PutResourcePolicy` API operation, you must set the `EnableHybrid` field to 'TRUE' (type = string).

  To update an existing resource policy, use the `glue:GetResourcePolicy` API operation to retrieve your current policy first, then modify it as needed before calling `glue:PutResourcePolicy`.

> **Note**
>
> When creating AWS Glue resource policies for cross-account access, grant only the minimum permissions required for your specific use case.

For more information, see PutResourcePolicy Action (Python: put_resource_policy) in the *AWS Glue Developer Guide*.

When granting cross-account permissions using the Lake Formation named resources method

If there is no Data Catalog resource policy in your account (producer account), Lake Formation cross-account grants that you make proceed as usual. However, if a Data Catalog resource policy exists, you must add the following statement to it to permit your cross-account grants to succeed if they are made with the named resource method. Replace *<region>* with a valid Region name and *<account-id>* with your AWS account ID (producer account ID).

```
{
  "Effect": "Allow",
  "Action": [
    "glue:ShareResource"
  ],
  "Principal": {"Service": [
    "ram.amazonaws.com"
  ]},
  "Resource": [
    "arn:aws:glue:<region>:<account-id>:table/*/*",
    "arn:aws:glue:<region>:<account-id>:database/*",
    "arn:aws:glue:<region>:<account-id>:catalog"
  ]
}
```

Without this additional statement, the Lake Formation grant succeeds, but becomes blocked in AWS RAM, and the recipient account can't access the granted resource.

Important

When using the Lake Formation tag-based access control (LF-TBAC) method to make cross-account grants, you must have a Data Catalog resource policy with at least the permissions specified in Prerequisites.

See Also:

- Metadata access control (for a discussion of the named resource method versus the Lake Formation tag-based access control (LF-TBAC) method).
- Viewing shared Data Catalog tables and databases
- Working with Data Catalog Settings on the AWS Glue Console in the *AWS Glue Developer Guide*
- Granting Cross-Account Access in the *AWS Glue Developer Guide* (for sample Data Catalog resource policies)

Viewing all cross-account grants using the GetResourceShares API operation

If your enterprise grants cross-account permissions using both an AWS Glue Data Catalog resource policy and Lake Formation grants, the only way to view all cross-account grants in one place is to use the `glue:GetResourceShares` API operation.

When you grant Lake Formation permissions across accounts by using the named resource method, AWS Resource Access Manager (AWS RAM) creates an AWS Identity and Access Management (IAM) resource policy and stores it in your AWS account. The policy grants the permissions required to access the resource. AWS RAM creates a separate resource policy for each cross-account grant. You can view all of these policies by using the `glue:GetResourceShares` API operation.

Note

This operation also returns the Data Catalog resource policy. However, if you enabled meta data encryption in Data Catalog settings, and you don't have permission on the AWS KMS key, the operation won't return the Data Catalog resource policy.

To view all cross-account grants

- Enter the following AWS CLI command.

```
aws glue get-resource-policies
```

The following is an example resource policy that AWS RAM creates and stores when you grant permissions on table `t` in database db1 to AWS account 1111-2222-3333.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "glue:GetTable",
         "glue:GetTables",
         "glue:GetTableVersion",
         "glue:GetTableVersions",
         "glue:GetPartition",
         "glue:GetPartitions",
         "glue:BatchGetPartition",
         "glue:SearchTables"
       ],
      "Principal": {"AWS": [
        "111122223333"
      ]},
      "Resource": [
      "arn:aws:glue:us-east-1:111122223333:table/db1/t"
     ]
    }
  ]
}
```

See also:

- GetResourceShares Action (Python: get_resource_policies) in the *AWS Glue Developer Guide*

Accessing and viewing shared Data Catalog tables and databases

For the data lake administrator and for principals who have been granted permissions, resources that are shared with your AWS account appear in the Data Catalog as if they were resources in your account. The console displays the account that owns the resource.

You can view resources that are shared with your account by using the Lake Formation console. You can also use the AWS Resource Access Manager (AWS RAM) console to view both resources that are shared with your account and resources that you've shared with other AWS accounts by using the named resource method.

Important

When someone uses the named resource method to grant cross-account permissions on a Data Catalog resource to your account or AWS organization, Lake Formation uses the AWS Resource Access Manager (AWS RAM) service to share the resource. If your account is in the same AWS organization as the granting account, the shared resource is available to you immediately.
However, if your account is not in the same organization, AWS RAM sends an invitation to your account to accept or reject the resource share. Then, to make the shared resource available, the data lake administrator in your account must use the AWS RAM console or CLI to accept the invitation.
The Lake Formation console displays an alert if there is an AWS RAM resource share invitation waiting to be accepted. Only users authorized to view AWS RAM invitations receive the alert.

See Also:

- Sharing Data Catalog tables and databases across AWS Accounts
- Cross-account data sharing in Lake Formation
- Accessing the underlying data of a shared table
- Metadata access control (for information about the named resource method versus the LF-TBAC method for sharing resources.)

Topics

- Accepting a resource share invitation from AWS RAM
- Viewing shared Data Catalog tables and databases

Accepting a resource share invitation from AWS RAM

If a Data Catalog resource is shared with your AWS account and your account is not in the same AWS organization as the sharing account, you do not have access to the shared resource until you accept a resource share invitation from AWS Resource Access Manager (AWS RAM). As a data lake administrator, you must first query AWS RAM for pending invitations and then accept the invitation.

You can use the AWS RAM console, API, or AWS Command Line Interface (AWS CLI) to view and accept invitations.

To view and accept a resource share invitation from AWS RAM (console)

1. Ensure that you have the required AWS Identity and Access Management (IAM) permissions to view and accept resource share invitations.

   For information about the suggested IAM policies for data lake administrators, see the section called “Data lake administrator permissions”.
2. Follow the instructions in Accepting and Rejecting Invitations in the *AWS RAM User Guide*.

To view and accept a resource share invitation from AWS RAM (AWS CLI)

1. Ensure that you have the required AWS Identity and Access Management (IAM) permissions to view and accept resource share invitations.

   For information about the suggested IAM policies for data lake administrators, see the section called “Data lake administrator permissions”.

2. Enter the following command to view pending resource share invitations.

   ```
   aws ram get-resource-share-invitations
   ```

   The output should be similar to the following.

   ```
   {
       "resourceShareInvitations": [
           {
               "resourceShareInvitationArn": "arn:aws:ram:us-
   east-1:111122223333:resource-share-invitation/a93aa60a-1bd9-46e8-96db-
   a4e72eec1d9f",
               "resourceShareName": "111122223333-123456789012-uswuU",
               "resourceShareArn": "arn:aws:ram:us-east-1:111122223333:resource-
   share/2a4ab5fb-d859-4751-84f7-8760b35fc1fe",
               "senderAccountId": "111122223333",
               "receiverAccountId": "123456789012",
               "invitationTimestamp": 1589576601.79,
               "status": "PENDING"
           }
       ]
   }
   ```

   Note the status of `PENDING`.

3. Copy the value of the `resourceShareInvitationArn` key to the clipboard.

4. Paste the value into the following command, replacing *<invitation-arn>*, and enter the command.

   ```
   aws ram accept-resource-share-invitation --resource-share-invitation-
   arn <invitation-arn>
   ```

   The output should be similar to the following.

```
{
    "resourceShareInvitations": [
        {
            "resourceShareInvitationArn": "arn:aws:ram:us-
east-1:111122223333:resource-share-invitation/a93aa60a-1bd9-46e8-96db-
a4e72eec1d9f",
            "resourceShareName": "111122223333-123456789012-uswuU",
            "resourceShareArn": "arn:aws:ram:us-east-1:111122223333:resource-
share/2a4ab5fb-d859-4751-84f7-8760b35fc1fe",
            "senderAccountId": "111122223333",
            "receiverAccountId": "123456789012",
            "invitationTimestamp": 1589576601.79,
            "status": "ACCEPTED"
        }
    ]
}
```

Note the status of ACCEPTED.

Viewing shared Data Catalog tables and databases

You can view resources that are shared with your account by using the Lake Formation console or AWS CLI. You can also use the AWS Resource Access Manager (AWS RAM) console or CLI to view both resources that are shared with your account and resources that you've shared with other AWS accounts.

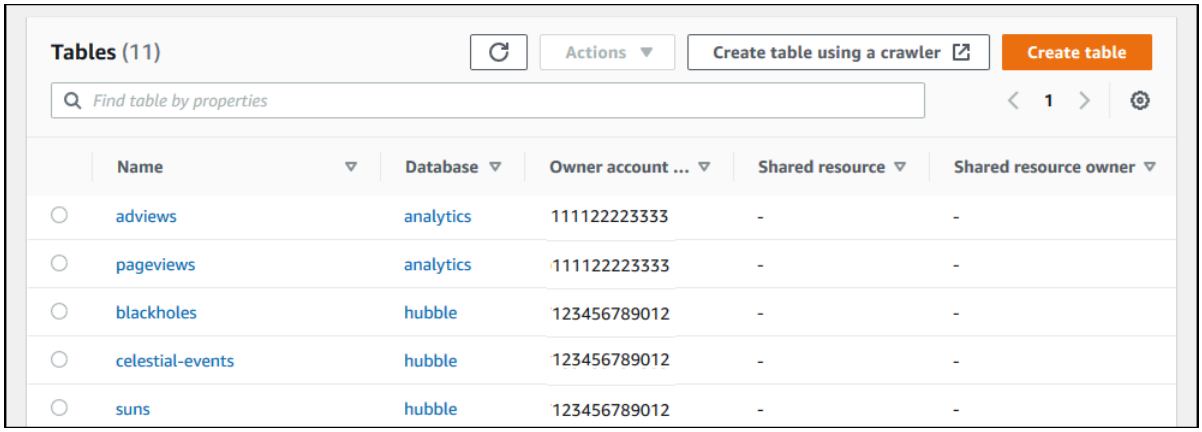
To view shared resources using the Lake Formation console

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.

   Sign in as a data lake administrator or a user who has been granted permissions on a shared table.
2. To view resources that are shared with your AWS account, do one of the following:

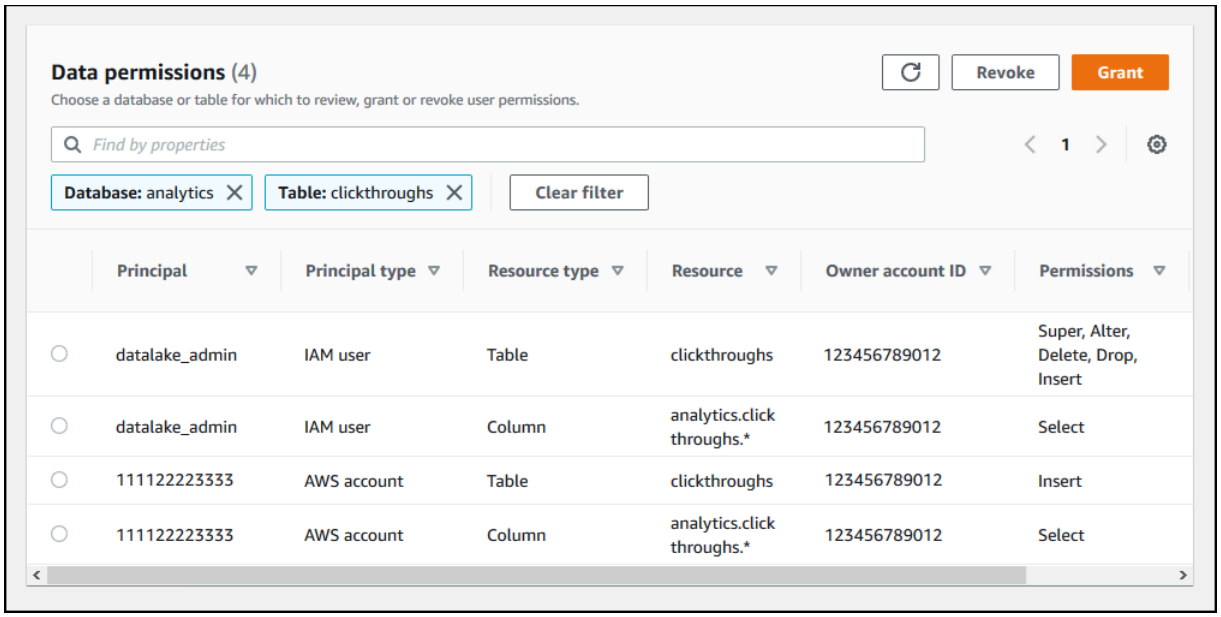
   - To view tables that are shared with your account, in the navigation pane, choose **Tables**.
   - To view databases that are shared with your account, in the navigation pane, choose **Databases**.

The console displays a list of databases or tables both in your account and shared with your account. For resources that are shared with your account, the console displays the owner's AWS account ID under the **Owner account ID** column (the third column in the following screenshot).



3. To view resources that you shared with other AWS accounts or organizations, in the navigation pane, choose **Data permissions**.

   Resources that you shared are listed on the **Data permissions** page with the external account number shown in the **Principal** column, as shown in the following image.



To view shared resources using the AWS RAM console

1. Ensure that you have the required AWS Identity and Access Management (IAM) permissions to view shared resources using AWS RAM.

At a minimum, you must have the `ram:ListResources` permission.This permission is included in the AWS managed policy `AWSLakeFormationCrossAccountManager`.

2. Sign in to the AWS Management Console and open the AWS RAM console at https://console.aws.amazon.com/ram/home.
3. Do one of the following:

   - To see resources that you shared, in the navigation pane, under **Shared by me**, choose **Shared resources**.
   - To see resources that are shared with you, in the navigation pane, under **Shared with me**, choose **Shared resources**.

Creating resource links

Resource links are Data Catalog objects that are links to metadata databases and tables—typically to shared databases and tables from other AWS accounts. They help to enable cross-account access to data in the data lake across all AWS Regions.

> **Note**
>
> Lake Formation supports querying Data Catalog tables across AWS Regions. You can access the Data Catalog databases and tables from any AWS Region by creating resource links in those regions that point to shared databases and tables in different Regions.

Topics

- How resource links work in Lake Formation
- Creating a resource link to a shared Data Catalog table
- Creating a resource link to a shared Data Catalog database
- Resource link handling in AWS Glue APIs

How resource links work in Lake Formation

A *resource link* is a Data Catalog object that is a link to a local or shared database or table. After you create a resource link to a database or table, you can use the resource link name wherever you

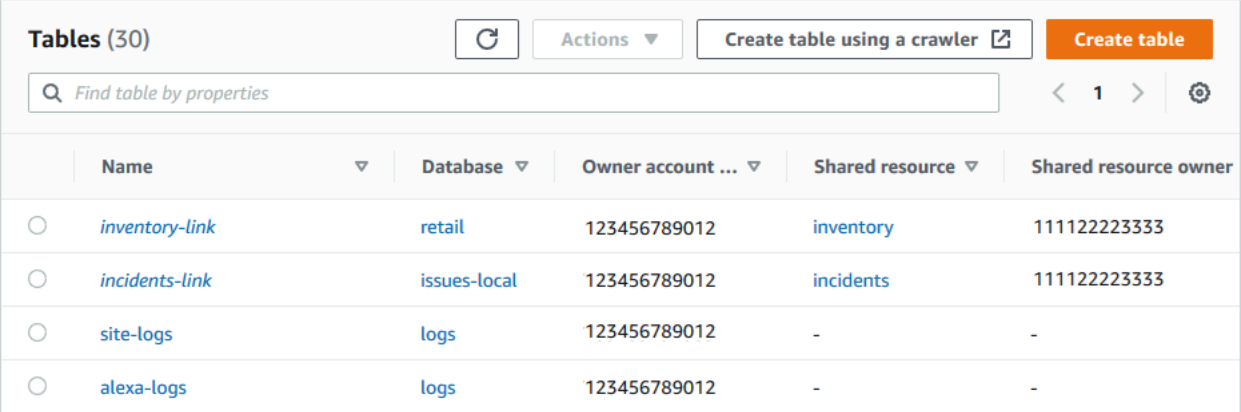
would use the database or table name. Along with tables that you own or tables that are shared with you, table resource links are returned by `glue:GetTables()` and appear as entries on the **Tables** page of the Lake Formation console. Resource links to databases act in a similar manner.

Creating a resource link to a database or table enables you to do the following:

- Assign a different name to a database or table in your Data Catalog. This is especially useful if different AWS accounts share databases or tables with the same name, or if multiple databases in your account have tables with the same name.
- Access the Data Catalog databases and tables from any AWS Region by creating resource links in those regions pointing to the database and tables in another region. You can run queries in any region with these resource links using Athena, Amazon EMR and run AWS Glue ETL Spark jobs, without copying source data nor the metadata in Glue Data Catalog.
- Use integrated AWS services such as Amazon Athena and Amazon Redshift Spectrum to run queries that access shared databases or tables. Some integrated services can't directly access databases or tables across accounts. However, they can access resource links in your account to databases and tables in other accounts.

> **Note**
>
> You don't need to create a resource link to reference a shared database or table in AWS Glue extract, transform, and load (ETL) scripts. However, to avoid ambiguity when multiple AWS accounts share a database or table with the same name, you can either create and use a resource link or specify the catalog ID when invoking ETL operations.

The following example shows the Lake Formation console **Tables** page, which lists two resource links. Resource link names are always displayed in italics. Each resource link is displayed along with the name and owner of its linked shared resource. In this example, a data lake administrator in AWS account 1111-2222-3333 shared the `inventory` and `incidents` tables with account 1234-5678-9012. A user in that account then created resource links to those shared tables.



The following are notes and restrictions on resource links:

- Resource links are required to enable integrated services such as Athena and Redshift Spectrum to query the underlying data of shared tables. Queries in these integrated services are constructed against the resource link names.
- Assuming that the setting **Use only IAM access control for new tables in this database** is turned off for the containing database, only the principal who created a resource link can view and access it. To enable other principals in your account to access a resource link, grant the `DESCRIBE` permission on it. To enable others to drop a resource link, grant the `DROP` permission on it. Data lake administrators can access all resource links in the account. To drop a resource link created by another principal, the data lake administrator must first grant themselves the `DROP` permission on the resource link. For more information, see Lake Formation permissions reference.

  > **Important**
  >
  > Granting permissions on a resource link doesn't grant permissions on the target (linked) database or table. You must grant permissions on the target separately.

- To create a resource link, you need the Lake Formation `CREATE_TABLE` or `CREATE_DATABASE` permission, as well as the `glue:CreateTable` or `glue:CreateDatabase` AWS Identity and Access Management (IAM) permission.
- You can create resource links to local (owned) Data Catalog resources, as well as to resources shared with your AWS account.
- When you create a resource link, no check is performed to see if the target shared resource exists or whether you have cross-account permissions on the resource. This enables you to create the resource link and shared resource in any order.

- If you delete a resource link, the linked shared resource is not dropped. If you drop a shared resource, resource links to that resource are not deleted.
- It's possible to create resource link chains. However, there is no value in doing so, because the APIs follow only the first resource link.

See also:

- Granting permissions on Data Catalog resources

Creating a resource link to a shared Data Catalog table

You can create a resource link to a shared table in any AWS Region by using the AWS Lake Formation console, API, or AWS Command Line Interface (AWS CLI).

To create a resource link to a shared table (console)

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as a principal who has the Lake Formation CREATE_TABLE permission on the database to contain the resource link.
2. In the navigation pane, choose **Tables** under Data Catalog, and then choose **Create**, **Resource link**.
3. On the **Create resource link** page, provide the following information:

   Resource link name

   Enter a name that adheres to the same rules as a table name. The name can be the same as the target shared table.

   Database

   The database in the local Data Catalog to contain the resource link.

   Shared table owner Region

   If you are creating the resource link in a different Region, select the region of the target shared table.

Shared table

Select a shared table from the list, or enter a local (owned) or shared table name.

The list contains all the tables shared with your account. Note the database and owner account ID that are listed with each table. If you don't see a table that you know was shared with your account, check the following:

- If you aren't a data lake administrator, check that the data lake administrator granted you Lake Formation permissions on the table.
- If you are a data lake administrator, and your account is not in the same AWS organization as the granting account, ensure that you have accepted the AWS Resource Access Manager (AWS RAM) resource share invitation for the table. For more information, see Accepting a resource share invitation from AWS RAM.

Shared table's database

If you selected a shared table from the list, this field is populated with the shared table's database in the external account. Otherwise, enter a local database (for a resource link to a local table) or the shared table's database in the external account.

Shared table owner

If you selected a shared table from the list, this field is populated with the shared table's owner account ID. Otherwise, enter your AWS account ID (for a resource link to a local table) or the ID of the AWS account that shared the table.

4. Choose **Create** to create the resource link.

   You can then view the resource link name under the **Name** column on the **Tables** page.

5. (Optional) Grant the Lake Formation `DESCRIBE` permission on the resource link to principals that must be able to view the link and access the target table.

   However, granting permissions on a resource link doesn't grant permissions on the target (linked) database or table. You must grant permissions on the target database separately for the table/resource link to be visible in Athena.

To create a resource link to a shared table in the same Region (AWS CLI)

1. Enter a command similar to the following.

```
aws glue create-table --database-name myissues --table-input
 '{"Name":"my_customers","TargetTable":
{"CatalogId":"111122223333","DatabaseName":"issues","Name":"customers"}}'
```

This command creates a resource link named `my_customers` to the shared table `customers`, which is in the database `issues` in the AWS account 1111-2222-3333. The resource link is stored in the local database `myissues`.

2. (Optional) Grant the Lake Formation `DESCRIBE` permission on the resource link to principals that must be able to view the link and access the target table.

   However, granting permissions on a resource link doesn't grant permissions on the target (linked) table. You must grant permissions on the target database separately for the table/resource link to be visible in Athena.

To create a resource link to a shared table in a different Region (AWS CLI)

1. Enter a command similar to the following.

```
aws glue create-table --region eu-west-1 --cli-input-json '{
    "CatalogId": "111122223333",
    "DatabaseName": "ireland_db",
    "TableInput": {
        "Name": "rl_useast1salestb_ireland",
        "TargetTable": {
            "CatalogId": "444455556666",
            "DatabaseName": "useast1_salesdb",
            "Region": "us-east-1",
            "Name":"useast1_salestb"
        }
    }
}‘
```

This command creates a resource link named `rl_useast1salestb_ireland` in the Europe (Ireland) Region to the shared table `useast1_salestb`, which is in the database `useast1_salesdb` in the AWS account 444455556666 in the US East (N. Virginia) Region. The resource link is stored in the local database `ireland_db`.

2. Grant the Lake Formation `DESCRIBE` permission to principals that must be able to view the link and access the link target through the link.

However, granting permissions on a resource link doesn't grant permissions on the target (linked) table. You must grant permissions on the target table separately for the table/resource link to be visible in Athena.

See also:

- How resource links work in Lake Formation
- DESCRIBE

Creating a resource link to a shared Data Catalog database

You can create a resource link to a shared database by using the AWS Lake Formation console, API, or AWS Command Line Interface (AWS CLI).

To create a resource link to a shared database (console)

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/. Sign in as a data lake administrator or as a database creator.

   A database creator is a principal who has been granted the Lake Formation CREATE_DATABASE permission.
2. In the navigation pane, choose **Databases**, and then choose **Create**, **Resource link**.
3. On the **Create resource link** page, provide the following information:

   Resource link name

   Enter a name that adheres to the same rules as a database name. The name can be the same as the target shared database.

   Destination catalog

   Select the destination catalog for the database resource link.

   Shared database owner Region

   If you are creating the resource link in a different Region, select the Region of the target shared database.

Shared database

Choose a database from the list, or enter a local (owned) or shared database name.

The list contains all the databases shared with your account. Note the owner account ID that is listed with each database. If you don't see a database that you know was shared with your account, check the following:

- If you aren't a data lake administrator, check that the data lake administrator granted you Lake Formation permissions on the database.
- If you are a data lake administrator, and your account is not in the same AWS organization as the granting account, ensure that you have accepted the AWS Resource Access Manager (AWS RAM) resource share invitation for the database. For more information, see Accepting a resource share invitation from AWS RAM.

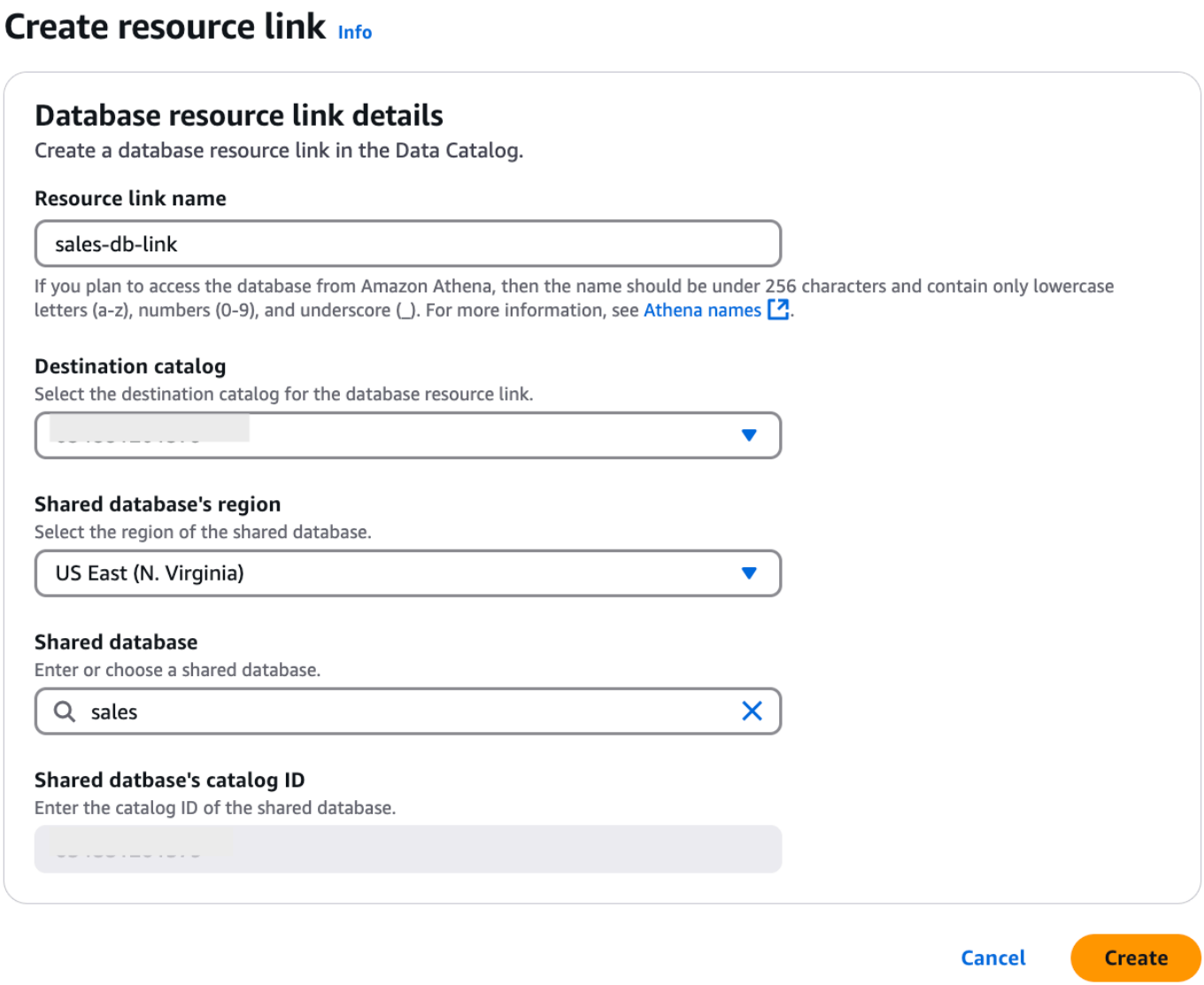
Shared database owner

If you selected a shared database from the list, this field is populated with the shared database's owner account ID. Otherwise, enter your AWS account ID (for a resource link to a local database) or the ID of the AWS account that shared the database.

Shared database's catalog ID

Enter the catalog ID for the shared database. When creating a resource link to a databse that's shared from another AWS account, you need to specify this catalog ID to identify which account's Data Catalog contains the source database.

When you select a shared database from the dropdown menu, the system automatically fills in the catalog ID of the account that owns and has shared that database with you.



4. Choose **Create** to create the resource link.

   You can then view the resource link name under the **Name** column on the **Databases** page.

5. (Optional) Grant the Lake Formation `DESCRIBE` permission on the resource link to principals from the Europe (Ireland) Region that must be able to view the link and access the target database.

   However, granting permissions on a resource link doesn't grant permissions on the target (linked) database or table. You must grant permissions on the target database separately for the table/resource link to be visible in Athena.

To create a resource link to a shared database in the same Region(AWS CLI)

1. Enter a command similar to the following.

```
aws glue create-database --database-input '{"Name":"myissues","TargetDatabase":
{"CatalogId":"111122223333","DatabaseName":"issues"}}'
```

   This command creates a resource link named `myissues` to the shared database `issues`, which is in the AWS account 1111-2222-3333.

2. (Optional) Grant the Lake Formation `DESCRIBE` permission to principals on the resource link that must be able to view the link and access the target database or table.

   However, granting permissions on a resource link doesn't grant permissions on the target (linked) database or table. You must grant permissions on the target database separately for the table/resource link to be visible in Athena.

To create a resource link to a shared database in a different Region(AWS CLI)

1. Enter a command similar to the following.

```
aws glue create-database --region eu-west-1 --cli-input-json '{
    "CatalogId": "111122223333",
    "DatabaseInput": {
      "Name": "rl_useast1shared_irelanddb",
      "TargetDatabase": {
          "CatalogId": "444455556666",
          "DatabaseName": "useast1shared_db",
          "Region": "us-east-1"
       }
    }
}'
```

   This command creates a resource link named `rl_useast1shared_irelanddb` in the AWS account 111122223333 in the Europe (Ireland) Region to the shared database `useast1shared_db`, which is in the AWS account 444455556666 in the US East (N. Virginia) Region.

2. Grant the Lake Formation `DESCRIBE` permission to principals from the Europe (Ireland) Region that must be able to view the link and access the link target through the link.

See also:

- How resource links work in Lake Formation
- DESCRIBE

Resource link handling in AWS Glue APIs

The following tables explain how the AWS Glue Data Catalog APIs handle database and table resource links. For all `Get*` API operations, only databases and tables that the caller has permissions on get returned. Also, when accessing a target database or table through a resource link, you must have both AWS Identity and Access Management (IAM) and Lake Formation permissions on both the target and the resource link. The Lake Formation permission that is required on resource links is `DESCRIBE`. For more information, see `DESCRIBE`.

Database API operations

API operation	Resource link handling
`CreateDatabase`	If the database is a resource link, creates the resource link to the designated target database.
`UpdateDatabase`	If the designated database is a resource link, follows the link and updates the target database. If the resource link must be modified to link to a different database, you must delete it and create a new one.
`DeleteDatabase`	Deletes the resource link. It doesn't delete the linked (target) database.
`GetDatabase`	If the caller has permissions on the target, follows the link to return the target's properties. Otherwise, it returns the properties of the link.
`GetDatabases`	Returns a list of databases, including resource links. For each resource link in the result set, the operation follows the link to get the properties of the link target. You must specify `ResourceShareType = ALL` to see the databases shared with your account.

Table API operations

API operation	Resource link handling
`CreateTable`	If the database is a resource link, follows the database link and creates a table in the target database. If the table is a resource link, the operation creates the resource link in the designate d database. Creating a table resource link through a database resource link is not supported.
`UpdateTable`	If either the table or designated database is a resource link, updates the target table. If both the table and database are resource links, the operation fails.
`DeleteTable`	If the designated database is a resource link, follows the link and deletes the table or table resource link in the target database. If the table is a resource link, the operation deletes the table resource link in the designated database. Deleting a table resource link does not delete the target table.
`BatchDeleteTable`	Same as `DeleteTable` .
`GetTable`	If the designated database is a resource link, follows the database link and returns the table or table resource link from the target database. Otherwise, if the table is a resource link, the operation follows the link and returns the target table properties.
`GetTables`	If the designated database is a resource link, follows the database link and returns the tables and table resource links from the target database. If the target database is a shared database from another AWS account, the operation returns only the shared tables in that database. It doesn't follow the table resource links in the target database. Otherwise, if the designated database is a local (owned) database, the operation returns all the tables in the local database, and follows each table resource link to return target table properties.

API operation	Resource link handling
SearchTables	Returns tables and table resource links. It doesn't follow links to return target table properties. You must specify ResourceShareType = ALL to see tables shared with your account.
GetTableVersion	Same as GetTable.
GetTableVersions	Same as GetTable.
DeleteTableVersion	Same as DeleteTable .
BatchDeleteTableVersion	Same as DeleteTable .

Partition API operations

API operation	Resource link handling
CreatePartition	If the designated database is a resource link, follows the database link and creates a partition in the designated table in the target database. If the table is a resource link, the operation follows the resource link and creates the partition in the target table. Creating a partition through both a table resource link and database resource link is not supported.
BatchCreatePartition	Same as CreatePartition .
UpdatePartition	If the designated database is a resource link, follows the database link and updates the partition in the designated table in the target database. If the table is a resource link, the operation follows the resource link and updates the partition in the target table. Updating a partition through both a table resource link and database resource link is not supported.
DeletePartition	If the designated database is a resource link, follows the database link and deletes the partition in the designated table in the target database. If the table is a resource link, the operation follows the

API operation	Resource link handling
	resource link and deletes the partition in the target table. Deleting a partition through both a table resource link and database resource link is not supported.
`BatchDeletePartition`	Same as `DeletePartition` .
`GetPartition`	If the designated database is a resource link, follows the database link and returns partition information from the designated table. Otherwise, if the table is a resource link, the operation follows the link and returns partition information. If both the table and database are resource links, it returns an empty result set.
`GetPartitions`	If the designated database is a resource link, follows the database link and returns partition information for all partitions in the designated table. Otherwise, if the table is a resource link, the operation follows the link and returns partition information. If both the table and database are resource links, it returns an empty result set.
`BatchGetPartition`	Same as `GetPartition` .

User-defined functions API operations

API operation	Resource Link Handling
(All API operations)	If the database is a resource link, follows the resource link and performs the operation on the target database.

See also:

- How resource links work in Lake Formation

Accessing tables across Regions

Lake Formation supports querying Data Catalog tables across AWS Regions. You can access data in a Region from other Regions using Amazon Athena, Amazon EMR, and AWS Glue ETL by creating resource links in other Regions pointing to the source databases and tables. With cross-Region table access, you can access data across Regions without copying the underlying data or the metadata into the Data Catalog.

For example, you can share a database or table in a producer account to a consumer account in Region A. After accepting the resource share invitation in Region A, the data lake administrator of the consumer account can create resource links to the shared resource in Region B. The consumer account administrator can grant permissions on the shared resource to the IAM principals in that account in Region A and can grant resource link permissions in Region B. Using the resource link, the principals in the consumer account can query the shared data from Region B.

You can also host the Amazon S3 data source in Region A in a producer account, and register the data location in a central account in Region B. You can create Data Catalog resources in the central account, set up Lake Formation permissions, and share data with consumers in your account or with external accounts in Region B. The cross-Region feature allows users to access these Data Catalog tables from Region C using resource links.

Using this feature, you can query federated databases in Apache Hive Metastores across Regions, and also join tables in the local Region with tables in another Region when running queries.

Lake Formation supports the following features with cross-Region table access:

- LF-Tag based access control
- Fine-grained access control permissions
- Write operations on the shared database or table with appropriate permissions
- Cross-account data sharing at account-level and direct with IAM principals-level

Non-administrative users with `Create_Database` and `Create_Table` permissions can create cross-Region resource links.

> **Note**
>
> You can create cross-Region resource links in any Region and access data without applying Lake Formation permissions. For source data in Amazon S3 that isn't registered with Lake

Formation, access is determined by IAM permissions policies for Amazon S3 and AWS Glue actions.

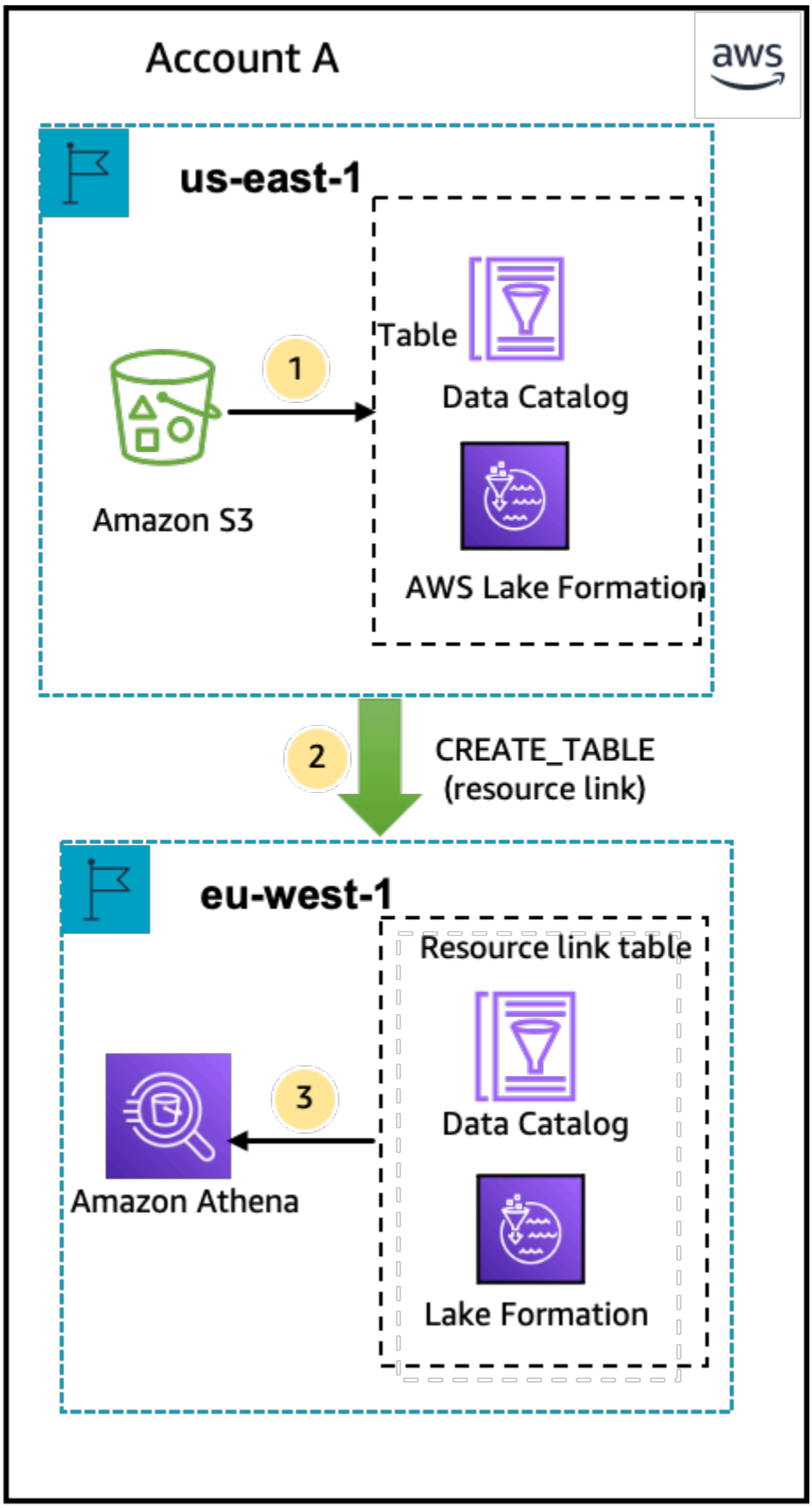
For limitations, see Cross-Region data access limitations.

Workflows

The following diagrams show the workflows for accessing data across AWS Regions from the same AWS account and from an external account.

Workflow for accessing tables shared within the same AWS account

In the diagram below, the data is shared with a user in the same AWS account in the US East (N. Virginia) Region, and the user queries the shared data from the Europe (Ireland) Region.


Account A
aws
us-east-1
Table
Data Catalog
1
Amazon S3
AWS Lake Formation
2
CREATE_TABLE
(resource link)
eu-west-1
Resource link table
Data Catalog
3
Amazon Athena
Lake Formation

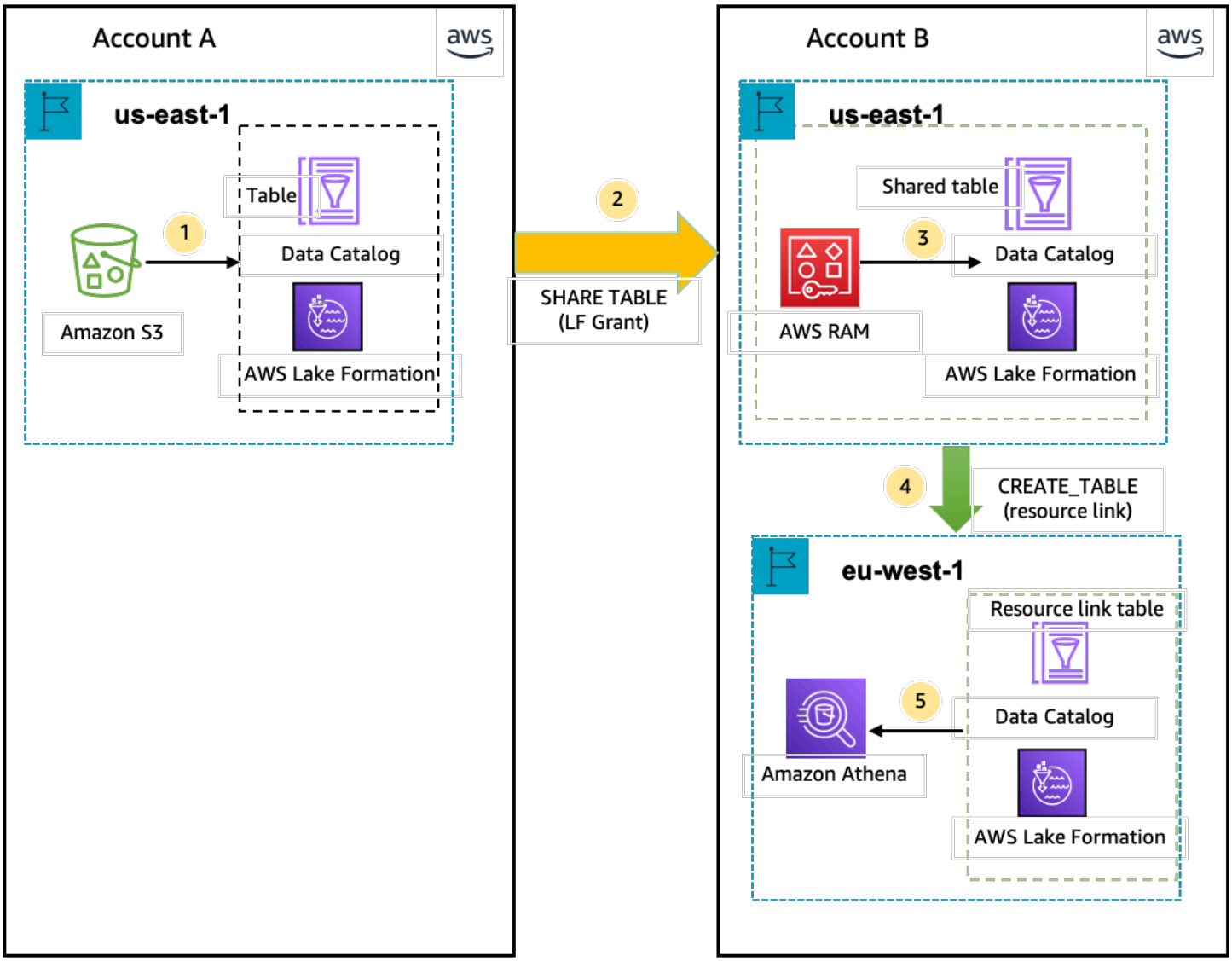
The data lake administrator performs the following activities (steps 1-2):

1. A data lake administrator sets up an AWS account with the Data Catalog databases and tables and registers an Amazon S3 data location with Lake Formation in the US East (N. Virginia) Region.

   Grants `Select` permission on a Data Catalog resource (product table in the diagram) to a principal (user) in the same account.
2. Creates a resource link in the Europe (Ireland) Region pointing to the source table in the US East (N. Virginia) Region. Grants `DESCRIBE` permission on the resource link from the Europe (Ireland) Region to the principal.
3. The user queries the table from the Europe (Ireland)Region using Athena.

Workflow for accessing tables shared with an external AWS account

In the diagram below, the producer account (Account A) hosts the Amazon S3 bucket, registers the data location, and shares a Data Catalog table with a consumer account (Account B) in the US East (N. Virginia) Region and a user from the consumer account (Account B) queries the table from the Europe (Ireland) Region.



1. A data lake administrator sets up an AWS account (producer account) with the Data Catalog resources and an Amazon S3 data location registered with Lake Formation in the US East (N. Virginia) Region.
2. The data lake administrator of the producer account shares a Data Catalog table to a consumer account.
3. The data lake administrator of the consumer account accepts the data share invitation in the US East (N. Virginia) Region and Grants `Select` permission on the shared table to a principal from the same Region.
4. The data lake administrator of the consumer account creates a resource link in the Europe (Ireland) Region pointing to the target shared table in the US East (N. Virginia) Region and grants the user `DESCRIBE` permission on the resource link from Europe (Ireland) Region.

5. The user queries the data from the Europe (Ireland) Region using Athena.

Setting up cross-Region table access

To access data from a different Region, you need to first set up the Data Catalog databases and tables in the Region where you register your Amazon S3 data location. You can share the Data Catalog databases and tables with principals in your account or in another account. Then, you need to create data lake administrators who can create resource links pointing to the target shared data location in the Regions where users query the data.

To query data shared within the same account from a different Region

In this section, the target shared table Region is referred to as Region A and users run queries from Region B.

1. **Account setup in Region A (where you create and share the data)**

   A data lake administrator needs to complete the following actions:

   a. Register an Amazon S3 data location.

      For more information, see Adding an Amazon S3 location to your data lake.

   b. Create databases and tables in the account. This can also be done by a non-administrative user who has permissions to create databases and tables.

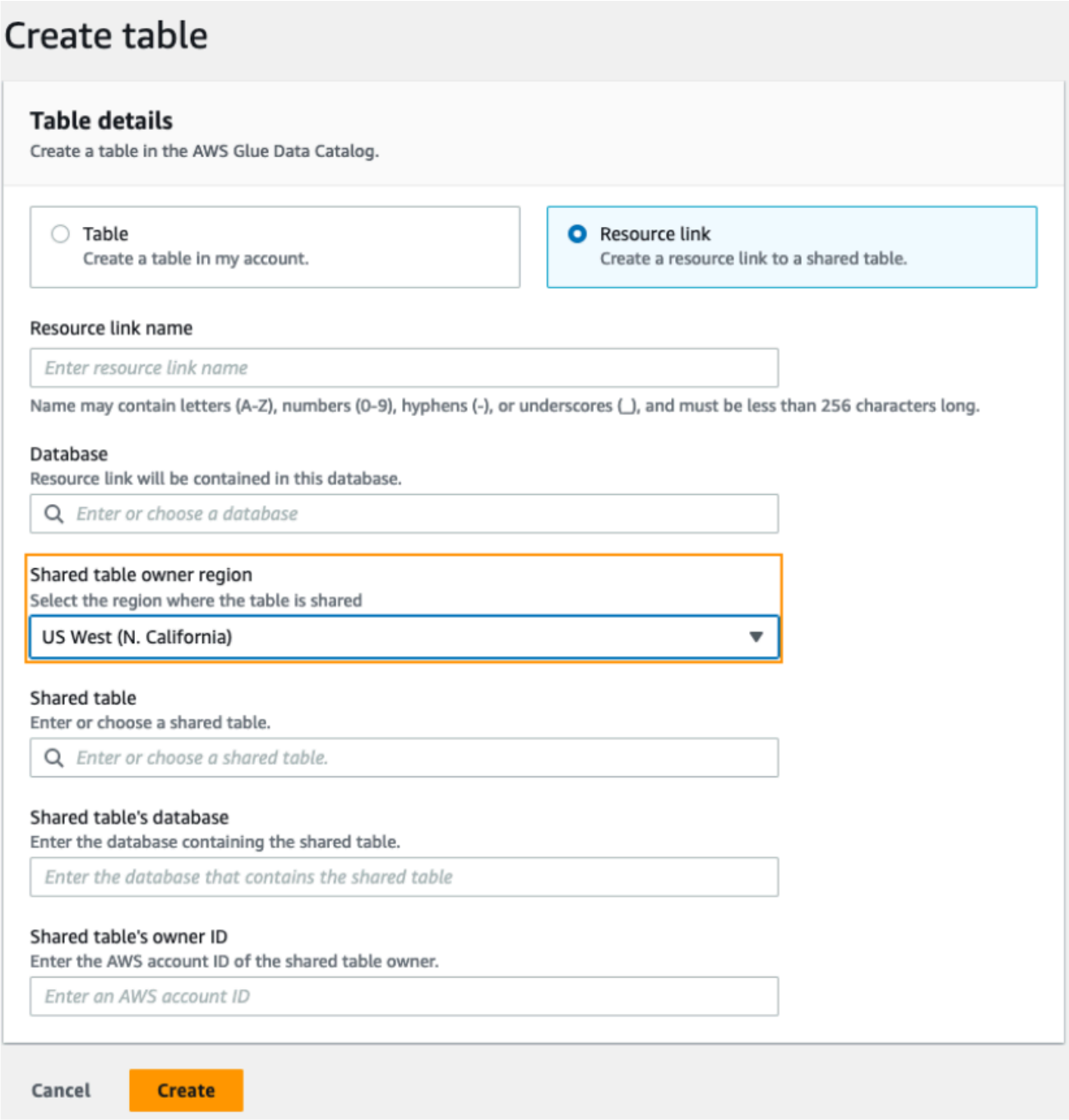
   c. Grant data permissions on a table to the principals with `Grantable permissions`.

      For more information see, Granting permissions on Data Catalog resources.

2. **Account setup in Region B (where you access the data)**

   A data lake administrator needs to complete the following actions:

   a. Create a resource link in Region B pointing to the target shared table in Region A. Specify the **Shared table owner Region** on the **Create table** screen.



For instructions on creating resource links to databases and tables, see Creating resource links.

b. Grant `Describe` permission to IAM principals on the resource link in Region B.

For more information on granting permissions on resource links, see Granting resource link permissions.

IAM principals in Region B can query the target table through the link using Athena.

To access cross-account data from a different Region

1. **Producer/grantor account setup**

   A data lake administrator needs to complete the following actions:

   a. Set up the producer/grantor account in Region A.
   b. Register an Amazon S3 data location in Region A.
   c. Create databases and tables. This can be done by a non-administrative user who has permissions to create tables.
   d. Grant data permissions to the consumer/grantee account on a table in Region A with `Grantable permissions`.

      For more information, see Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts.

2. **Consumer/grantee account setup**

   A data lake administrator needs to complete the following actions:

   a. Accept the resource share invitation from AWS RAM in Region A.
   b. Create a resource link in Region B pointing to the shared table. Region B is where users will want to query the table.
   c. Grant data permissions on the shared table to IAM principals in Region A.

      > **Note**
      >
      > You must grant permissions to the shared table in the the same Region where the table was shared.

   d. Grant permissions to principals on the resource link in Region B.

      Principals in the consumer account in Region B then query the shared table from Region B using Athena.

Security in AWS Lake Formation

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to AWS Lake Formation, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Lake Formation. The following topics show you how to configure Lake Formation to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Lake Formation resources.

Topics

- Data Protection in Lake Formation
- Infrastructure Security in AWS Lake Formation
- Cross-service confused deputy prevention
- Security event logging in AWS Lake Formation

Data Protection in Lake Formation

The AWS shared responsibility model applies to data protection in . As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that

you use. For more information about data privacy, see Data Privacy FAQ. For information about data protection in Europe, see the General Data Protection Regulation (GDPR) Center.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at Rest

AWS Lake Formation supports data encryption in the following areas:

- Data in your Amazon Simple Storage Service (Amazon S3) data lake.

  Lake Formation supports data encryption with AWS Key Management Service (AWS KMS). Data is typically written to the data lake by means of AWS Glue extract, transform, and load (ETL) jobs. For information about how to encrypt data written by AWS Glue jobs, see Encrypting Data Written by Crawlers, Jobs, and Development Endpoints in the *AWS Glue Developer Guide*.

- The AWS Glue Data Catalog, which is where Lake Formation stores metadata tables that describe data in the data lake.

  For more information, see Encrypting Your Data Catalog in the *AWS Glue Developer Guide*.

To add an Amazon S3 location as storage in your data lake, you *register* the location with AWS Lake Formation. You can then use Lake Formation permissions for fine-grained access control to AWS Glue Data Catalog objects that point to this location, and to the underlying data in the location.

Lake Formation supports registering an Amazon S3 location that contains encrypted data. For more information, see Registering an encrypted Amazon S3 location.

Infrastructure Security in AWS Lake Formation

As a managed service, AWS Lake Formation is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Lake Formation through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the aws:SourceArn and aws:SourceAccount global condition context keys in resource policies to limit the permissions that AWS Lake Formation gives another service to the resource. If you use both global condition context keys, the aws:SourceAccount value and the account in the aws:SourceArn value must use the same account ID when used in the same policy statement.

Currently, Lake Formation only supports aws:SourceArn in the following format:

```
arn:aws:lakeformation:aws-region:account-id:*
```

The following example shows how you can use the aws:SourceArn and aws:SourceAccount global condition context keys in Lake Formation to prevent the confused deputy problem.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ConfusedDeputyPreventionExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "lakeformation.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:lakeformation:aws-region:123456789012:*"
        }
      }
    }
  ]
}
```

Security event logging in AWS Lake Formation

AWS Lake Formation is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Lake Formation. CloudTrail captures all API calls for Lake Formation as events. The calls captured include calls from the Lake Formation console, the AWS Command Line Interface, and code calls to the Lake Formation API operations.

For more information about event logging in Lake Formation, see Logging AWS Lake Formation API calls using AWS CloudTrail.

Note

`GetTableObjects`, `UpdateTableObjects`, and `GetWorkUnitResults` are high-volume data plane operations. Calls to these APIs are not currently logged to CloudTrail. For more information about data plane operations in CloudTrail, see Logging data events for trails in the *AWS CloudTrail User Guide*.
Changes in Lake Formation to support additional CloudTrail events will be documented at Document history for AWS Lake Formation.

Integrating third-party services with Lake Formation

Integrating with AWS Lake Formation enables third-party services to securely access data in their Amazon S3 based data lakes. You can use Lake Formation as your authorization engine to manage or enforce permissions to your data lake with integrated AWS services such as AWS Glue ETL, Amazon Athena, Amazon EMR, and Redshift Spectrum. Lake Formation provides two options for integrating services:

1. The Lake Formation application integration settings: Lake Formation can vend scoped-down temporary credentials in the form of AWS STS tokens to registered Amazon S3 locations based on the effective permissions, so that authorized applications can access data on behalf of users.
2. Central enforcement: Lake Formation querying API operations retrieve data from Amazon S3 and filter the results based on effective permissions. The engine or application integrating with the querying API operation can depend on Lake Formation to evaluate the calling identity's permissions and securely filter the data based on these permissions. Third-party query engines only see and operate on filtered data.

Lake Formation credential vending doesn't integrate with spark sql queries. Credential vending only works with queries that run through the AWS Glue ETL library.

Topics

- Using Lake Formation application integration

Using Lake Formation application integration

Lake Formation allows third-party services to integrate with Lake Formation and get temporary access to Amazon S3 data on behalf of their users by using GetTemporaryGlueTableCredentials and GetTemporaryGluePartitionCredentials operations. This allows third-party services to use the same authorization and credential vending feature that the rest of AWS analytics services use. This section describes how to use these API operations to integrate a third-party query engine with Lake Formation.

These API operations are disabled by default. There are two options to authorize Lake Formation to integrate applications:

- Configure IAM session tags that are validated every time the application integration API operations are called

  For more information, see Enabling permissions for a third-party query engine to call application integration API operations.

- Enable the option that **Allows external engines to access data in Amazon S3 locations with full table access**

  This option allows query engines and applications to get credentials without IAM session tags if the user has full table access. It provides query engines and applications performance benefits as well as simplifies data access. Amazon EMR on Amazon EC2 is able to leverage this setting.

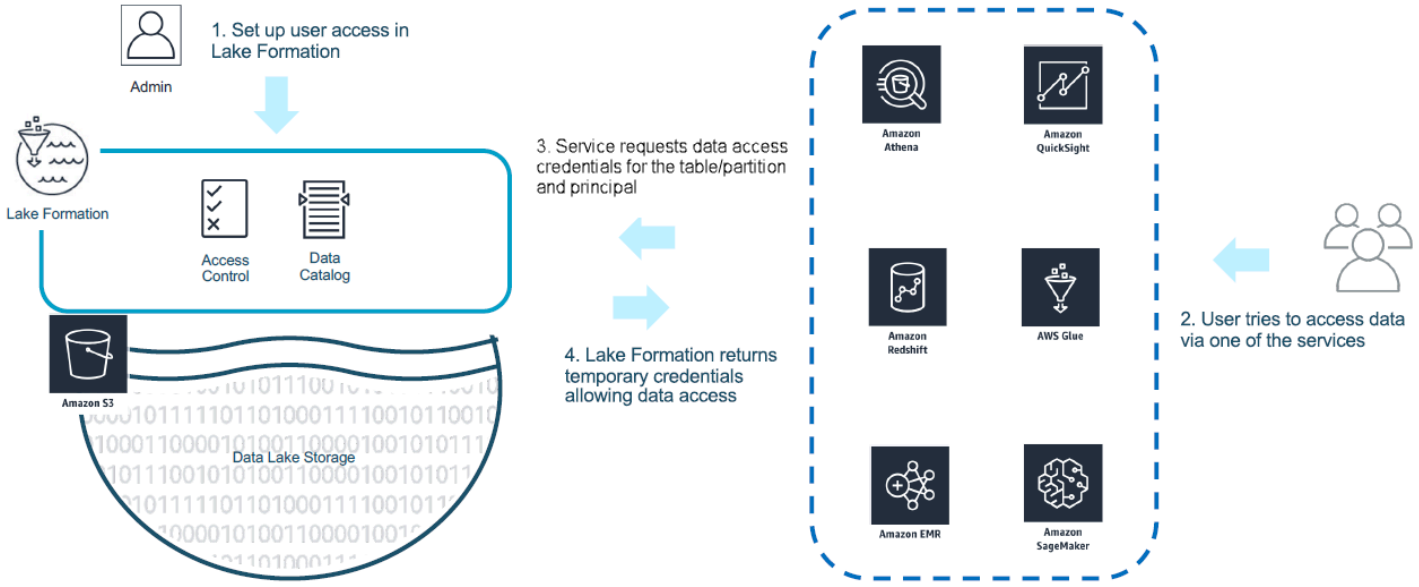
  For more information, see Application integration for full table access.

Topics

- How Lake Formation application integration works
- Roles and responsibilities in Lake Formation application integration
- Lake Formation workflow for application integration API operations
- Registering a third-party query engine
- Enabling permissions for a third-party query engine to call application integration API operations
- Application integration for full table access

How Lake Formation application integration works

This section describes how to use application integration API operations to integrate a third-party application (query engine) with Lake Formation.



1. The Lake Formation administrator performs the following activities:
   - Registers an Amazon S3 location with Lake Formation by providing an IAM role (used for vending credentials) that has appropriate permissions to access data within the Amazon S3 location
   - Registers a third-party application to be able to call Lake Formation's credential vending API operations. See the section called “Registering a third-party query engine”
   - Grants permissions for users to access databases and tables

     For example, if you want to publish a user sessions data set that includes some columns containing personally identifiable information (PII), to restrict access, you assign these columns an LF-TBAC tag named “classification” with a value of “sensitive”. Next, you define a permission that allows a business analyst to access the user sessions data, but exclude those columns tagged with *classification = sensitive*.
2. A principal (user) submits a query to an integrated service.
3. The integrated application sends the request to Lake Formation asking for table information and credentials to access the table.
4. If the querying principal is authorized to access the table, Lake Formation returns the credentials to the integrated application, which allows data access.

> **Note**
>
> Lake Formation doesn't access the underlying data when vending credentials.

5. The integrated service reads data from Amazon S3, filters columns based on the policies it received, and returns the results back to the principal.

> **Important**
>
> Lake Formation credential vending API operations enable a **distributed-enforcement with explicit deny on failure (fail-close) model.** This introduces a three-party security model between customers, third-party services and Lake Formation. Integrated services are trusted to properly enforce Lake Formation permissions (distributed-enforcement).

The integrated service is responsible for filtering the data read from Amazon S3 based on the policies returned from Lake Formation before the filtered data is returned back to the user. Integrated services follow a fail-close model, which means that they must fail the query if they are unable to enforce required Lake Formation permissions.

Roles and responsibilities in Lake Formation application integration

The following are the roles and their associated responsibilities for enabling third-party application integration with AWS Lake Formation.

Role	Responsibility
The customer	• Enable Lake Formation application integration setting (see the section called “Registering a third-party query engine”). • Explicitly registers approved third parties with Lake Formation (see the section called “Registering a third-party query engine”). • Tests and validates third-party solutions with Lake Formation permissions. • Monitors and audits third-party usage of Lake Formation credential vending API operations.

Role	Responsibility
The third-party	• Publicly documents the supported capability for every software revision and provides instructions to enable it correctly. • Accurately advertises the supported capabilities when calling Lake Formation credential vending API operations (according to the documentation). • Securely stores and handles vended credentials to avoid credential leaks and privilege escalation. • Enforces permissions based on supported capabilities and returns only filtered data to users • Fails the query when unable to properly enforce required permissions
`AWS Lake Formation`	• Correctly derives and returns effective permissions for a given principal. • Validates third-party supported capabilities on an API operation call-by-call basis. • Returns scoped-down IAM credentials only when the engine's advertised capabilities match those defined on the catalog resources, otherwise returns an error.

Lake Formation workflow for application integration API operations

The following is the work flow for application integration API operations:

1. A user submits a query or request for data using an integrated third-party query engine. The query engine assumes an IAM role that represents the user or a group of users, and retrieves trusted credentials to be used when calling the application integration API operations.
2. The query engine calls `GetUnfilteredTableMetadata`, and if it is a partitioned table, the query engine calls `GetUnfilteredPartitionsMetadata` to retrieve metadata and policy information from the Data Catalog.
3. Lake Formation performs authorization for the request. If the user doesn't have appropriate permissions on the table, then *AccessDeniedException* is thrown.
4. As part of the request, the query engine sends the filtering it supports. There are two flags that can be sent within an array: *COLUMN_PERMISSIONS* and *CELL_FILTER_PERMISSION*. If the query

engine doesn't support any of these features, and a policy exists on the table for the feature, then a *PermissionTypeMismatchException* is thrown and the query fails. This is to avoid data leakage.

5. The returned response contains the following:
   - The entire schema for the table so that query engines can use it to parse the data from storage.
   - A list of authorized columns that the user has access. If the authorized column list is empty, it indicates that the user has `DESCRIBE` permissions, but does not have `SELECT` permissions, and the query fails.
   - A flag, `IsRegisteredWithLakeFormation`, which indicates if Lake Formation can vend credentials to this resources data. If this returns false, then the customers' credentials should be used to access Amazon S3.
   - A list of `CellFilters` if any that should be applied to rows of data. This list contains columns and an expression to evaluate each row. This should only be populated if *CELL_FILTER_PERMISSION* is sent as part of the request and there is a data filter against the table for the calling user.
6. After the metadata is retrieved, the query engine calls `GetTemporaryGlueTableCredentials` or `GetTemporaryGluePartitionCredentials` to get AWS credentials to retrieve data from the Amazon S3 location.
7. The query engine reads relevant objects from Amazon S3, filters the data based on the policies it received in step 2, and returns the results to the user.

The application integration API operations for Lake Formation contain additional content for configuring integration with third-party query engines. You can see the operation details in the Credential vending API operations section.

The `QuerySessionContext` is a structure that query engines can additionally send to Lake Formation for these application integration API operations. It allows Lake Formation to store and utilize additional context for a given query. The following provides an example of how QuerySessionContext should be used:

1. The query engine makes a `GetInternalUnfilteredMetadata` call, passing in a QSC structure containing a unique query id in the request:

```
{
    "QuerySessionContext": {
```

```
        "QueryId": "your-unique-identifier-here"
    }
}
```

2. The `GetInternalUnfilteredMetadata` call will have returned a `QueryAuthorizationId` string in the response. On the next (and any subsequent) query call that accepts a QSC structure in the input, the query engine passes the same QSC structure that now also contains the `QueryAuthorizationId` returned by Lake Formation. Suppose this next call is `GetTemporaryGlueTableCredentials`; the request will contain:

```
{
    "QuerySessionContext": {
        "QueryAuthorizationId": "lf-returned-query-authz-id-here",
        "QueryId": "your-unique-identifier-here"
    },
}
```

Registering a third-party query engine

Before a third-party query engine can use the application integration API operations, you need to explicitly enable permissions for the query engine to call the API operations on your behalf. This is done in a few steps:

1. You need to specify the AWS accounts and IAM session tags that require permission to call the application integration API operations through the AWS Lake Formation console, the AWS CLI or the API/SDK.
2. When the third-party query engine assumes the execution role in your account, the query engine must attach a session tag that is registered with Lake Formation representing the third-party engine. Lake Formation uses this tag to validate that if the request is coming from an approved engine. For more information about session tags, see Session tags in the IAM User Guide.
3. When setting up a third-party query engine execution role, you must have the following minimum set of permissions in the IAM policy:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": {"Effect": "Allow",
    "Action": [
```

```
        "lakeformation:GetDataAccess",
        "glue:GetTable",
        "glue:GetTables",
        "glue:GetDatabase",
        "glue:GetDatabases",
        "glue:CreateDatabase",
        "glue:GetUserDefinedFunction",
        "glue:GetUserDefinedFunctions",
        "glue:GetPartition",
        "glue:GetPartitions"
      ],
      "Resource": "*"
    }
}
```

4. Set up a role trust policy on the query engine execution role to have fine access control on which session tag key value pair can be attached to this role. In the following example, this role is only allowed to have session tag key `"LakeFormationAuthorizedCaller"` and session tag value `"engine1"` to be attached, and no other session tag key value pair is allowed.

```
{
    "Sid": "AllowPassSessionTags",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/query-execution-role"
    },
    "Action": "sts:TagSession",
    "Condition": {
    "StringLike": {
        "aws:RequestTag/LakeFormationAuthorizedCaller": "engine1"        }
    }
}
```

When `LakeFormationAuthorizedCaller` calls the STS:AssumeRole API operation to fetch credentials for the query engine to use, the session tag must be included in the AssumeRole request. The returned temporary credential can be used to make Lake Formation application integration API requests.

Lake Formation application integration API operations require the calling principal to be an IAM role. The IAM role must include a session tag with a predetermined value that has been registered

with Lake Formation. This tag allows Lake Formation to verify that the role used to call the application integration API operations is allowed to do so.

Enabling permissions for a third-party query engine to call application integration API operations

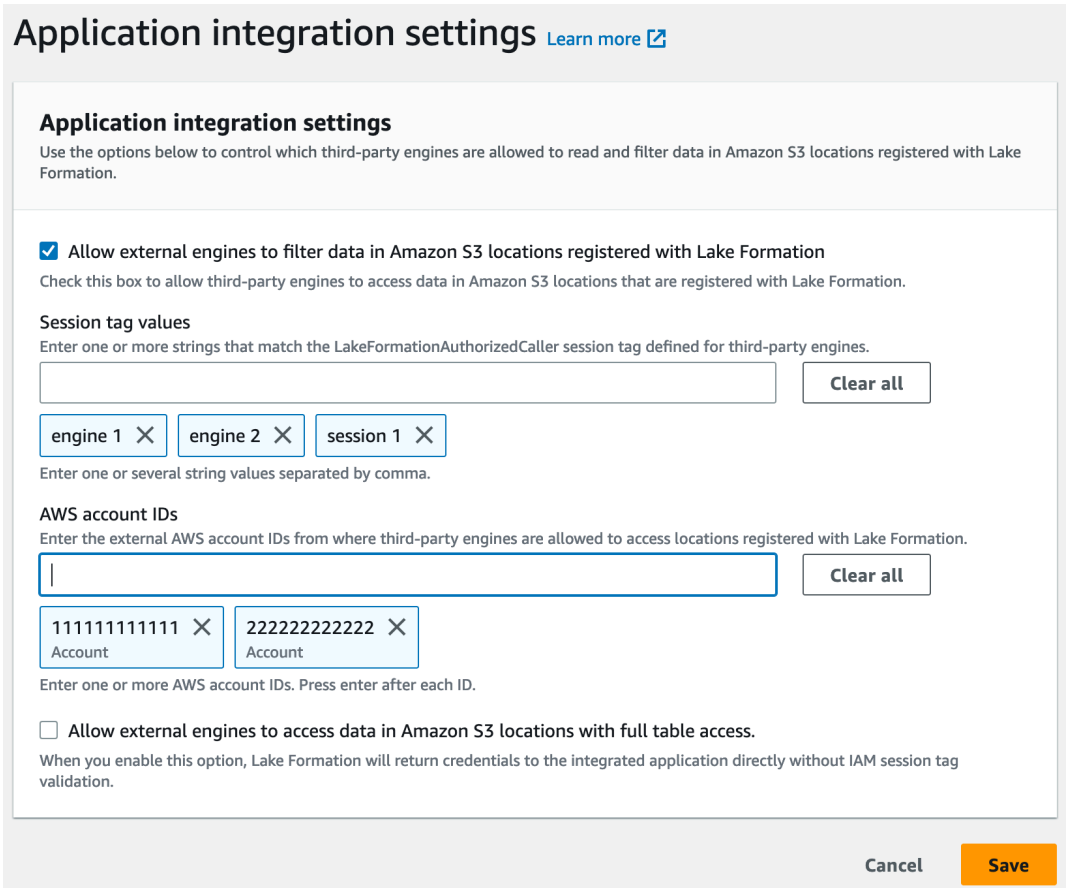
Follow these steps to allow a third-party query engine to call application integration API operations through the AWS Lake Formation console, the AWS CLI or API/SDK.

Console

To register your account for external data filtering:

1. Sign in to the AWS Management Console, and open the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the left-side navigation, expand **Administration**, and then choose **Application integration setting**.
3. On the **Application integration setting** page, choose the option **Allow external engines to filter data in Amazon S3 locations registered with Lake Formation**.
4. Enter the session tags that you created for the third-party engine. For information about session tags, see Passing session tags in AWS STS in the *AWS Identity and Access Management User Guide*.
5. Enter the account IDs for users that can use the third-party engine to access unfiltered metadata information and the data access credentials of resources in the current account.

   You can also use the AWS account ID field for configuring cross-account access.



CLI

Use the `put-data-lake-settings` CLI command to set the following parameters.

There are three fields to configure when using this AWS CLI command:

- `allow-external-data-filtering` – (boolean) Indicates that a third-party engine can access unfiltered metadata information and data access credentials of resources in the current account.
- `external-data-filtering-allow-list` – (array) A list of account IDs that can access unfiltered metadata information and data access credentials of resources in the current account when using a third-party engine. When AllowExternalDataFiltering is set to true, the ExternalDataFilteringAllowList property must include at least one account ID. An empty list is not allowed.
- `authorized-sessions-tag-value-list` – (array) A list of authorized session tag values (strings). If an IAM role credential has been attached with an authorized key-value pair, then if the session tag is included in the list, the session is granted access to unfiltered

metadata information and data access credentials on resources in the configured account. The authorized session tag key is defined as `*LakeFormationAuthorizedCaller*`.

- `AllowFullTableExternalDataAccess` - (boolean) Whether to allow a third-party query engine to get data access credentials without session tags when a caller has full data access permissions.

For example:

```
aws lakeformation put-data-lake-settings --cli-input-json file://
datalakesettings.json

{
  "DataLakeSettings": {
    "DataLakeAdmins": [
      {
        "DataLakePrincipalIdentifier": "arn:aws:iam::111111111111:user/lakeAdmin"
      }
    ],
    "CreateDatabaseDefaultPermissions": [],
    "CreateTableDefaultPermissions": [],
    "TrustedResourceOwners": [],
    "AllowExternalDataFiltering": true,
    "ExternalDataFilteringAllowList": [
        {"DataLakePrincipalIdentifier": "111111111111"}
        ],
    "AuthorizedSessionTagValueList": ["engine1"],
    "AllowFullTableExternalDataAccess": false
    }


}
```

API/SDK

Use the `PutDataLakeSetting` API operation to set the following parameters.

There are three fields to configure when using this API operation:

- `AllowExternalDataFiltering` – (Boolean) Indicates whether a third-party engine can access unfiltered metadata information and data access credentials of resources in the current account.

- `ExternalDataFilteringAllowList` – (array) A list of account IDs that can access unfiltered metadata information and the data access credentials of resources in the current account using a third-party engine.
- `AuthorizedSectionsTagValueList` – (array) A list of authorized tag values (strings). If an IAM role credential has been attached with an authorized tag, then the session is granted access to unfiltered metadata information and the data access credentials on resources in the configured account. The authorized session tag key is defined as `*LakeFormationAuthorizedCaller*`.
- `AllowFullTableExternalDataAccess` - (boolean) Whether to allow a third-party query engine to get data access credentials without session tags when a caller has full data access permissions.

For example:

```
//Enable session tag on existing data lake settings
public void sessionTagSetUpForExternalFiltering(AWSLakeFormationClient
 lakeformation) {
    GetDataLakeSettingsResult getDataLakeSettingsResult =
 lfClient.getDataLakeSettings(new GetDataLakeSettingsRequest());
    DataLakeSettings dataLakeSettings =
 getDataLakeSettingsResult.getDataLakeSettings();

    //set account level flag to allow external filtering
    dataLakeSettings.setAllowExternalDataFiltering(true);

    //set account that are allowed to call credential vending or Glue
 GetFilteredMetadata API
    List<DataLakePrincipal> allowlist = new ArrayList<>();
    allowlist.add(new
 DataLakePrincipal().withDataLakePrincipalIdentifier("111111111111"));
    dataLakeSettings.setWhitelistedForExternalDataFiltering(allowlist);

    //set registered session tag values
    List<String> registeredTagValues = new ArrayList<>();
    registeredTagValues.add("engine1");
    dataLakeSettings.setAuthorizedSessionTagValueList(registeredTagValues);

    lakeformation.putDataLakeSettings(new
 PutDataLakeSettingsRequest().withDataLakeSettings(dataLakeSettings));
}
```

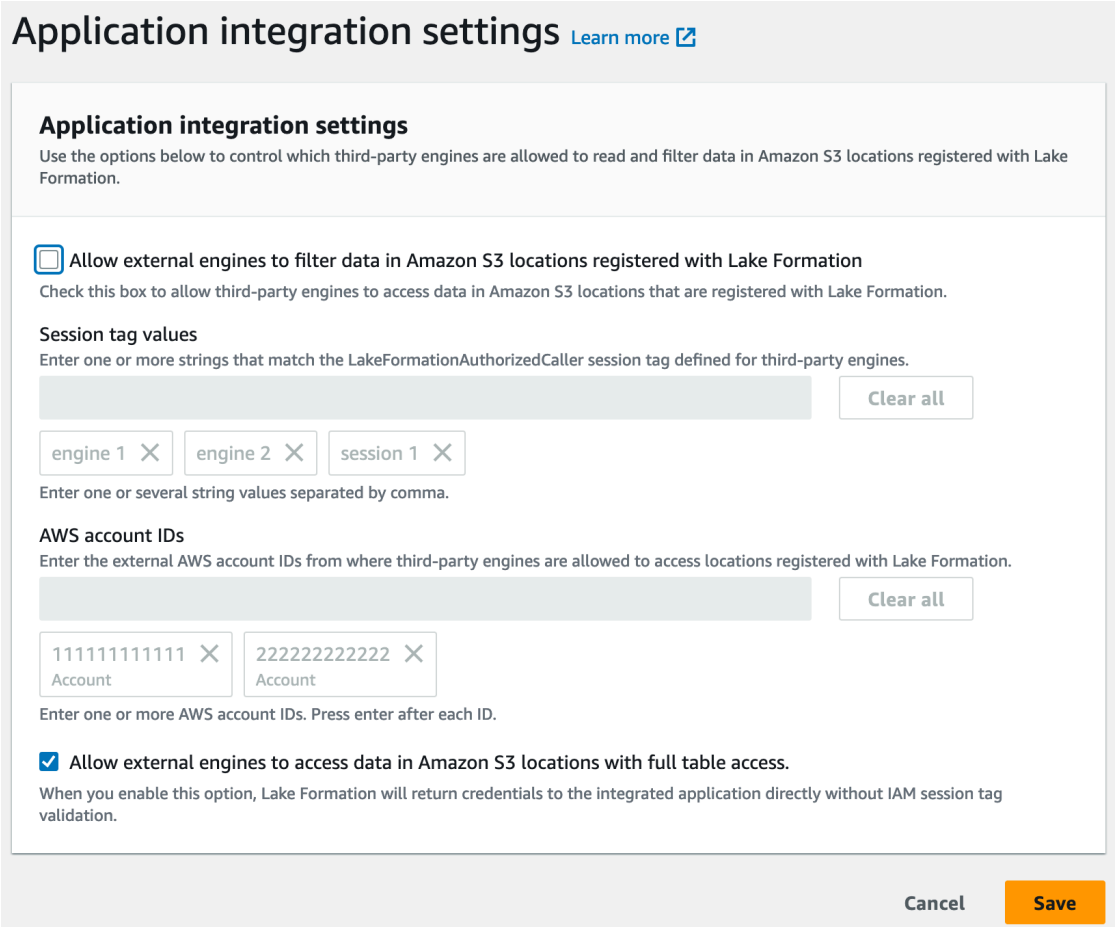
Application integration for full table access

Follow these steps to enable third-party query engines to access data without the IAM session tag validation:

Console

1. Sign in to the Lake Formation console at https://console.aws.amazon.com/lakeformation/.
2. In the left-side navigation, expand **Administration**, and choose **Application integration settings**.
3. On the **Application integration settings** page, choose the **Allow external engines to access data in Amazon S3 locations with full table access** option.

   When you enable this option, Lake Formation returns credentials to the querying application directly without IAM session tag validation.



AWS CLI

Use the `put-data-lake-settings` CLI command to set the `AllowFullTableExternalDataAccess` parameter.

```
aws lakeformation put-data-lake-settings –cli-input-json file://put-data-lake-
settings.json –region ap-northeast-1
{
    "DataLakeSettings": {
        "DataLakeAdmins": [
            {
                "DataLakePrincipalIdentifier": "arn:aws:iam::111111111111:user/
lakeAdmin"
            }
        ],
        "AllowFullTableExternalDataAccess": true
    }
}
```

Working with other AWS services

AWS services such as Amazon Athena, AWS Glue, Amazon Redshift Spectrum, and Amazon EMR can use AWS Lake Formation to securely access data in Amazon S3 locations registered with Lake Formation. With Lake Formation, you can define and manage fine-grained access control (FGAC) permissions for your tables in the AWS Glue Data Catalog. Each of these AWS services is a trusted caller to Lake Formation, and Lake Formation provides access to data stored in Amazon S3 through temporary credentials. For more information, see How Lake Formation application integration works.

To avail these capabilities, Lake Formation requires you to first register the Amazon S3 location, and assign appropriate permissions to the IAM principal for accessing the table, the database, and the Amazon S3 location. For more information see, Managing Lake Formation permissions.

The following tables lists the types of Lake Formation permissions supported by Amazon Athena, AWS Glue, Amazon EMR, and Amazon Redshift Spectrum to access data from AWS Glue standard tables and transactional tables (Apache Iceberg, Apache Hudi, and Linux foundation Delta Lake) with data stored in Amazon S3 and table metadata in the Data Catalog .

AWS services and supported permission types for AWS Glue standard tables and views

AWS service	Table-level permissions	Column-level permissions	Row and cell-level permissions
Athena SQL	Read/write access	Read access	Read access
Athena Spark	Not supported	Not supported	Not supported
Redshift Spectrum on a provisioned cluster or Amazon Redshift serverless	Read/write access	Read access	Read access
Apache Spark on Amazon EMR (EC2)	Read/write access	Read access	Read access
Apache Hive on Amazon EMR (EC2)	Read/write access	Read access	Not supported

AWS service	Table-level permissions	Column-level permissions	Row and cell-level permissions
Apache Spark on EMR Serverless	Read/write access	Read access	Read access
Apache Hive on EMR Serverless	Not supported	Not supported	Not supported
Amazon EMR on EKS	Not supported	Not supported	Not supported
AWS Glue ETL	Read/write access	AWS Glue 5.0 or higher supports read access.	AWS Glue 5.0 or higher supports read access.

Considerations and limitations

- Athena Spark doesn't support querying Data Catalog tables with Lake Formation permissions.
- Athena SAML-based users can read data sources secured using Lake Formation permissions by enabling SAML 2.0-based federation. SAML users can insert data into Parquet tables.
- Apache Spark on EMR Serverless doesn't support querying Data Catalog views.
- Apache Hive on EMR Serverless doesn't support querying tables with Lake Formation permissions.
- AWS Glue 5.0 or higher supports fine-grained access controls on Iceberg and Hive tables in the Data Catalog that are backed by S3. This capability lets you configure table, row, column, and cell level access controls for read queries within your AWS Glue for Apache Spark jobs.

  For more information, see AWS Glue versions.

AWS services and supported permission types for transactional table formats

AWS service	Iceberg	Hudi	Delta Lake (native)	Delta Lake (symlink tables)
Athena SQL	Supports reading tables with table,	Supports read and create operations on	Athena (engine version 3) supports reading	Athena (engine version 3) supports reading

AWS service	Iceberg	Hudi	Delta Lake (native)	Delta Lake (symlink tables)
	column, row, and cell-leve l permissions. Write operations require full table access.	tables with table, column, row, and cell-level permissio ns. Write operations are not supported.	native Delta Lake tables with table, column, row, and cell-level permissio ns. Write operations are not supported.	symlink Delta Lake tables with table, column, row, and cell-level permissio ns. Write operations are not supported.
Redshift Spectrum on a provisioned cluster	Supports reading tables with table, column, row, and cell-leve l permissions. Write operation s are not supported.	Supports reading tables with table, column, row, and cell-leve l permissions. Write operation s are not supported.	No supported	Supports reading Delta Lake tables via symlink manifest with table, column, row, and cell-level permissio ns. Write operations are not supported.
Apache Spark on Amazon EMR (EC2)	Supports reading tables with table, column, row, and cell-leve l permissions. Write operations require full table access.	Supports reading tables with table, column, row, and cell-leve l permissions. Write operations require full table access.	Supports reading tables with table, column, row, and cell-leve l permissions. Write operation s are not supported.	Supports reading tables with table, column, row, and cell-leve l permissions. Write operations require full table access.

AWS service	Iceberg	Hudi	Delta Lake (native)	Delta Lake (symlink tables)
AWS Glue ETL	AWS Glue 5.0 or higher supports reading tables with table, column, row, and cell-level permissions.	Supports read/write on tables with table-level permissions.	Supports read/write on tables with table-level permissions.	Supports read/write on tables with table-level permissions.

Topics

- Using AWS Lake Formation with Amazon Athena
- Using AWS Lake Formation with Amazon Redshift Spectrum
- Using AWS Lake Formation with AWS Glue
- Using AWS Lake Formation with Amazon EMR
- Using AWS Lake Formation with Quick
- Using AWS Lake Formation with AWS CloudTrail Lake

Using AWS Lake Formation with Amazon Athena

Amazon Athena is a server-less query service that helps you analyze structured, semi-structured, and unstructured data stored in Amazon S3. You can use Athena SQL to query data from CSV, JSON, Parquet, and Avro data formats. Athena SQL also supports table formats like Apache Hive, Apache Hudi, and Apache Iceberg. Athena integrates with the AWS Glue Data Catalog to store metadata of your data sets in Amazon S3. Athena can use Lake Formation to define and maintain access control policies on those data sets.

Here are some common use cases where you can use Lake Formation with Athena.

- Use Lake Formation permissions for accessing the Data Catalog resources (database and tables) from Athena. You can use either the named resource method or LF-tags to define permissions on database and tables. For more information, see:
  - Granting database permissions using the named resource method
  - Lake Formation tag-based access control

> **Note**
>
> Lake Formation permissions apply only when using Athena SQL to query source data from Amazon S3 and metadata in the Data Catalog.
> Athena Spark doesn't support querying Data Catalog tables with Lake Formation permissions. Lake Formation permissions support both read and write operations on databases and tables.

> **Note**
>
> You can't apply data filters when you use LF-Tags to manage permissions on Data Catalog resources.

- Control the query results by using Data filters in Lake Formation to secure tables in your Amazon S3 data lakes by granting permissions at column, row, and cell-levels. See the limitation on partition projection in Amazon Athena User Guide.
- Enforce fine-grained access control on the data available to the SAML-based Athena user when running federated queries.

  Athena JDBC and ODBC drivers support configuring federated access to your data source using SAML-based Identity Provider (IdP). Use Quick integrated with Lake Formation with your existing IAM role or SAML users or groups to visualize Athena query results.

  > **Note**
  >
  > Lake Formation permissions for SAML users and groups will apply only when you submit queries to Athena using the JDBC or ODBC driver.

  For more information, see Using Lake Formation and the Athena JDBC and ODBC drivers for federated access to Athena.

  > **Note**
  >
  > Currently, authorizing access to SAML identities in Lake Formation is not supported in the following regions:

- Middle East (Bahrain) - me-south-1
- Asia Pacific (Hong Kong) - ap-east-1
- Africa (Cape Town) - af-south-1
- China (Ningxia) - cn-northwest-1
- Asia Pacific (Osaka) - ap-northeast-3

- Use Cross-account data sharing in Lake Formation to query tables in another account.

Note

For more information on limitations when using Lake Formation permissions to `Views`, see Considerations and Limitations.

Support for transactional table formats

Applying Lake Formation permissions allows you to secure your transactional data in your Amazon S3 based data lakes. The table below lists transactional table formats supported in Athena and the Lake Formation permissions. Lake Formation enforces these permissions when Athena users run their queries.

Table format	Description and allowed operations	Lake Formation permissions supported in Athena
Apache Hudi	A format used to simplify incremental data processing and data pipeline developme nt. Athena supports create and read operations using Apache Hudi table formats on Amazon S3 data sets for both **Copy on Write (CoW)** and **Merge On Read (MoR)** Hudi table types. Athena does	Use Data filtering and cell-level security in Lake Formation to secure Hudi table using table, column, row, and cell-level permissio ns.

Table format	Description and allowed operations	Lake Formation permissions supported in Athena
	not support write operations on Hudi tables. Use Athena to query Hudi datasets.	
Apache Iceberg	An open table format that manages large collections of files as tables, and supports modern analytic data lake operations such as record-level insert, update, delete, and time travel queries. For more information on Athena's support for Iceberg tables, see Using Iceberg tables.	Table, column, row, and cell-level permissions are supported. Currently, Lake Formation doesn't support managing permissions on write operations such as `VACUUM`, `MERGE`, `UPDATE` and `OPTIMIZE` on tables in Open Table Formats.

Table format	Description and allowed operations	Lake Formation permissions supported in Athena
Linux Foundation Delta Lake	Delta Lake is an open-sour ce project that helps to implement modern data lake architectures commonly built on Amazon S3 or Hadoop Distributed File System (HDFS). Athena supports Delta lake tables created using a symlink-based manifest table definition on AWS Glue Data Catalog from a Delta Lake table. For more information, see Crawl Delta Lake tables using AWS Glue crawlers. Athena (engine version 3) supports reading native Delta Lake tables. For more information, see Introducing native Delta Lake table support with AWS Glue crawlers .	Table, column, row, and cell-level permissions are supported for symlink tables and native Delta Lake tables.

Additional resources

Blog posts, videos, and workshops

- Query an Apache Hudi dataset in an Amazon S3 data lake with Amazon Athena
- Build an Apache Iceberg data lake using Amazon Athena, Amazon EMR, and AWS Glue

- Insert, update, delete on Amazon S3 with Athena and Apache Iceberg
- LF-Tag based access control Lake Formation workshop on querying a data lake.

Using AWS Lake Formation with Amazon Redshift Spectrum

Amazon Redshift Spectrum lets you to query and retrieve data in Amazon S3 data lakes without loading data into Amazon Redshift cluster nodes.

Redshift Spectrum supports two ways of registering an external AWS Glue data catalog enabled with Lake Formation.

- Using a cluster attached IAM role that has permission to the Data Catalog

  To create an IAM role, follow the steps outlined in the below procedure.

  Controlling access to the AWS Glue Data Catalog
- Using a federated IAM identity configured to manage access to external AWS Glue Data Catalog resources

  Redshift Spectrum supports querying Lake Formation tables using federated IAM identities. The IAM identities can be an IAM user or an IAM role. For more information on IAM identity federation in Redshift Spectrum, see Using a federated identity to manage Amazon Redshift access to local resources and Redshift Spectrum external tables.

With Lake Formation integration with Redshift Spectrum, you can define row, column, and cell-level access control permissions on tables after your data is registered with Lake Formation.

For more information see Using Redshift Spectrum with AWS Lake Formation.

Redshift Spectrum supports reads or SELECT queries on the Lake Formation managed external schema tables.

For more information, see Creating external schemas for Redshift Spectrum.

Support for transactional table types

This table lists transactional table formats supported in Redshift Spectrum and the applicable Lake Formation permissions.

Supported table formats

Table format	Description and allowed operations	Lake Formation permissions supported in Redshift Spectrum
Apache Hudi	A format used to simplify incremental data processing and data pipeline development. Redshift Spectrum supports insert, delete, and upsert write operations using Apache Hudi **Copy on Write (CoW)** table format on Amazon S3. For more information, see Creating external tables for data managed in Apache Hudi.	Use Data filtering and cell-level security in Lake Formation to secure Hudi tables using table, column, row, and cell-level permissions.
Apache Iceberg	An open table format that manages large collections of files as tables and supports modern analytic data lake operations such as record-level insert, update, delete, and time travel queries. For more information, see Using Apache Iceberg tables with Amazon Redshift.	Redshift Spectrum supports Apache Iceberg tables for querying.
Linux Foundation Delta Lake	Delta Lake is an open-source project that helps implement modern data lake architectures commonly built on Amazon S3 or Hadoop	Table, column, row, and cell-level permissions are supported.

Table format	Description and allowed operations	Lake Formation permissions supported in Redshift Spectrum
	Distributed File System (HDFS). Redshift Spectrum supports querying Delta Lake tables. For more information, see Creating external tables for data managed in Delta Lake.	

Additional resources

Blog posts and workshops

- Centralize governance for your data lake using AWS Lake Formation while enabling a modern data architecture with Amazon Redshift Spectrum
- Use Redshift Spectrum to query Apache HUDI Copy On Write (CoW) tables in Amazon S3 data lake

Using AWS Lake Formation with AWS Glue

Data engineers and DevOps professionals use AWS Glue with Extract, Transform and Load (ETL) with Apache Spark to perform transformations on their data sets in Amazon S3 and load the transformed data into data lakes and data warehouses for analytics, machine learning, and application development. With different teams accessing the same data set in Amazon S3, it is imperative to grant and restrict permissions based on their roles.

AWS Lake Formation is built on AWS Glue, and the services interact in the following ways:

- Lake Formation and AWS Glue share the same Data Catalog.
- The following Lake Formation console features invoke the AWS Glue console:
  - Jobs – For more information, see Adding Jobs in the *AWS Glue Developer Guide*.
  - Crawlers – For more information, see Cataloging Tables with a Crawler in the *AWS Glue Developer Guide*.

- The workflows generated when you use a Lake Formation blueprint are AWS Glue workflows. You can view and manage these workflows in both the Lake Formation console and the AWS Glue console.
- Machine learning transforms are provided with Lake Formation and are built on AWS Glue API operations. You create and manage machine learning transforms on the AWS Glue console. For more information, see Machine Learning Transforms in the *AWS Glue Developer Guide*.

You can use the Lake Formation fine-grained access control to manage your existing Data Catalog resources and Amazon S3 data locations.

> **Note**
>
> AWS Glue 5.0 or higher supports fine-grained access controls on Iceberg and Hive tables that are backed by S3. This capability lets you configure table, row, column, and cell level access controls for read queries within your AWS Glue for Apache Spark jobs.

Support for transactional table types

Applying Lake Formation permissions allows you to secure your transactional data in your Amazon S3 based data lakes. The table below lists transactional table formats supported in AWS Glue and the Lake Formation permissions. Lake Formation enforces these permissions for AWS Glue operations.

Supported table formats

Table format	Description and allowed operations	Lake Formation permissions supported in AWS Glue
Apache Hudi	A open table format used to simplify incremental data processing and data pipeline development. For examples, see Using the Hudi framework in AWS Glue.	Table-level permissions are available for Hudi tables. For more information, see Limitations.

Table format	Description and allowed operations	Lake Formation permissions supported in AWS Glue
Apache Iceberg	An open table format that manages large collections of files as tables. For examples, see Using the Iceberg framework in AWS Glue.	AWS Glue version 5.0 and higher lets you configure table, row, column, and cell level access controls for read queries within your AWS Glue for Apache Spark jobs for Iceberg tables. For more information, see Limitations.
Linux Foundation Delta Lake	Delta Lake is an open-source project that helps implement modern data lake architect ures commonly built on Amazon S3 or Hadoop Distributed File System (HDFS). For examples, see Using the Delta Lake framework in AWS Glue.	Table-level permissions are available for Delta Lake tables. For more information, see Limitations.

Additional resources

Blog posts and repositories

- Use the AWS Glue connector to read and write Apache Iceberg tables with ACID transactions and perform time travel
- Writing to Apache Hudi tables using AWS Glue custom connector
- AWS repository of Cloudformation template and pyspark code sample to analyze streaming data using AWS Glue, Apache Hudi, and Amazon S3.

Using AWS Lake Formation with Amazon EMR

Amazon EMR is a flexible AWS managed cluster platform on which you can run any custom code on supported big data frameworks like Hadoop Map-Reduce, Spark, Hive, Presto, etc. Organizations also use Amazon EMR to run both batch and stream data processing applications across a highly distributed cluster. Using Apache Spark on Amazon EMR, you can run your data transformations and custom code on database and tables whose permissions are managed by Lake Formation.

There are three options for deploying Amazon EMR:

- EMR on EC2
- EMR Serverless
- Amazon EMR on EKS

For more information, see Integrate Amazon EMR with Lake Formation or Using EMR Serverless with AWS Lake Formation for fine-grained access control

Support for transactional table formats

Amazon EMR releases 6.15.0 and higher include support for Lake Formation table, row, column, and cell-level access control permissions on Apache Hudi , Apache Iceberg and Delta Lake table formats when you read and write data with Spark SQL.

For limitations, see Considerations for Amazon EMR with Lake Formation.

Supported table formats

Table format	Description and allowed operations	Lake Formation permissions supported in Amazon EMR
Apache Hudi	A open table format used to simplify incremental data processing and data pipeline development. For a list of supported operations, see Apache Hudi and Lake Formation.	Amazon EMR supports table, row, column, and cell-level access control with Apache Hudi.

Table format	Description and allowed operations	Lake Formation permissions supported in Amazon EMR
Apache Iceberg	An open table format that manages large collections of files as tables. For a list of supported operations, see Apache Iceberg and Lake Formation.	Amazon EMR supports table, row, column, and cell-level access control with Apache Iceberg.
Linux Foundation Delta Lake	Delta Lake is an open-source project that helps implement modern data lake architect ures commonly built on Amazon S3 or Hadoop Distributed File System (HDFS). For a list of supported operations, see Delta Lake and Lake Formation.	Amazon EMR supports table, row, column, and cell-level access control with Delta Lake tables.

Additional resources

User guide, blog posts, and workshops

- Integration with Amazon EMR using Runtime Roles
- Get a quick start with Apache Hudi, Apache Iceberg, and Delta Lake with Amazon EMR on EKS
- Using Delta Lake OSS with EMR Serverless

Using AWS Lake Formation with Quick

Quick supports exploring datasets managed by Lake Formation permissions in Amazon S3 using Athena.

Both Standard and Enterprise edition users of Quick integrate with Lake Formation, but slightly differently.

- Enterprise edition – Grant fine-grained access control (FGAC) permissions to individual Quick users and groups to access databases and tables.
- Standard edition – Grant permissions to IAM roles to access databases and tables.

> **Note**
>
> By default, Quick uses a role named `aws-quicksight-service-role-v0`. You can also define custom roles with required permissions that enable Quick to access Athena.

For more information, see Authorizing connections through AWS Lake Formation

Additional resources

Blog posts

- Enable fine-grained permissions for Quick authors in AWS Lake Formation
- Securely analyze your data with AWS Lake Formation and Quick

Using AWS Lake Formation with AWS CloudTrail Lake

AWS CloudTrail Lake supports exploring event data stores using Amazon Athena with fine-grained permissions in AWS Lake Formation.

> **Note**
>
> CloudTrail Lake can only be queried through Amazon Athena.

To register your CloudTrail Lake event data store with Lake Formation, see Federate an event data store.

Logging AWS Lake Formation API calls using AWS CloudTrail

AWS Lake Formation is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Lake Formation. CloudTrail captures all Lake Formation API calls as events. The calls captured include calls from the Lake Formation console, the AWS Command Line Interface, and code calls to the Lake Formation API actions. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Lake Formation. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Lake Formation, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the *AWS CloudTrail User Guide*.

Lake Formation information in CloudTrail

CloudTrail is enabled by default when you create a new AWS account. When activity occurs in Lake Formation, that activity is recorded as a CloudTrail event along with other AWS service events in **Event history**. An event represents a single request from any source and includes information about the requested action, the date and time of the action, and request parameters. In addition, every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

You can view, search, and download recent events for your AWS account. For more information, see Viewing events with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for Lake Formation, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you

create a trail on the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services, such as Amazon Athena, to further analyze and act upon the event data collected in CloudTrail logs. CloudTrail can also deliver log files to Amazon CloudWatch Logs and CloudWatch Events.

For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

Understanding Lake Formation events

All Lake Formation API actions are logged by CloudTrail and are documented in the AWS Lake Formation Developer Guide. For example, calls to the `PutDataLakeSettings`, `GrantPermissions`, and `RevokePermissions` actions generate entries in the CloudTrail log files.

The following example shows a CloudTrail event for the `GrantPermissions` action. The entry includes the user who granted the permission (`datalake_admin`), the principal that the permission was granted to (`datalake_user1`), and the permission that was granted (`CREATE_TABLE`). The entry also shows that the grant failed because the target database was not specified in the `resource` argument.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAZKE67KM3P775X74U2",
        "arn": "arn:aws:iam::111122223333:user/datalake_admin",
        "accountId": "111122223333",
        "accessKeyId": "...",
        "userName": "datalake_admin"
    },
    "eventTime": "2021-02-06T00:43:21Z",
    "eventSource": "lakeformation.amazonaws.com",
```

```
    "eventName": "GrantPermissions",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.65",
    "userAgent": "aws-cli/1.19.0 Python/3.6.12
 Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 botocore/1.20.0",
    "errorCode": "InvalidInputException",
    "errorMessage": "Resource must have one of the have either the catalog, table or
 database field populated.",
    "requestParameters": {
        "principal": {
            "dataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/
datalake_user1"
        },
        "resource": {},
        "permissions": [
            "CREATE_TABLE"
        ]
    },
    "responseElements": null,
    "requestID": "b85e863f-e75d-4fc0-9ff0-97f943f706e7",
    "eventID": "8d2ccef0-55f3-42d3-9ede-3a6faedaa5c1",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

The next example shows a CloudTrail log entry for the `GetDataAccess` action. Principals do not directly call this API. Rather, `GetDataAccess` is logged whenever a principal or integrated AWS service requests temporary credentials to access data in a data lake location that is registered with Lake Formation.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AWSAccount",
        "principalId": "AROAQGFTBBBGOBWV2EMZA:GlueJobRunnerSession",
        "accountId": "111122223333"
    },
    "eventSource": "lakeformation.amazonaws.com",
    "eventName": "GetDataAccess",
...
```

```
...
    "additionalEventData": {
        "requesterService": "GLUE_JOB",
        "lakeFormationPrincipal": "arn:aws:iam::111122223333:role/ETL-Glue-Role",
        "lakeFormationRoleSessionName": "AWSLF-00-GL-111122223333-G13T0Rmng2"
    },
...
}
```

Tracing data access in CloudTrail with Lake Formation

Lake Formation issues temporary, scoped-down credentials to query engines—such as , Amazon EMR, and AWS Glue—for accessing table data locations in Amazon S3. Lake Formation uses the IAM role that you specify when you register a data location with Lake Formation for vending these credentials.

Change in CloudTrail logging behavior (February 2026)

Before February 2026, when a query engine made an `s3:GetObject` call to fetch query data, CloudTrail logs displayed the Lake Formation registration IAM role as the principal.

Starting in February 2026, CloudTrail logs for `s3:GetObject` calls also include the IAM unique identifier (beginning with AR0A) of the IAM role that initiated the query. This identifier corresponds to the IAM role that initiated the query, Amazon EMR job, or AWS Glue ETL job.

Tracing data access from query initiation to Amazon S3

Each `s3:GetObject` call in CloudTrail corresponds to a `lakeformation:GetDataAccess` call. You can use the IAM role identifier that is present in both log entries to trace the caller's IAM role from query initiation through to data access in Amazon S3.

To enable the information of source identity in the Amazon S3 data events CloudTrail logs, follow the below steps. For enabling Amazon S3 data events in CloudTrail, refer Enabling CloudTrail event logging for Amazon S3 buckets and objects.

> **Important**
>
> Enabling Amazon S3 data events in CloudTrail incurs additional charges.

To enable source identity tracking in CloudTrail logs

1. Using , run the `get-data-lake-settings` command. In the response, edit the `Parameters` field to add `"SET_SOURCE_IDENTITY": "TRUE"`.

   Run the `put-data-lake-settings` command.

   ```
   "Parameters": {
       "CROSS_ACCOUNT_VERSION": "4",
       "SET_SOURCE_IDENTITY": "TRUE"
   },
   ```

2. In the trust policy of the IAM role used to register your Amazon S3 data location with Lake Formation, add `sts:SetSourceIdentity`:

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": [
                       "lakeformation.amazonaws.com"
                   ]
               },
               "Action": [
                   "sts:AssumeRole",
                   "sts:SetSourceIdentity"
               ]
           }
       ]
   }
   ```

The following log shows the source identity information in an example `s3:GetObject` CloudTrail log.

```
"sessionContext": {
    "sessionIssuer": {
        "type": "Role",
        "principalId": "AROA2EXAMPLEXYZ3AB6CDKLM",
        "arn": "arn:aws:iam::111122223333:role/LFRegistrationRole",
```

```
        "accountId": "111122223333",
        "userName": "LFRegistrationRole"
    },
    "sourceIdentity": "V2.111122223333.AROA3EXAMPLEPQR7ST9UVWXY"
},
"invokedBy": "glue.amazonaws.com"
```

See Also

- For information about tracking IAM Identity Center user context in CloudTrail logs, see Integrating IAM Identity Center.

Amazon Redshift federated catalog access monitoring

Amazon Redshift federated catalogs in the AWS Glue Data Catalog do not generate `GetDataAccess` events in Lake Formation CloudTrail logs. To track data access for Amazon Redshift tables in the Data Catalog, monitor `GetTable` and `BatchGetTable` events, which capture metadata access patterns for Redshift federated tables, including table name, database name, and catalog ID information.

See Also

- Cross-account CloudTrail logging

Lake Formation best practices, considerations, and limitations

Use this section to quickly find best practices, considerations, and limitations within AWS Lake Formation.

See Service quotas for the maximum number of service resources or operations for your AWS account.

Topics

- Cross-account data sharing best practices and considerations
- Service-linked role limitations
- Cross-Region data access limitations
- Data Catalog views considerations and limitations
- Data filtering limitations
- Hybrid access mode considerations and limitations
- Limitations for bringing Amazon Redshift data warehouse data into the AWS Glue Data Catalog
- S3 Tables catalog integration limitations
- Hive metadata store data sharing considerations and limitations
- Amazon Redshift data sharing limitations
- IAM Identity Center integration limitations
- Lake Formation tag-based access control best practices and considerations
- Attribute-based access control considerations, limitations, and supported regions
- Row-level security policy limitations

Cross-account data sharing best practices and considerations

Lake Formation cross-account capabilities allow users to securely share distributed data lakes across multiple AWS accounts, AWS organizations or directly with IAM principals in another account providing fine-grained access to the Data Catalog metadata and underlying data.

Consider the following best practices when using Lake Formation cross-account data sharing:

- There is no limit to the number of Lake Formation permission grants that you can make to principals in your own AWS account. However, Lake Formation uses AWS Resource Access Manager (AWS RAM) capacity for cross-account grants that your account can make with the named resource method. To maximize the AWS RAM capacity, follow these best practices for the named resource method:
  - Use the new cross-account grant mode (**Version 3** and above under **Cross account version settings**) to share a resource with an external AWS account. For more information, see Updating cross-account data sharing version settings.
  - Arrange AWS accounts into organizations, and grant permissions to organizations or organizational units. A grant to an organization or organizational unit counts as one grant.

    Granting to organizations or organizational units also eliminates the need to accept an AWS Resource Access Manager (AWS RAM) resource share invitation for the grant. For more information, see Accessing and viewing shared Data Catalog tables and databases.
  - Instead of granting permissions on many individual tables in a database, use the special **All tables** wildcard to grant permissions on all tables in the database. Granting on **All tables** counts as a single grant. For more information, see Granting permissions on Data Catalog resources.

  > **Note**
  >
  > For more information about requesting a higher limit for the number of resource shares in AWS RAM, see AWS service quotas in the *AWS General Reference*.

- You must create a resource link to a shared database for that database to appear in the Amazon Athena and Amazon Redshift Spectrum query editors. Similarly, to be able to query shared tables using Athena and Redshift Spectrum, you must create resource links to the tables. The resource links then appear in the tables list of the query editors.

  Instead of creating resource links for many individual tables for querying, you can use the **All tables** wildcard to grant permissions on all tables in a database. Then, when you create a resource link for that database and select that database resource link in the query editor, you'll have access to all tables in that database for your query. For more information, see Creating resource links.
- When you share resources directly with principals in another account, the IAM principal in the recipient account may not have permission to create resource links to be able to query the shared tables using Athena and Amazon Redshift Spectrum. Instead of creating a resource link for each

table that is shared, the data lake administrator can create a placeholder database and grant `CREATE_TABLE` permission to the `ALLIAMPrincipal` group. Then, all IAM principals in the recipient account can create resource links in the placeholder database and start querying the shared tables.

See the example CLI command for granting permissions to `ALLIAMPrincipals` in Granting database permissions using the named resource method.

- When cross-account permissions are granted directly to a principal, only the recipient of the grant can view these permissions. The data lake administrator in the recipient's AWS account cannot view these direct grants.
- Athena and Redshift Spectrum support column-level access control, but only for inclusion, not exclusion. Column-level access control is not supported in AWS Glue ETL jobs.
- When a resource is shared with your AWS account, you can grant permissions on the resource only to users in your account. You can't grant permissions on the resource to other AWS accounts, to organizations (not even your own organization), or to the `IAMAllowedPrincipals` group.
- You can't grant `DROP` or `Super` on a database to an external account.
- Revoke cross-account permissions before you delete a database or table. Otherwise, you must delete orphaned resource shares in AWS Resource Access Manager.

See also

- Lake Formation tag-based access control best practices and considerations
- `CREATE_TABLE` in the Lake Formation permissions reference for more cross-account access rules and limitations.

Service-linked role limitations

A service-linked role is a special type of IAM role that's linked directly to AWS Lake Formation. This role has pre-defined permissions that allow Lake Formation to perform actions on your behalf across AWS services.

The following limitations apply when using a service-linked role (SLR) to register data locations with Lake Formation.

- You can't modify service-linked role policies once created.
- A service linked role doesn't support encrypted catalog resources sharing across accounts. Encrypted resources require specific AWS KMS key permissions. Service-linked roles have pre-defined permissions that don't include the ability to work with encrypted catalog resources across accounts.
- When registering multiple Amazon S3 locations, using service-linked role may cause you to exceed your IAM policy limits quickly. This happens because with service-linked roles, AWS writes the policy for you, and it increments as one large block that includes all your registrations. You can write customer-managed policies more efficiently, distribute permissions across multiple policies, or use different roles for different Regions.
- Amazon EMR on EC2 can't access data you register data locations with service-linked roles.
- Service-linked role operations bypass your AWS service control policies.
- When you register data locations with a service-linked role, it updates IAM policies with eventual consistency. For more information, see the the Troubleshoot IAM documentation in the IAM User Guide.
- You can't set `SET_CONTEXT = TRUE` in Lake Formation data lake settings when using service-linked roles, and you are using IAM Identity Center. The reason is that service-linked roles have immutable trust policies that are incompatible with the trusted identity propagation needed for `SetContext` auditing with IAM Identity Center principals.

Cross-Region data access limitations

Lake Formation supports querying Data Catalog tables across AWS Regions. You can access data in a Region from other Regions using Amazon Athena, Amazon EMR, and AWS Glue ETL by creating resource links in other Regions pointing to the source databases and tables. With cross-Region table access, you can access data across Regions without copying the underlying data or the metadata into the Data Catalog.

The following limitations apply to cross-Region table access.

- Lake Formation doesn't support querying Data Catalog tables from another Region using Amazon Redshift Spectrum.
- In the Lake Formation console, the database and table views don't show the source Region database/table names.

- To view the list of tables under a shared database from another Region, you need to first create a resource link to the shared database, then select the resource link, and choose **View tables**.
- Lake Formation doesn't support cross-Region resource link calls made by SAML users.
- Lake Formation's cross-Region feature doesn't involve additional charges for data transfers.

Data Catalog views considerations and limitations

The following considerations and limitations apply to Data Catalog views.

- You cannot create a Data Catalog view from the Lake Formation console. You can create views using the AWS CLI or SDK.
- You can create Data Catalog views from 10 tables. It is a hard limit. Underlying reference tables for a view can belong to the same database or different databases within the same AWS account.
- For additional considerations and limitations specific to creating Data Catalog views using Redshift, see the Data Catalog views considerations and limitations section in the Amazon Redshift Database Developer Guide. For Athena, see the Data Catalog views considerations and limitations section in the Amazon Athena User Guide.
- You can create Data Catalog views on tables registered with Lake Formation in both hybrid access mode and Lake Formation mode.

  When using Data Catalog views with Lake Formation hybrid access mode, it is recommended to ensure that the view consuming principals are opted in to Lake Formation permissions for the base tables referenced in the view without granting access. This ensures that the base tables will not be revealed to consumers through AWS Glue IAM permissions.
- There are no restrictions on the cross-account sharing version to share views.
- Views get versioned just like Data Catalog tables, when you use the `ALTER VIEW` statement for an already created view dialect. You cannot roll back to a previous view because the view version changes with the underlying data changes. You can delete a view version and it will default to the next available latest version. When you change the view version, make sure your data is in sync with the selected view version schema.
- No new Data Catalog APIs are introduced. The existing `CreateTable`, `UpdateTable`, `DeleteTable` and `GetTable` APIs are updated.
- Amazon Redshift always creates views with varchar columns from tables with strings. You must cast string columns to varchar with an explicit length when adding dialects from other engines.

- Granting data lake permissions to `All tables` within a database will result in the grantee having permissions on all tables and views within the database.
- You can't create views:
  - That reference other views.
  - When the reference table is a resource link.
  - When the reference table is in another account.
  - From external Hive metastores.
- Cross-account definer roles are not supported for Redshift Spectrum Dialect views.
- Resource links for the Athena dialect in the Athena query editor is not supported. To use cross-account definer roles for the Athena dialect, add the account that hosts the base tables as a Data Source in Athena.

Data filtering limitations

When you grant Lake Formation permissions on a Data Catalog table, you can include data filtering specifications to restrict access to certain data in query results and engines integrated with Lake Formation. Lake Formation uses data filtering to achieve column-level security, row-level security, and cell-level security. You can define and apply data filters on nested columns if your source data contains nested structures.

Notes and restrictions for column-level filtering

There are three ways to specify column filtering:

- By using data filters
- By using simple column filtering or nested column filtering.
- By using TAGs.

Simple column filtering just specifies a list of columns to include or exclude. Both the Lake Formation console, the API, and the AWS CLI support simple column filtering. For an example, see Grant with Simple Column Filtering.

The following notes and restrictions apply to column filtering:

- AWS Glue 5.0 or higher supports fine-grained access control via Lake Formation only for Apache Hive and Apache Iceberg tables.

- To grant SELECT with the grant option and column filtering, you must use an include list, not an exclude list. Without the grant option, you can use either include or exclude lists.
- To grant SELECT on a table with column filtering, you must have been granted SELECT on the table with the grant option and without any row restrictions. You must have access to all rows.
- If you grant SELECT with the grant option and column filtering to a principal in your account, that principal must specify column filtering for the same columns or a subset of the granted columns when granting to another principal. If you grant SELECT with the grant option and column filtering to an external account, the data lake administrator in the external account can grant SELECT on all columns to another principal in their account. However, even with SELECT on all columns, that principal will have visibility only on the columns granted to the external account.
- You can't apply column filtering on partition keys.
- A principal with the SELECT permission on a subset of columns in a table can't be granted the ALTER, DROP, DELETE, or INSERT permission on that table. For a principal with the ALTER, DROP, DELETE, or INSERT permission on a table, if you grant the SELECT permission with column filtering, it has no effect.

The following notes and restrictions apply to nested column filtering:

- You can include or exclude five-levels of nested fields in a data filter.

  Example

  Col1.Col1_1.Col1_1_1.Col1_1_1_1.Col1_1_1_1_1
- You can't apply column filtering on nested fields within partition columns.
- If your table schema contains a top-level column name ("customer"."address") that has the same pattern of a nested field representation within a data filter (a nested column with a top level column name `customer` and a nested field name `address` is specified as `"customer"."address"` in a data filter), you can't explicitly specify access to either top level column or nested field because both are represented using the same pattern in the inclusion/exclusion lists. This is ambiguous, and Lake Formation can't resolve if you're specifying the top level column or the nested field.
- If a top level column or nested field contains a double quote within the name, you must include a second double quote when you specify access to a nested field within a data cells filter's include and exclude list.

Example

Example nested column name with double quotes – `a.b.double"quote`

Example

Example nested column representation within a data filter – `"a"."b"."double""quote"`

Cell-level filtering limitations

Keep in mind the following notes and restrictions for row-level and cell-level filtering.

- Cell-level security is not supported on nested columns, views, and resource links.
- All expressions that are supported on top level columns are also supported on nested columns. However, nested fields under partition columns should **NOT** be referenced when defining nested row-level expressions.
- Cell-level security is available in all regions when using Athena engine version 3 or Amazon Redshift Spectrum. For other services, cell-level security is only available in the regions mentioned on the Supported Regions.
- `SELECT INTO` statements are not supported.
- The `array`, and `map` data types aren't supported in row filter expressions. The `struct` data type is supported.
- There is no limit to the number of data filters that can be defined on a table, but there is a limit of 100 data filter for a single principal on a table.
- To apply a data filter with a row filter expression, you must have `SELECT` with the grant option on all table columns. This restriction doesn't apply to administrators in external accounts when the grant was made to the external account.
- If a principal is a member of a group and both the principal and the group are granted permissions on a subset of rows, the principal's effective row permissions are the union of the principal's permissions and the group's permissions.
- The following column names are restricted in a table for row-level and cell-level filtering:
  - ctid
  - oid
  - xmin

- cmin
- xmax
- cmax
- tableoid
- insertxid
- deletexid
- importoid
- redcatuniqueid

- If you apply the all-rows filter expression on a table concurrently with other filter expressions with predicates, the all-rows expression will prevail over all other filter expressions.
- When permissions on a subset of rows are granted to an external AWS account and the data lake administrator of the external account grants those permissions to a principal in that account, the principal's effective filter predicate is the intersection of the account's predicate and any predicate that was directly granted to the principal.

  For example, if the account has row permissions with the predicate `dept='hr'` and the principal was separately granted permission for `country='us'`, the principal has access only to rows with `dept='hr'` and `country='us'`.

For more information about cell-level filtering, see Data filtering and cell-level security in Lake Formation.

For consideration and limitations when querying tables using Amazon Redshift Spectrum with row-level security policies, see Considerations and limitations using RLS policies in the Amazon Redshift Database Developer Guide.

Hybrid access mode considerations and limitations

Hybrid access mode provides the flexibility to selectively enable Lake Formation permissions for databases and tables in your AWS Glue Data Catalog.
With the Hybrid access mode, you now have an incremental path that allows you to set Lake Formation permissions for a specific set of users without interrupting the permission policies of other existing users or workloads.

The following considerations and limitations apply to hybrid access mode.

Limitations

- **Update Amazon S3 location registration** – You can't edit parameters of a location that is registered with Lake Formation using a service linked role.
- **Opt in option when using LF-Tags** – When you can grant Lake Formation permissions using LF-Tags, you can opt in principals to enforce Lake Formation permissions as a consecutive step by choosing databases and tables that has LF-Tags attached.
- **Hybrid access mode access** – Access to hybrid access mode in Lake Formation is limited to users with data lake administrator or read-only administrator permissions.
- **Opt in principals** – Currently, only a data lake administrator role can opt in principals to resources.
- **Opt in all tables in a database** – In cross-account grants, when you grant permissions, and opt in all tables in a database, you need to opt in the database also for the permissions to work.

Considerations

- **Updating Amazon S3 location registered with Lake Formation to hybrid access mode** – We do not recommend converting a Amazon S3 data location that is already registered with Lake Formation to hybrid access mode though it can be done.
- **API behaviors when a data location is registered in hybrid access mode**
  - CreateTable – The location is considered as registered with Lake Formation regardless of the hybrid access mode flag and opt in status. Thus, the user requires the data location permission to create a table.
  - CreatePartition/BatchCreatePartition/UpdatePartition (when partition location is updated to point to the location registered with hybrid) – The Amazon S3 location is considered as registered with Lake Formation regardless of the hybrid access mode flag and opt in status.
  - CreateDatabase/UpdateDatabase (when database location is updated to point to the location registered in hybrid access mode) – The location is considered as registered with Lake Formation regardless of the hybrid access mode flag and opt in status. Thus, the user requires the data location permission to create or update a database.
  - UpdateTable (when a table location is updated to point to the location registered in hybrid access mode) – The location is considered as registered with Lake Formation regardless of the hybrid access mode flag and opt in status. Thus, the user requires data location permission to update the table. If the table location is not updated or it is pointing to a location that is not

registered with Lake Formation, the user doesn't require data location permission to update the table.

Limitations for bringing Amazon Redshift data warehouse data into the AWS Glue Data Catalog

You can catalog, and manage access to analytic data in Amazon Redshift data warehouses using the AWS Glue Data Catalog. The following limitations apply:

- Cross-account sharing is not supported at the federated catalog level. However, you can share individual databases and tables from within a federated catalog across AWS accounts.
- You must have **Cross account version settings** version 4 or higher for sharing databases or tables in the federated catalog across AWS accounts.
- The Data Catalog supports the creation of only top-level catalogs.
- You can only update the description of catalogs in the Redshift Managed Storage (RMS).
- Setting up permissions on federated catalogs as well as databases and tables in the federated catalog to `IAMAllowedPrincipals` group is not supported.

- Data Definition Language (DDL) operations on the catalog from engines like Athena, Amazon EMR Spark, or others, including setting catalog configurations, are not supported.
- Performing DDL operations on RMS tables using Athena is not supported.
- Creating materialized views is not supported, whether it is through Athena, Apache Spark, the AWS Glue Data Catalog, or the Amazon Redshift consumer.
- Athena doesn't support a multi-catalog experience. It can only connect to a single, specific catalog at a time. Athena can't access or query across multiple catalogs simultaneously.
- Tagging and branching operations on Iceberg tables through Athena and Amazon Redshift are not supported.
- Time Travel on RMS tables is not supported.
- Multi-level catalogs with data lake tables are not supported. All data stored in Amazon S3 for use with data lake tables must reside in the default AWS Glue Data Catalog, and can't be organized into multi-level catalogs.

- In Amazon Redshift, datashares are not added to the registered namespace. Clusters and namespaces are synonymous, once you publish a cluster to the AWS Glue Data Catalog, you can't add new data.
- Amazon EMR on EC2 doesn't support joining across RMS tables and Amazon S3 tables. Only EMR Serverless supports this capability.
- External schemas and tables are not supported.
- RMS tables are accessible only from the extension endpoint in AWS Glue Iceberg REST Catalog.
- Hive tables are not accessible from third-party engines connected to the AWS Glue Iceberg REST Catalog.
- The read_committed isolation level on RMS tables through Spark will be supported.
- Redshift database names are treated as case-insensitive in the AWS Glue Data Catalog, restricted to 128 characters, and can be alphanumeric with dashes (-) and underscores (_).
- The catalog names are case-insensitive, restricted to 50 characters, and can be alphanumeric with dashes (-) and underscores (_).
- Amazon Redshift doesn't support using Lake Formation SQL-style GRANT and REVOKE commands to manage access permissions on tables published to the AWS Glue Data Catalog.
- Row-level security and dynamic data masking policies that are attached to the producer (source) Amazon Redshift cluster will not be enforced. Instead, access permissions defined in Lake Formation will be enforced on the shared data.
- Performing Data Definition Language (DDL) and Data Manipulation Language (DML) operations on table links are not supported.
- If reserved keywords are not properly escaped, it will result in failures or errors.
- Encryption of data in multi-catalog scenarios is not supported.

S3 Tables catalog integration limitations

Amazon S3 Tables integrates with AWS Glue Data Catalog (Data Catalog) and registers the catalog as a Lake Formation data location. You can set up this registration from the Lake Formation console or by using the service APIs.

> **Note**
>
> If you have an IAM or S3 Tables resource-based policy that restricts IAM users and IAM roles based on principal tags, you must attach the same principal tags to the IAM role that Lake

Formation uses to access your S3 data (for example, LakeFormationDataAccessRole) and grant this role the necessary permissions. This is required for your tag-based access control policy to work correctly with your S3 Tables analytics integration.

The following limitations apply to integrating S3 Tables catalog with Data Catalog and Lake Formation:

- AWS Glue and Lake Formation don't support mixed-case column names, and convert all column names to lowercase. You must verify that table column names are unique when converted to lowercase. Use `customer_id` instead of `customerId`. The use of mixed-case column names was supported only during the preview release.
- The CreateCatalog API cannot create table buckets in Amazon S3.
- The SearchTables API cannot search S3 Tables.

Hive metadata store data sharing considerations and limitations

With AWS Glue Data Catalog metadata federation (Data Catalog federation), you can connect the Data Catalog to external metastores that store metadata for your Amazon S3 data, and securely manage data access permissions using AWS Lake Formation.

The following considerations and limitations apply to federated databases that are created from Hive databases:

Considerations

- **AWS SAM application support** – You're responsible for the availability of application resources that AWS SAM deploys (Amazon API Gateway and Lambda function). Make sure that the connection between the AWS Glue Data Catalog and the Hive metastore is working when users run queries.
- **Hive metastore version requirement** – You can create federated databases only using Apache Hive version 3 and above.
- **Mapped database requirement** – Every Hive database must be mapped to a new database in Lake Formation.

- **Database-level federation support** – You can connect to Hive metastore only at the database level.
- **Permissions on federated databases** – The permissions applied on a federated database or tables under a federated database persist even when a source table or a database is deleted. When the source database or table is recreated, you don't need to regrant the permissions. When a federated table with Lake Formation permissions is deleted at source, Lake Formation permissions are still visible, and you can revoke them if needed.

  If a user deletes a federated database, all of its corresponding permissions are lost. Recreating the same database with the same name, will not recover Lake Formation permissions. Users will have to setup new permissions again.
- **IAMAllowedPrincipal group permissions** on federated databases – Based on the `DataLakeSettings`, Lake Formation might set permissions to all databases and tables to a virtual group named `IAMAllowedPrincipal`. The `IAMAllowedPrincipal` refers to all IAM principals who have access to Data Catalog resources through IAM principal policies and AWS Glue resource policies. If these permissions exist on a database or a table, all principals are granted access to the database or table.

  However, Lake Formation doesn't allow `IAMAllowedPrincipal` permissions on tables under federated databases. When you create federated databases, make sure that you pass the `CreateTableDefaultPermissions` parameter as an empty list.

  For more information, see Changing the default settings for your data lake.
- **Joining tables in queries** – You can join Hive metastore tables with Data Catalog native tables to run queries.

Limitations

- **Limitation on syncing metadata between the AWS Glue Data Catalog and the Hive metastore** – After establishing the Hive metastore connection, you need to create a federated database to sync metadata in the Hive metastore with the AWS Glue Data Catalog. The tables under the federated database are synced at runtime when users run queries.
- **Limitation on creating new tables under a federated database** – You will not be able to create new tables under federated databases.
- **Data permission limitation** – Support for permissions on Hive metastore table views is not available.

Amazon Redshift data sharing limitations

AWS Lake Formation allows you to securely manage data in a datashare from Amazon Redshift. Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the AWS Cloud. Using the data sharing capability, Amazon Redshift helps you to share data across AWS accounts. For more information about Amazon Redshift data sharing, see Overview of data sharing in Amazon Redshift.

The following notes and restrictions apply to federated databases that are created from Amazon Redshift datashares:

- **Mapped database requirement** – Every Amazon Redshift datashare must be mapped to a new database in Lake Formation. This is required to maintain unique table names when the datashare objects representation is flattened in the Data Catalog database.
- **Limitation on creating new tables under a federated database** – You will not be able to create new tables under federated databases.
- **Permissions on the federated databases** – The permissions applied on a federated database or tables under a federated database persist even when a source table or a database is deleted. When the source database or table is recreated, you do not need to regrant the permissions. When a federated table with Lake Formation permissions is deleted at source, Lake Formation permissions will still be visible and you can revoke them if needed.

  If a user deletes a federated database, all its corresponding permissions are lost. Recreating the same database with the same name, will not recover Lake Formation permissions. Users will have to setup new permissions again.
- **IAMAllowedPrincipal group permissions on federated databases** – Based on the `DataLakeSettings`, Lake Formation might set permissions to all databases and tables to a virtual group named `IAMAllowedPrincipal`. The `IAMAllowedPrincipal` refers to all IAM principals who have access to Data Catalog resources through IAM principal policies and AWS Glue resource policies. If these permissions exist on a database or a table, all principals are granted access to the database or table.

  However, Lake Formation doesn't allow `IAMAllowedPrincipal` permissions on tables under federated databases. When you create federated databases, make sure that you pass the `CreateTableDefaultPermissions` parameter as an empty list.

  For more information, see Changing the default settings for your data lake.

- **Data filtering** – In Lake Formation, you can grant permissions on a table under a federated database with column-level and row-level filtering. However, you can't combine column-level and row-level filtering to restrict access at cell-level granularity on tables under federated databases.
- **Case sensitivity identifier** – Amazon Redshift datashare objects managed by Lake Formation, will support table names and column names only in lowercase. Don't turn on case sensitivity identifier for databases, tables, and columns in Amazon Redshift datashares, if they will be shared and managed using Lake Formation.
- **Query support** – You can query Amazon Redshift datashares managed by Lake Formation with Amazon Redshift. Athena doesn't support querying Amazon Redshift datashares managed by Lake Formation.

For more information on limitations when working with datashares in Amazon Redshift, see Limitations for data sharing in the Amazon Redshift Database Developer Guide.

IAM Identity Center integration limitations

With AWS IAM Identity Center, you can connect to identity providers (IdPs) and centrally manage access for users and groups across AWS analytics services. You can configure AWS Lake Formation as an enabled application in IAM Identity Center, and data lake administrators can grant fine-grained permissions to authorized users and groups on AWS Glue Data Catalog resources.

The following limitations apply to Lake Formation integration with IAM Identity Center:

- You can't assign IAM Identity Center users and groups as data lake administrators or read-only administrators in Lake Formation.

  IAM Identity Center users and groups can query encrypted Data Catalog resources if you are using an IAM role that AWS Glue can assume on your behalf for encrypting and decrypting the Data Catalog. AWS managed keys don't support trusted identity propagation.
- IAM Identity Center users and groups can only invoke API operations listed in the `AWSIAMIdentityCenterAllowListForIdentityContext` policy provided by IAM Identity Center.
- Lake Formation permits IAM roles from external accounts to act as carrier roles on behalf of IAM Identity Center users and groups for accessing Data Catalog resources, but permissions can only be granted on Data Catalog resources within the owning account. If you try to grant permissions to IAM Identity Centerusers and groups on Data Catalog resources in an external

account, Lake Formation throws the following error - "Cross-account grants are not supported for the principal."

- When using Lake Formation with IAM Identity Center, the application assignment configuration is set to `false` by default. If you modify this configuration directly through the IAM Identity Center API, you must then manage all application assignments manually using the API. Lake Formation does not automatically synchronize or manage assignment changes made outside of its standard workflows, which may impact access patterns and authorization flows within your data lake environment.

Lake Formation tag-based access control best practices and considerations

You can create, maintain, and assign LF-Tags to control access to Data Catalog databases, tables, and columns.

Consider the following best practices when using Lake Formation tag-based access control:

- All LF-Tags must be predefined before they can be assigned to Data Catalog resources or granted to principals.

  The data lake administrator can delegate tag management tasks by creating *LF-Tag creators* with the required IAM permissions. Data engineers and analysts decide on the characteristics and relationships for LF-Tags. The LF-Tag creators then creates and maintains the LF-Tags in Lake Formation.
- You can assign multiple LF-Tags to Data Catalog resources. Only one value for a particular key can be assigned to a particular resource.

  For example, you can assign `module=Orders`, `region=West`, `division=Consumer`, and so on to a database, table, or column. You can't assign `module=Orders,Customers`.
- You can't assign LF-Tags to resources when you create the resource. You can only add LF-Tags to existing resources.
- You can grant LF-Tag expressions, not just single LF-Tags, to a principal.

  A LF-Tag expression looks something like the following (in pseudo-code).

```
module=sales AND division=(consumer OR commercial)
```

A principal that is granted this LF-Tag expression can access only Data Catalog resources (databases, tables, and columns) that are assigned `module=sales` *and* either `division=consumer` or `division=commercial`. If you want the principal to be able to access resources that have `module=sales` *or* `division=commercial`, don't include both in the same grant. Make two grants, one for `module=sales` and one for `division=commercial`.

The simplest LF-Tag expression consists of just one LF-Tag, such as `module=sales`.

- A principal that is granted permissions on a LF-Tag with multiple values can access Data Catalog resources with either of those values. For example, if a user is granted a LF-Tag with key=`module` and values=`orders,customers`, the user has access to resources that are assigned either `module=orders` or `module=customers`.
- You need to have `Grant with LF-Tag expressions` permission to grant data permissions on Data Catalog resources by using the LF-TBAC method. The data lake administrator and the LF-Tag creator implicitly receive this permission. A principal that has the `Grant with LFTag expressions` permission can grant data permissions on the resources using:
  - the named resource method
  - the LF-TBAC method, but only using the same LF-Tag expression

  For example, assume that the data lake administrator makes the following grant (in pseudo-code).

  ```
  GRANT (SELECT ON TABLES) ON TAGS module=customers, region=west,south TO user1 WITH
   GRANT OPTION
  ```

  In this case, `user1` can grant `SELECT` on tables to other principals by using the LF-TBAC method, but only with the complete LF-Tag expression `module=customers, region=west,south`.
- If a principal is granted permissions on a resource with both the LF-TBAC method and the named resource method, the permissions that the principal has on the resource is the union of the permissions granted by both methods.
- Lake Formation supports granting `DESCRIBE` and `ASSOCIATE` on LF-Tags across accounts, and granting permissions on Data Catalog resources across accounts using the LF-TBAC method. In both cases, the principal is an AWS account ID.

> **Note**
>
> Lake Formation supports cross-account grants to organizations and organizational units using LF-TBAC method. To use this capability, you need to update **Cross account version settings** to **Version 3**.

For more information, see Cross-account data sharing in Lake Formation.

- Data Catalog resources created in one account can only be tagged using LF-Tags created in the same account. LF-Tags created in one account can't be associated with shared resources from another account.
- Using Lake Formation tag-based access control (LF-TBAC) to grant cross-account access to Data Catalog resources requires additions to the Data Catalog resource policy for your AWS account. For more information, see Prerequisites.
- LF-Tag keys and LF-Tag values can't exceed 50 characters in length.
- The maximum number of LF-Tags that can be assigned to a Data Catalog resource is 50.
- The following limits are soft limits:
  - The maximum number of LF-Tags that can be created is 10000.
  - The maximum number of values that can be defined for a LF-Tag is 1000.
- Tags keys and values are converted to all lower case when they are stored.
- Only one value for a LF-Tag can be assigned to a particular resource.
- If multiple LF-Tags are granted to a principal with a single grant, the principal can access only Data Catalog resources that have all of the LF-Tags.
- If a LF-Tag expression evaluation results in access to only a subset of table columns, but the Lake Formation permission granted when there is a match is one of the permissions that required full column access, namely `Alter`, `Drop`, `Insert`, or `Delete`, then none of those permissions is granted. Instead, only `Describe` is granted. If the granted permission is `All` (`Super`), then only `Select` and `Describe` are granted.
- Wildcards are not used with LF-Tags. To assign a LF-Tag to all columns of a table, you assign the LF-Tag to the table, and all columns in the table inherit the LF-Tag. To assign a LF-Tag to all tables in a database, you assign the LF-Tag to the database, and all tables in the database inherit that LF-Tag.
- You can create up to 1000 LF-Tag expressions in an account.

- You can use up to 50 LF-Tag expressions to grant permissions to a principal on the Data Catalog resources.
- When granting or revoking permissions on an inline LF-Tag expression, the size of the LF-Tag expression can not exceed 900 bytes. To grant permissions on larger LF-Tag expressions, use the saved LF-Tag expressions. For more information, see Creating LF-Tag expressions.
- To add LF-Tag to existing Redshift federated catalogs that were created before the general availability release of LF-Tag support for federated catalogs, you need to contact the AWS support team for assistance.

Attribute-based access control considerations, limitations, and supported regions

The following considerations and limitations apply to Attribute based access control (ABAC).

- ABAC doesn't support granting access using LF-Tag policies.
- Grantable permissions are not available with ABAC.
- ABAC doesn't support granting permissions to IAM Identity Center users.
- When using ABAC grants on a table in Lake Formation, Lake Formation doesn't grant `DESCRIBE` permissions to the parent database or catalog. This differs from non-ABAC scenarios, where Lake Formation provides implicit `DESCRIBE` permissions to parent resources.
- All principals with the `AmazonDataZoneProject` tag key are always treated as opted in to Lake Formation for all Data Catalog resources.
- ABAC supports only string attributes.

Row-level security policy limitations

The following limitations apply to row-level security (RLS) policies in AWS Lake Formation. For more information about RLS policies, see Row-level security policies in AWS Lake Formation.

Filter expression limitations

The following limitations apply to RLS policy filter expressions.

- Subqueries, joins, and references to external tables are not supported. All filtering logic must use columns within the target table.

- User-defined functions (UDFs) are not supported.
- Only static filter expressions are supported. Dynamic filters that reference user attributes (for example, `table.region = user.region`) are not supported. Create a separate policy for each distinct filter value.
- Column-to-column comparisons are not supported (for example, `column_a = column_b`). Compare columns to constant values instead.
- Aggregate functions (such as `SUM`, `COUNT`, `AVG`) and window functions are not supported.
- Every column referenced in the filter expression must be declared in `AttachmentColumns`.
- The total length of a filter expression must be less than 2048 characters.
- `IS [NOT] NULL` on partition columns is not supported.

For the full filter expression syntax and supported operators, see Filter expression syntax for RLS policies.

Attachment validation

When you attach RLS policies through Lake Formation tags, validation occurs at query time rather than at attachment time. For a matched table that does not contain a referenced column, the query returns an empty result for that table rather than failing, and no error is raised when the policy is attached.

Column names in RLS filter expressions are late binding: they are resolved against the table's columns at query time, not when you create or attach the policy. If you rename a column that a filter expression references, the policy no longer matches that column, which can cause queries to return an empty result. Review and update affected RLS policies before renaming columns.

Identity limitations

Only IAM principal attributes are supported for attribute-based access control (ABAC). IAM Identity Center attributes are not currently supported.

Operation limitations

RLS policies apply to `SELECT` operations only. They filter the rows a principal can read and do not filter or authorize write operations. Write operations such as `INSERT`, `UPDATE`, and `DELETE` are governed by the principal's separate Lake Formation table permissions and are not restricted by an RLS policy.

Interaction with data cell filters

RLS policies and data cell filters (DCF) can apply to the same table at the same time. When a principal is subject to both a DCF row filter and an RLS policy, Lake Formation combines the two row filters using union (OR) semantics: the principal sees every row that matches either filter. Because the filters are additive, using both together can only expand the rows a principal sees.

To consolidate DCF row filters into RLS policies, create the corresponding RLS policies first, then revoke the existing data cell filter row-level permissions. For a step-by-step migration workflow, see Best practices for RLS policies.

Troubleshooting Lake Formation

If you encounter issues when working with AWS Lake Formation, consult the topics in this section.

Topics

- General troubleshooting
- Troubleshooting cross-account access
- Troubleshooting blueprints and workflows
- Known issues for AWS Lake Formation
- Updated error message

General troubleshooting

Use the information here to help you diagnose and fix various Lake Formation issues.

Error: Insufficient Lake Formation permissions on <Amazon S3 location>

An attempt was made to create or alter a Data Catalog resource without data location permissions on the Amazon S3 location pointed to by the resource.

If a Data Catalog database or table points to an Amazon S3 location, when you grant the Lake Formation permissions `CREATE_TABLE` or `ALTER`, you must also grant the `DATA_LOCATION_ACCESS` permission on the location. If you are granting these permissions to external accounts or to organizations, you must include the grant option.

After these permissions are granted to an external account, the data lake administrator in that account must then grant the permissions to principals (users or roles) in the account. When granting the `DATA_LOCATION_ACCESS` permission that was received from another account, you must specify the catalog ID (AWS account ID) of the owner account. The owner account is the account that registered the location.

For more information, see Underlying data access control and Granting data location permissions.

Error: "Insufficient encryption key permissions for Glue API"

An attempt was made to grant Lake Formation permissions without AWS Identity and Access Management (IAM) permissions on the AWS KMS encryption key for an encrypted Data Catalog.

My Amazon Athena or Amazon Redshift query that uses manifests is failing

Lake Formation does not support queries that use manifests.

Error: "Insufficient Lake Formation permission(s): Required create tag on catalog"

The user/role must be a data lake administrator.

Error when deleting invalid data lake administrators

You should delete all invalid data lake administrators (deleted IAM roles that are defined as data lake administrators) simultaneously. If you try to delete invalid data lake administrators separately, Lake Formation throws invalid principal error.

Error: "Caller is temporarily not authorized to perform this operation on resource..."

An attempt was made to query a table using a resource link that may have been redirected. To avoid this error, your application should adopt the QuerySessionContext protocol. See Lake Formation workflow for application integration API operations for more information. Otherwise, please wait 24 hours before attempting to execute your query again.

Troubleshooting cross-account access

Use the information here to help you diagnose and fix cross-account access issues.

Topics

- I granted a cross-account Lake Formation permission but the recipient can't see the resource
- Principals in the recipient account can see the Data Catalog resource but can't access the underlying data

- Error: "Association failed because the caller was not authorized" when accepting a AWS RAM resource share invitation
- Error: "Not authorized to grant permissions for the resource"
- Error: "Access denied to retrieve AWS Organization information"
- Error: "Organization <organization-ID> not found"
- Error: "Insufficient Lake Formation permissions: Illegal combination"
- ConcurrentModificationException on grant/revoke requests to external accounts
- Error when using Amazon EMR to access data shared via cross-account

I granted a cross-account Lake Formation permission but the recipient can't see the resource

- Is the user in the recipient account a data lake administrator? Only data lake administrators can see the resource at the time of sharing.
- Are you sharing with an account external to your organization by using the named resource method? If so, the data lake administrator of the recipient account must accept a resource share invitation in AWS Resource Access Manager (AWS RAM).

  For more information, see the section called “Accepting an AWS RAM resource share invitation”.
- Are you using account-level (Data Catalog) resource policies in AWS Glue? If yes, then if you use the named resources method, you must include a special statement in the policy that authorizes AWS RAM to share policies on your behalf.

  For more information, see the section called “Managing cross-account permissions using both AWS Glue and Lake Formation”.
- Do you have the AWS Identity and Access Management (IAM) permissions required to grant cross-account access?

  For more information, see the section called “Prerequisites”.
- The resource that you've granted permissions on must not have any Lake Formation permissions granted to the `IAMAllowedPrincipals` group.
- Is there a `deny` statement on the resource in the account-level policy?

Principals in the recipient account can see the Data Catalog resource but can't access the underlying data

Principals in the recipient account must have the required AWS Identity and Access Management (IAM) permissions. For details, see Accessing the underlying data of a shared table.

Error: "Association failed because the caller was not authorized" when accepting a AWS RAM resource share invitation

After granting access to a resource to a different account, when the receiving account attempts to accept the resource share invitation, the action fails.

```
$ aws ram get-resource-share-associations --association-type PRINCIPAL --resource-
share-arns arn:aws:ram:aws-region:444444444444:resource-share/e1d1f4ba-xxxx-xxxx-xxxx-
xxxxxxxx5d8d
{
    "resourceShareAssociations": [
        {
            "resourceShareArn": "arn:aws:ram:aws-region:444444444444:resource-share/
e1d1f4ba-xxxx-xxxx-xxxx-xxxxxxxx5d8d
",
            "resourceShareName": "LakeFormation-MMCC0XQBH3Y",
            "associatedEntity": "5815803XXXXX",
            "associationType": "PRINCIPAL",
            "status": "FAILED",
            "statusMessage": "Association failed because the caller was not
 authorized.",
            "creationTime": "2021-07-12T02:20:10.267000+00:00",
            "lastUpdatedTime": "2021-07-12T02:20:51.830000+00:00",
            "external": true
        }
    ]
}
```

The error occurs because the `glue:PutResourcePolicy` is invoked by AWS Glue when the receiving account accepts the resource share invitation. To resolve the issue, allow the `glue:PutResourcePolicy` action by the assumed role used by the producer/grantor account.

Error: "Not authorized to grant permissions for the resource"

An attempt was made to grant cross-account permissions on a database or table that is owned by another account. When a database or table is shared with your account, as a data lake administrator, you can grant permissions on it only to users in your account.

Error: "Access denied to retrieve AWS Organization information"

Your account is an AWS Organizations management account and you do not have the required permissions to retrieve organization information, such as organizational units in the account.

For more information, see Required permissions for cross-account grants.

Error: "Organization <organization-ID> not found"

An attempt was made to share a resource with an organization, but sharing with organizations is not enabled. Enable resource sharing with organizations.

For more information, see Enable Sharing with AWS Organizations in the *AWS RAM User Guide*.

Error: "Insufficient Lake Formation permissions: Illegal combination"

A user shared a Data Catalog resource while Lake Formation permissions were granted to the `IAMAllowedPrincipals` group for the resource. The user must revoke all Lake Formation permissions from `IAMAllowedPrincipals` before sharing the resource.

ConcurrentModificationException on grant/revoke requests to external accounts

When users make multiple concurrent grant and/or revoke permission requests for a principal on LF-Tag policies, then Lake Formation throws ConcurrentModificationException. Users need to catch the exception and retry the failed the grant/revoke request. Using batch versions of the `GrantPermissions`/`RevokePermissions` API operations - BatchGrantPermissionsand BatchRevokePermissions alleviates this problem to an extent by reducing the number of concurrent grant/revoke requests.

Error when using Amazon EMR to access data shared via cross-account

When you use Amazon EMR to access data shared with you from another account, some Spark libraries will attempt to call `Glue:GetUserDefinedFunctions` API operation. Since versions

1 and 2 of the AWS RAM managed permissions does not support this action, you receive the following error message:

```
"ERROR: User: arn:aws:sts::012345678901:assumed-role/my-
spark-role/i-06ab8c2b59299508a is not authorized to perform:
glue:GetUserDefinedFunctions on resource: arn:exampleCatalogResource
because no resource-based policy allows the glue:GetUserDefinedFunctions
action"
```

To resolve this error, the data lake administrator who created the resource share must update the AWS RAM managed permissions attached to the resource share. Version 3 of the AWS RAM managed permissions allows principals to perform the `glue:GetUserDefinedFunctions` action.

If you create a new resource share, Lake Formation applies the latest version of the AWS RAM managed permission by default, and no action is required by you. To enable cross-account data access for existing resource shares, you need to update the AWS RAM managed permissions to version 3.

You can view the AWS RAM permissions assigned to resources shared with you in AWS RAM. The following permissions are included in version 3:

```
Databases
  AWSRAMPermissionGlueDatabaseReadWriteForCatalog
  AWSRAMPermissionGlueDatabaseReadWrite

Tables
  AWSRAMPermissionGlueTableReadWriteForCatalog
  AWSRAMPermissionGlueTableReadWriteForDatabase

AllTables
  AWSRAMPermissionGlueAllTablesReadWriteForCatalog
  AWSRAMPermissionGlueAllTablesReadWriteForDatabase
```

To update AWS RAM managed permissions version of existing resource shares

You (data lake administrator) can either update AWS RAM managed permissions to a newer version by following instructions in the *AWS RAM User Guide* or you can revoke all existing permissions for the resource type and regrant them. If you revoke permissions, AWS RAM deletes the AWS RAM resource share associated with the resource type. When you regrant permissions, AWS RAM creates new resource shares attaching the latest version of AWS RAM managed permissions.

Troubleshooting blueprints and workflows

Use the information here to help you diagnose and fix blueprint and workflow issues.

Topics

- My blueprint failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>"
- My workflow failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>"
- A crawler in my workflow failed with "Resource does not exist or requester is not authorized to access requested permissions"
- A crawler in my workflow failed with "An error occurred (AccessDeniedException) when calling the CreateTable operation..."

My blueprint failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>"

An attempt was made to create a blueprint by a user who does not have sufficient permissions to pass the chosen role.

Update the user's IAM policy to be able to pass the role, or ask the user to choose a different role with the required passrole permissions.

For more information, see the section called "Lake Formation personas and IAM permissions reference".

My workflow failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>"

The role that you specified for the workflow did not have an inline policy allowing the role to pass itself.

For more information, see the section called "(Optional) Create an IAM role for workflows".

A crawler in my workflow failed with "Resource does not exist or requester is not authorized to access requested permissions"

One possible cause is that the passed role did not have sufficient permissions to create a table in the target database. Grant the role the `CREATE_TABLE` permission on the database.

A crawler in my workflow failed with "An error occurred (AccessDeniedException) when calling the CreateTable operation..."

One possible cause is that the workflow role did not have data location permissions on the target storage location. Grant data location permissions to the role.

For more information, see the section called “`DATA_LOCATION_ACCESS`”.

Known issues for AWS Lake Formation

Review these known issues for AWS Lake Formation.

Topics

- Limitation on filtering of table metadata
- Issue with renaming an excluded column
- Issue with deleting columns in CSV tables
- Table partitions must be added under a common path
- Issue with creating a database during workflow creation
- Issue with deleting and then re-creating a user
- Data Catalog API operations do not update the value for the IsRegisteredWithLakeFormation parameter
- Lake Formation operations do not support AWS Glue Schema Registry

Limitation on filtering of table metadata

AWS Lake Formation column-level permissions can be used to restrict access to specific columns in a table. When a user retrieves metadata about the table using the console or an API like `glue:GetTable`, the column list in the table object contains only the fields to which they have access. It is important to understand the limitations of this metadata filtering.

Although Lake Formation makes available metadata about column permissions to integrated services, the actual filtering of columns in query responses is the responsibility of the integrated service. Lake Formation clients that support column-level filtering, including Amazon Athena, Amazon Redshift Spectrum, and Amazon EMR filter the data based on the column permissions registered with Lake Formation. Users won't be able to read any data to which they should not have access. Currently, AWS Glue ETL doesn't support column filtering.

> **Note**
>
> EMR clusters are not completely managed by AWS. Therefore, it's the responsibility of EMR administrators to properly secure the clusters to avoid unauthorized access to data.

Certain applications or formats might store additional metadata, including column names and types, in the `Parameters` map as table properties. These properties are returned unmodified and are accessible by any user with `SELECT` permission on any column.

For example, the Avro SerDe stores a JSON representation of the table schema in a table property named `avro.schema.literal`, which is available to all users with access to the table. We recommend that you avoid storing sensitive information in table properties and be aware that users can learn the complete schema of Avro format tables. This limitation is specific to the metadata about a table.

AWS Lake Formation removes any table property beginning with `spark.sql.sources.schema` when responding to a `glue:GetTable` or similar request if the caller does not have `SELECT` permissions on all columns in the table. This prevents users from gaining access to additional metadata about tables created with Apache Spark. When run on Amazon EMR, Apache Spark applications still can read these tables, but certain optimizations might not be applied, and case-sensitive column names are not supported. If the user has access to all columns in the table, Lake Formation returns the table unmodified with all table properties.

Issue with renaming an excluded column

If you use column-level permissions to exclude a column and then rename the column, the column is no longer excluded from queries, such as `SELECT *`.

Issue with deleting columns in CSV tables

If you create a Data Catalog table with the CSV format and then delete a column from the schema, queries could return erroneous data, and column-level permissions might not be adhered to.

Workaround: Create a new table instead.

Table partitions must be added under a common path

Lake Formation expects all partitions of a table to be under a common path that is set in the table's location field. When you use the crawler to add partitions to a catalog, this works seamlessly. But if you add partitions manually, and these partitions are not under the location set in the parent table, data access does not work.

Issue with creating a database during workflow creation

When creating a workflow from a blueprint using the Lake Formation console, you can create the target database if it doesn't exist. When you do so, the user who is signed in gets the `CREATE_TABLE` permission on the database that is created. However, the crawler that the workflow generates assumes the workflow's role as it tries to create a table. This fails because the role doesn't have the `CREATE_TABLE` permission on the database.

Workaround: If you create the database through the console during the workflow setup, before you run the workflow, you must give the role associated with the workflow the `CREATE_TABLE` permission on the database that you just created.

Issue with deleting and then re-creating a user

The following scenario results in erroneous Lake Formation permissions returned by `lakeformation:ListPermissions`:

1. Create a user and grant Lake Formation permissions.
2. Delete the user.
3. Re-create the user with the same name.

`ListPermissions` returns two entries, one for the old user and one for the new user. If you try to revoke permissions granted to the old user, the permissions are revoked from the new user.

Data Catalog API operations do not update the value for the `IsRegisteredWithLakeFormation` parameter

There is a known limitation that Data Catalog API operations such as `GetTables` and `SearchTables` do not update the value for the `IsRegisteredWithLakeFormation` parameter, and return the default, which is false. It is recommended to use the `GetTable` API to view the correct value for the `IsRegisteredWithLakeFormation` parameter.

Lake Formation operations do not support AWS Glue Schema Registry

Lake Formation operations do not support AWS Glue tables that contain a `SchemaReference` in the `StorageDescriptor` to be utilized in the Schema Registery.

Updated error message

AWS Lake Formation has updated the resource specific exceptions to general `EntityNotFound` error message for the following API operations to meet security and compliance objectives.

- RevokePermissions
- GrantPermissions
- GetResourceLFTags
- GetTable
- GetDatabase

AWS Lake Formation API

> **Note**
>
> Updated API Reference for the AWS Lake Formation service is now available.

Contents

- Permissions APIs
  - Operations
  - Data Types
- Data lake settings APIs
  - Operations
  - Data Types
- IAM Identity Center integration APIs
  - Operations
  - Data Types
- Hybrid access mode APIs
  - Operations
  - Data Types
- Credential vending APIs
  - Operations
  - Data Types
- Tagging APIs
  - Operations
  - Data Types
- Data filter APIs
  - Operations
  - Data types
- Common data types
  - ErrorDetail structure

- String patterns

Permissions APIs

The Permissions API section describes operations and data types that are required for granting and revoking permissions in AWS Lake Formation. See Lake Formation API Reference Guide for all AWS Lake Formation API operations and data types.

Operations

- GrantPermissions
- RevokePermissions
- BatchGrantPermissions
- BatchRevokePermissions
- GetEffectivePermissionsForPath
- ListPermissions
- GetDataLakePrincipal

Data Types

- Resource
- DatabaseResource
- TableResource
- TableWithColumnsResource
- DataCellsFilterResourcee
- DataLocationResource
- DataLakePrincipal
- PrincipalPermissions
- PrincipalResourcePermissions
- DetailsMap
- ColumnWildcard
- BatchPermissionsRequestEntry

- BatchPermissionsFailureEntry

Data lake settings APIs

This section contains the Data lake settings API operations and data types for managing the data lake administrators.

Operations

- GetDataLakeSettings
- PutDataLakeSettings

Data Types

- DataLakeSettings

IAM Identity Center integration APIs

This section contains the operations for creating and managing Lake Formation integration with IAM Identity Center.

Operations

- CreateLakeFormationIdentityCenterConfiguration
- DeleteLakeFormationIdentityCenterConfiguration
- DescribeLakeFormationIdentityCenterConfiguration
- UpdateLakeFormationIdentityCenterConfiguration

Data Types

- ExternalFilteringConfiguration

Hybrid access mode APIs

The Hybrid access mode API section describes operations and data types that are required for setting up hybrid access mode in AWS Lake Formation. See Lake Formation API Reference Guide for all AWS Lake Formation API operations and data types.

Operations

- CreateLakeFormationOptIn
- DeleteLakeFormationOptIn
- ListLakeFormationOptIns

Data Types

- Resource
- DatabaseResource
- TableResource
- Resource Info
- LakeFormationOptInsInfo
- DataLocationResource

Credential vending APIs

The Credential Vending API section describes the operations and data types related to working with the AWS Lake Formation service to vend credentials and to register and manage a data lake resource.

Operations

- RegisterResource
- DeregisterResource
- ListResources
- GetUnfilteredTableMetadata
- GetUnfilteredPartitionsMetadata

- GetTemporaryGluePartitionCredentials
- GetTemporaryGlueTableCredentials
- UpdateResource

Data Types

- FilterCondition
- RowFilter
- ResourceInfo

Tagging APIs

The Tagging API section describes the operations and data types related to an authorization strategy that defines a permissions model on attributes or key-value pair tags.

Operations

- GetLFTagExpression
- ListLFTagExpressions
- DeleteLFTagExpression
- UpdateLFTagExpression
- CreateLFTagExpression
- AddLFTagsToResource
- RemoveLFTagsFromResource
- GetResourceLFTags
- ListLFTags
- CreateLFTag
- GetLFTag
- UpdateLFTag
- DeleteLFTag
- SearchTablesByLFTags

- SearchDatabasesByLFTags

Data Types

- LFTagKeyResource
- LFTagPolicyResource
- TaggedTable
- TaggedDatabase
- LFTag
- LFTagPair
- LFTagError
- ColumnLFTag

Data filter APIs

The Data Filter APIs describe how to manage data cell filters in AWS Lake Formation.

Operations

- CreateDataCellsFilter
- DeleteDataCellsFilter
- ListDataCellsFilter
- GetDataCellsFilter
- UpdateDataCellsFilter

Data types

- DataCellsFilter
- RowFilter

Common data types

The Common Data Types describes miscellaneous common data types in AWS Lake Formation.

ErrorDetail structure

Contains details about an error.

Fields

- `ErrorCode` – UTF-8 string, not less than 1 or more than 255 bytes long, matching the Single-line string pattern.

  The code associated with this error.
- `ErrorMessage` – Description string, not more than 2048 bytes long, matching the URI address multi-line string pattern.

  A message describing the error.

String patterns

The API uses the following regular expressions to define what is valid content for various string parameters and members:

- Single-line string pattern – `"[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*"`
- URI address multi-line string pattern – `"[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\r\n\t]*"`
- Custom string pattern #3 – `"^\w+\.\w+\.\w+$"`
- Custom string pattern #4 – `"^\w+\.\w+$"`
- Custom string pattern #5 – `"arn:aws:iam::[0-9]*:role/.*"`
- Custom string pattern #6 – `"arn:aws:iam::[0-9]*:user/.*"`
- Custom string pattern #7 – `"arn:aws:iam::[0-9]*:group/.*"`
- Custom string pattern #8 – `"arn:aws:iam::[0-9]*:saml-provider/.*"`
- Custom string pattern #9 – `"^([\p{L}\p{Z}\p{N}_.:\/=+\-@%]*)$"`
- Custom string pattern #10 – `"^([\p{L}\p{Z}\p{N}_.:\*\/=+\-@%]*)$"`
- Custom string pattern #11 – `"[\p{L}\p{N}\p{P}]*"`

Supported Regions

This section has information on supported AWS Regions and functionality for Lake Formation.

General availability

For the AWS Regions supported by AWS Lake Formation, see List of AWS services available by Region.

For a list of the Lake Formation service endpoints for each Region and the Lake Formation service quotas, see AWS Lake Formation endpoints and quotas.

AWS GovCloud (US)

For an overview of differences between AWS GovCloud (US) Region and standard AWS Regions, see How AWS Lake Formation differs for AWS GovCloud (US).

Transactions and storage optimization

The governed tables, transaction support, and storage optimizations features for Lake Formation are available in the following AWS Regions:

Region name	Region parameter	Endpoint
US East (N. Virginia)	us-east-1	lakeformation.us-east-1.amazonaws.com lakeformation-fips.us-east-1.amazonaws.com
US East (Ohio)	us-east-2	lakeformation.us-east-2.amazonaws.com lakeformation-fips.us-east-2.amazonaws.com
US West (Oregon)	us-west-2	lakeformation.us-west-2.amazonaws.com

Region name	Region parameter	Endpoint
		lakeformation-fips.us-west-2.amazonaws.com
Asia Pacific (Mumbai)	ap-south-1	lakeformation.ap-south-1.amazonaws.com
Asia Pacific (Seoul)	ap-northeast-2	lakeformation.ap-northeast-2.amazonaws.com
Asia Pacific (Singapore)	ap-southeast-1	lakeformation.ap-southeast-1.amazonaws.com
Asia Pacific (Sydney)	ap-southeast-2	lakeformation.ap-southeast-2.amazonaws.com
Asia Pacific (Tokyo)	ap-northeast-1	lakeformation.ap-northeast-1.amazonaws.com
Europe (Frankfurt)	eu-central-1	lakeformation.eu-central-1.amazonaws.com
Europe (Ireland)	eu-west-1	lakeformation.eu-west-1.amazonaws.com
Europe (London)	eu-west-2	lakeformation.eu-west-2.amazonaws.com
Europe (Stockholm)	eu-north-1	lakeformation.eu-north-1.amazonaws.com
Canada (Central)	ca-central-1	lakeformation.ca-central-1.amazonaws.com

Region name	Region parameter	Endpoint
South America (São Paulo)	sa-east-1	lakeformation.sa-east-1.amazonaws.com

Document history for AWS Lake Formation

The following table describes important changes to the documentation for AWS Lake Formation.

Change	Description	Date
Updated documentation for attribute-based access control	Lake Formation allows you to grant permissions to principals with matching attributes on Data Catalog resources. For more information, see Attribute-based access control.	April 17, 2025
Amazon S3 Tables integration with AWS Lake Formation and AWS Glue Data Catalog	You can now integrate and catalog S3 Tables as AWS Glue Data Catalog objects and register the catalog as a Lake Formation data location. For more information, see Creating an Amazon S3 Tables catalog in the AWS Glue Data Catalog.	March 13, 2025
Lake Formation updated `AWSLakeFormationCrossAccountManager` policy.	Lake Formation enhanced the AWSLakeFormationCrossAccountManager policy by replacing the `StringLike` condition operator with the `ArnLike` operator that allows IAM to perform the ARN format check.	January 25, 2025
Updated policy change	Documented the change to the AWSLakeFormationDataAdmin policy.	December 3, 2024

Multi-catalog updates	AWS Glue Data Catalog allows you to create federated catalogs, and unify data across Amazon S3 data lakes and Amazon Redshift data warehouses as well as integrate data from your operational databases such as Amazon DynamoDB, and third-party data sources such as Snowflake, MySQL, and so on. For more information, see Bringing your data into the AWS Glue Data Catalog.	December 3, 2024
Updated documentation for LF-Tag expressions	You can save LF-Tag expressions and reuse them to grant permissions on the Data Catalog resources. For more information, see Managing LF-Tag expressions.	November 7, 2024
Updated documentation for Data Catalog views	You can create views in the AWS Glue Data Catalog using AWS Glue APIs in addition to DDLs using Amazon Athena and Amazon Redshift. For more information, see Building Data Catalog views.	August 7, 2024

Added documentation for auditable credential vending	Lake Formation allows you to include the IAM Identity Center user's context in the CloudTrail events, and then track the users that access your resources. For more information, see Including IAM Identity Center user context in CloudTrail logs.	July 14, 2024
Updated policy change	Documented the change (added statement IDs and removed redundant permissions) to the AWSLakeFormationCrossAccountManager and AWSLakeFormationDataAdmin policies.	March 14, 2024
Updated setting up Lake Formation	Updated the steps in the Set up AWS Lake Formation section.	February 7, 2024
Updated policy change	Added new permissions to the service-linked role's inline policy. For more information, see Using service-linked roles for Lake Formation.	February 7, 2024
Updated policy change	Documented the change to the LakeFormationDataAccessServiceRolePolicy policy.	February 2, 2024
Consolidated Lake Formation limitations	Created a unified section for Lake Formation limitations and considerations. For more information, see Lake Formation limitations.	December 15, 2023

Added documentation for Iceberg compaction	For better read performance by AWS analytics services such as Athena and Amazon EMR, and AWS Glue ETL jobs, AWS Glue Data Catalog provides managed compaction (a process that compacts small Amazon S3 objects into larger objects) for Iceberg tables in the Data Catalog. For more information, see Optimizing Iceberg tables.	November 25, 2023
Added documentation for IAM Identity Center integration	IAM Identity Center integrations allows users and groups to access Data Catalog resources enforcing Lake Formation permissions. For more information, see IAM Identity Center integration.	November 25, 2023
Added documentation for Data Catalog views	You can create views in the AWS Glue Data Catalog that references up to 10 tables using SQL editors for Amazon Athena or Amazon Redshift. For more information, see Creating views.	November 25, 2023
Updated the policy change	Documented the change to the AWSLakeFormationCrossAccountManager policy.	October 25, 2023

Added documentation for hybrid access mode	Hybrid access mode provides the flexibility to selective ly enable Lake Formation permissions for databases and tables in your AWS Glue Data Catalog. With hybrid access mode, you now have an incremental path that allows you to set Lake Formation permissions for a specific set of users without interrupt ing the permission policies of other existing users or workloads. For more informati on, see Hybrid access mode.	September 26, 2023
Added documentation for creating Apache Iceberg tables	You can now create Apache Iceberg tables that use the Apache Parquet data format in the AWS Glue Data Catalog with data residing in Amazon S3. For more information, see Creating Iceberg tables.	August 16, 2023

Added documentation for cross-Region data access	Lake Formation supports querying Data Catalog tables across AWS Regions. You can access data in a Region from other Regions using Athena, Amazon EMR, and run AWS Glue ETL by creating resource links in other Regions pointing to the source databases and tables. You can connect the Data Catalog to external metastores that store metadata for your Amazon S3 data, and securely manage data access permissions using AWS Lake Formation. For more information, see Accessing tables across Regions.	June 30, 2023
Re-organized contents	Re-organized chapters in the guide to match Lake Formation user journey.	May 15, 2023
Added documentation for HMS federation	You can connect the Data Catalog to external metastores that store metadata for your Amazon S3 data, and securely manage data access permissions using AWS Lake Formation. For more information, see Managing permissions on datasets that use external metastores.	April 15, 2023

Added documentation for Amazon Redshift data sharing	You can now securely manage data in a datashare from Amazon Redshift using Lake Formation permissions. Lake Formation supports licensing access to your data through AWS Data Exchange. For more information, see Data sharing in AWS Lake Formation.	November 30, 2022
Support for cross-account data sharing directly with principals	Added information about sharing data directly with IAM principals in another account. For more information see Cross-account data sharing in AWS Lake Formation.	November 10, 2022
Support for AWS RAM enabled data sharing using TBAC	Added information about The LF-TBAC method of granting Data Catalog permissions use AWS Resource Access Manager for cross-account grants.	November 10, 2022
Added a section on working with other services	Added information on how AWS services such as Athena, AWS Glue, Redshift Spectrum, and Amazon EMR can use Lake Formation to securely access data in Amazon S3 locations registered with Lake Formation. For more information see Working with other AWS services.	November 10, 2022

???	Added information on troubleshooting an error when using Amazon EMR to access cross-account data. For more information, see Error when using Amazon EMR to access data shared via cross-account.	November 7, 2022
Updates to cross-account resource share	Added a description for how cross-account resource shares work in Lake Formation. Documented the change to the AWSLakeFormationCrossAccountManager policy.	May 6, 2022
New tutorials	Added new tutorials for creating governed tables, securing data lakes, and sharing data lakes. For more details, see Get started section.	April 20, 2022
New Lake Formation landing page	Updated the Lake Formation landing page to include links for tutorials that provide step-by-step instructions on how to build a data lake, ingest data, share, and secure data lakes using Lake Formation.	April 20, 2022

Support for credential vending	Added information about credential vending, which supports Lake Formation to allow third-party services to integrate with Lake Formation by using credentia l vending API operations. For more information, see How credential vending works in Lake Formation.	February 28, 2022
Support for governed tables and advanced data filtering	Added information about governed tables, which support ACID transactions, automatic data compactio n, and time-travel queries. Added information about creating data filters to support for column-level security, row-level security, and cell-level security. For more information, see Governed Tables in Lake Formation and Data Filtering and Cell-Level Security in Lake Formation.	November 30, 2021

Support for VPC interface endpoints	Added information about creating a virtual private cloud (VPC) interface endpoint for Lake Formation , so that communication between your VPC and Lake Formation is conducted entirely and securely within the AWS network. For more information, see Using Lake Formation with VPC Endpoints .	October 11, 2021
Support for VPC endpoint policies	Added information about support for Virtual Private Cloud (VPC) endpoint policies in Lake Formation. For more information, see Using Lake Formation with VPC Endpoints.	October 11, 2021
Support for tag-based access control	Lake Formation tag-based access control provides a new, more scalable way to manage access to Data Catalog resources and underlying data by using LF-Tags. For more information, see Lake Formation Tag-Based Access Control.	May 7, 2021

New opt-in requirement for data filtering on Amazon EMR.	Added information about the requirement to opt in to allow Amazon EMR to filter data that is managed by Lake Formation. For more information, see Allow Data Filtering on Amazon EMR.	October 9, 2020
Support for granting full cross-account permissions on Data Catalog databases	Added information about granting full Lake Formation permissions on Data Catalog databases across AWS accounts, including `CREATE_TABLE` . For more information, see Sharing Data Catalog Databases.	October 1, 2020
Support for Amazon Athena users authenticating through SAML.	Added information about support for Athena users who connect through the JDBC or ODBC driver and authentic ate through SAML identity providers such as Okta and Microsoft Active Directory Federation Service (AD FS). For more information, see AWS Service Integrations with Lake Formation.	September 30, 2020
Support for cross-account access with an encrypted Data Catalog	Added information about granting cross-account permissions when the Data Catalog is encrypted. For more information, see Cross-Account Access Prerequisites.	July 30, 2020

Support for cross-account access to the data lake	Added information about granting AWS Lake Formation permissions on Data Catalog databases and tables to external AWS accounts and organizations, and about accessing Data Catalog objects shared from external accounts. For more informati on, see Cross-Account Access.	July 7, 2020
Integration with Amazon Quick	Added information about how to grant Lake Formation permissions to Quick Enterprise Edition users so that they may access datasets residing in registered Amazon S3 locations. For more information, see Granting Data Catalog Permissions.	June 29, 2020
Updates to setting up and Getting Started chapters	Reorganized and improved the Setting Up and Getting Started chapters. Updated the recommended AWS Identity and Access Management (IAM) permissions for the data lake administrator.	February 27, 2020

Support for AWS Key Management Service	Added information about how Lake Formation support for AWS Key Management Service (AWS KMS) simplifies setting up integrated services to read and write encrypted data in registered Amazon Simple Storage Service (Amazon S3) locations. Added informati on about how to register Amazon S3 locations that are encrypted with AWS KMS keys. For more information, see the section called "Adding an Amazon S3 location to your data lake".	February 27, 2020
Updates to blueprints and data lake administrator IAM policies	Clarified input parameter s for incremental database blueprints. Updated the IAM policies required for a data lake administrator.	December 20, 2019
Security chapter rewrite and upgrade chapter revisions	Improved the security and upgrading chapters.	October 29, 2019
Super permission replaces All permission	Updated the Security and Upgrading chapters to reflect the replacement of the permission `All` with `Super`.	October 10, 2019

Additions, corrections, and clarifications	Made additions, corrections, and clarifications based on feedback. Revised the security chapter. Updated the Security and Upgrading chapters to reflect the replacement of the group `Everyone` with `IAMAllowedPrincipals` .	September 11, 2019
New guide	This is the initial release of the *AWS Lake Formation Developer Guide*.	August 8, 2019

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.