

---

# AWS Lake Formation

## Developer Guide



## **AWS Lake Formation: Developer Guide**

Copyright © 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
AWS Service Integrations with Lake Formation .....	1
Getting Started with Lake Formation .....	2
How It Works .....	3
Lake Formation Terminology .....	3
Data Lake .....	3
Data Access .....	4
Blueprint .....	4
Workflow .....	4
Data Catalog .....	4
Underlying Data .....	4
Principal .....	4
Data Lake Administrator .....	5
Lake Formation Components .....	5
Lake Formation Console .....	5
Lake Formation API and Command Line Interface .....	5
Other AWS Services .....	5
Setting Up AWS Lake Formation .....	6
Sign Up for AWS .....	6
Create an Administrator IAM User .....	6
Create an IAM Role for Workflows .....	7
Create a Data Lake Administrator .....	8
Change Data Catalog Settings .....	11
(Optional) Allow Data Filtering on Amazon EMR Clusters .....	12
(Optional) Grant Access to the Data Catalog Encryption Key .....	12
Getting Started .....	13
Tutorial: Creating a Data Lake from an AWS CloudTrail Source .....	13
About the Personas in This Tutorial .....	14
AWS CloudTrail Tutorial Prerequisites .....	14
Step 1: Create the IAM User to Be the Data Analyst .....	14
Step 2: Add Permissions to Read AWS CloudTrail Logs to the Workflow Role .....	15
Step 3: Create an Amazon S3 Bucket for the Data Lake .....	16
Step 4: Register an Amazon S3 Path .....	16
Step 5: Grant Data Location Permissions .....	16
Step 6: Create a Database in the Data Catalog .....	16
Step 7: Grant Data Permissions .....	17
Step 8: Use a Blueprint to Create a Workflow .....	18
Step 9: Run the Workflow .....	19
Step 10: Grant SELECT on the Tables .....	20
Step 11: Query the Data Lake Using Amazon Athena .....	20
Tutorial: Creating a Data Lake from a JDBC Source .....	21
Personas in This Tutorial .....	21
Prerequisites .....	22
Step 1: Create the IAM User to Be the Data Analyst .....	22
Step 2: Create a Connection in AWS Glue .....	23
Step 3: Create an Amazon S3 Bucket for the Data Lake .....	23
Step 4: Register an Amazon S3 Path .....	23
Step 5: Grant Data Location Permissions .....	24
Step 6: Create a Database in the Data Catalog .....	24
Step 7: Grant Data Permissions .....	24
Step 8: Use a Blueprint to Create a Workflow .....	24
Step 9: Run the Workflow .....	25
Step 10: Grant SELECT on the Tables .....	26
Step 11: Query the Data Lake Using Amazon Athena .....	26

Step 12: Query the Data in the Data Lake Using Amazon Redshift Spectrum .....	27
Step 13: Grant or Revoke Lake Formation Permissions Using Amazon Redshift Spectrum .....	31
Adding an Amazon S3 Location to Your Data Lake .....	32
Requirements for Roles Used to Register Locations .....	32
Registering an Amazon S3 Location .....	34
Registering an Encrypted Amazon S3 Location .....	35
Registering an Amazon S3 Location in Another AWS Account .....	37
Registering an Encrypted Amazon S3 Location Across AWS Accounts .....	39
Deregistering an Amazon S3 Location .....	41
Managing Data Catalog Tables and Databases .....	43
Creating a Database .....	43
Creating Tables .....	44
Searching for Tables .....	44
Sharing Data Catalog Tables and Databases Across Accounts .....	45
Accessing and Viewing Shared Data Catalog Tables and Databases .....	45
Accepting an AWS RAM Resource Share Invitation .....	46
Viewing Shared Data Catalog Tables and Databases .....	48
Creating Resource Links .....	49
How Resource Links Work .....	49
Creating a Resource Link to a Shared Table .....	51
Creating a Resource Link to a Shared Database .....	53
Resource Link Handling in AWS Glue APIs .....	54
Importing Data Using Workflows .....	58
Blueprints and Workflows .....	58
Creating a workflow .....	59
Running a workflow .....	61
Security .....	62
Data Protection .....	62
Encryption at Rest .....	63
Infrastructure Security .....	63
Security and Access Control .....	64
Lake Formation Access Control Overview .....	64
Managing Policy Tags for Metadata Access Control .....	92
Granting Lake Formation Permissions .....	104
Viewing Database and Table Permissions .....	144
AWS Managed Policies for Lake Formation .....	147
Changing the Default Security Settings for Your Data Lake .....	147
Permissions Example Scenario .....	149
Security Event Logging in AWS Lake Formation .....	150
Using Service-Linked Roles .....	150
Service-Linked Role Permissions for Lake Formation .....	151
AWS Glue Features in Lake Formation .....	152
Logging AWS Lake Formation API Calls Using AWS CloudTrail .....	153
Lake Formation Information in CloudTrail .....	153
Understanding Lake Formation Events .....	154
Upgrading AWS Glue Data Permissions to the Lake Formation Model .....	156
About Upgrading to the Lake Formation Permissions Model .....	156
Step 1: List Existing Permissions .....	157
Using the API .....	157
Using the AWS Management Console .....	158
Using AWS CloudTrail .....	158
Step 2: Set up Lake Formation Permissions .....	158
Step 3: Give Users IAM Permissions .....	159
Step 4: Switch to the Lake Formation Permissions Model .....	159
Verify Lake Formation Permissions .....	159
Secure Existing Data Catalog Resources .....	160
Turn On Lake Formation Permissions for Your Amazon S3 Location .....	161

Step 5: Secure New Data Catalog Resources .....	161
Step 6: Give Users a New IAM Policy .....	162
Step 7: Clean Up Existing IAM Policies .....	162
AWS Lake Formation API .....	164
Permissions .....	165
— data types — .....	165
Resource .....	166
DatabaseResource .....	166
TableResource .....	167
TableWithColumnsResource .....	167
DataLocationResource .....	168
DataLakePrincipal .....	168
ResourcePermissions .....	168
ResourcePermissionsError .....	168
PrincipalResourcePermissions .....	169
DetailsMap .....	169
PrincipalResourcePermissionsError .....	169
ColumnWildcard .....	170
BatchPermissionsRequestEntry .....	170
BatchPermissionsFailureEntry .....	170
PrincipalPermissions .....	170
— operations — .....	171
GrantPermissions (grant_permissions) .....	171
RevokePermissions (revoke_permissions) .....	172
BatchGrantPermissions (batch_grant_permissions) .....	172
BatchRevokePermissions (batch_revoke_permissions) .....	173
GetEffectivePermissionsForPath (get_effective_permissions_for_path) .....	174
ListPermissions (list_permissions) .....	174
Data Lake Settings .....	175
— data types — .....	175
DataLakeSettings .....	176
— operations — .....	177
GetDataLakeSettings (get_data_lake_settings) .....	177
PutDataLakeSettings (put_data_lake_settings) .....	177
Credential Vending .....	178
— data types — .....	178
FilterCondition .....	178
ColumnNames .....	179
ResourceInfo .....	179
— operations — .....	179
RegisterResource (register_resource) .....	179
DeregisterResource (deregister_resource) .....	180
ListResources (list_resources) .....	180
Tagging .....	181
— data types — .....	181
LFTagKeyResource .....	181
LFTagPolicyResource .....	182
TaggedTable .....	182
TaggedDatabase .....	183
LFTag .....	183
LFTagPair .....	183
TagError .....	183
ColumnLFTag .....	184
— operations — .....	184
AddLFTagsToResource (add_lf_tags_to_resource) .....	184
RemoveLFTagsFromResource (remove_lf_tags_from_resource) .....	185
GetResourceLFTags (get_resource_lf_tags) .....	186

ListLFTags (list_lf_tags) .....	186
CreateLFTag (create_lf_tag) .....	187
GetLFTag (get_lf_tag) .....	188
UpdateLFTag (update_lf_tag) .....	189
DeleteLFTag (delete_lf_tag) .....	189
SearchTablesByLFTags (search_tables_by_lf_tags) .....	190
SearchDatabasesByLFTags (search_databases_by_lf_tags) .....	191
Common Data Types .....	192
ErrorDetail .....	192
String Patterns .....	192
Lake Formation Personas and IAM Permissions Reference .....	193
AWS Lake Formation Personas .....	193
Personas Suggested Permissions .....	193
Data Lake Administrator Permissions .....	194
Data Engineer Permissions .....	195
Data Analyst Permissions .....	197
Workflow Role Permissions .....	197
Troubleshooting Lake Formation .....	199
General Troubleshooting .....	199
Error: Insufficient Lake Formation permissions on <Amazon S3 location> .....	199
Error: "Insufficient encryption key permissions for Glue API" .....	199
My Amazon Athena or Amazon Redshift query that uses manifests is failing .....	199
Troubleshooting Cross-Account Access .....	199
I granted a cross-account Lake Formation permission but the recipient can't see the resource ....	200
Principals in the recipient account can see the Data Catalog resource but can't access the	
underlying data .....	200
Error: "Not authorized to grant permissions for the resource" .....	200
Error: "Access denied to retrieve AWS Organization information" .....	201
Error: "Organization <organization-ID> not found" .....	201
Error: "Insufficient Lake Formation permissions: Illegal combination" .....	201
Troubleshooting Blueprints and Workflows .....	201
My blueprint failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on	
resource: <role-ARN>" .....	201
My workflow failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on	
resource: <role-ARN>" .....	202
A crawler in my workflow failed with "Resource does not exist or requester is not authorized to	
access requested permissions" .....	202
A crawler in my workflow failed with "An error occurred (AccessDeniedException) when calling	
the CreateTable operation..." .....	202
Known Issues for AWS Lake Formation .....	203
Limitation on Filtering of Table Metadata .....	203
Issue with Renaming an Excluded Column .....	204
Issue with Deleting Columns in CSV Tables .....	204
Table Partitions Must Be Added Under a Common Path .....	204
Issue with Creating a Database during Workflow Creation .....	204
Issue with Deleting and Then Re-Creating a User .....	204
Document History .....	205
AWS glossary .....	208

# What Is AWS Lake Formation?

Welcome to the AWS Lake Formation Developer Guide.

AWS Lake Formation is a fully managed service that makes it easier for you to build, secure, and manage data lakes. Lake Formation simplifies and automates many of the complex manual steps that are usually required to create data lakes. These steps include collecting, cleansing, moving, and cataloging data, and securely making that data available for analytics and machine learning. You point Lake Formation at your data sources, and Lake Formation crawls those sources and moves the data into your new Amazon Simple Storage Service (Amazon S3) data lake.

Lake Formation provides its own permissions model that augments the AWS Identity and Access Management (IAM) permissions model. This centrally defined permissions model enables fine-grained access to data stored in data lakes through a simple grant/revoke mechanism.

Lake Formation permissions are enforced at the table and column level across the full portfolio of AWS analytics and machine learning services.

## Topics

- [AWS Service Integrations with Lake Formation \(p. 1\)](#)
- [Getting Started with Lake Formation \(p. 2\)](#)

## AWS Service Integrations with Lake Formation

The following AWS services integrate with AWS Lake Formation and honor Lake Formation permissions.

AWS Service	How Integrated
<a href="#">AWS Glue</a>	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. AWS Glue does not support Lake Formation column permissions.
<a href="#">Amazon Athena</a>	<p>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.</p> <p>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). For more information, see <a href="#">Using Lake Formation and the Athena JDBC and ODBC Drivers for Federated Access to Athena</a> in the <i>Amazon Athena User Guide</i>.</p>
<a href="#">Amazon Redshift Spectrum</a>	<p>When Amazon Redshift users create an external schema on a database in the AWS Glue catalog, they can query only the tables and columns in that schema on which they have Lake Formation permissions.</p> <p>Queries using manifests are not supported.</p>

AWS Service	How Integrated
<a href="#">Amazon QuickSight Enterprise Edition</a>	When an Amazon QuickSight Enterprise Edition user queries a dataset in an Amazon S3 location that is registered with Lake Formation, the user must have the Lake Formation <code>SELECT</code> permission on the data.
<a href="#">Amazon EMR</a>	Lake Formation permissions are enforced when Apache Spark applications are submitted using Apache Zeppelin or EMR Notebooks.

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.

## Getting Started with Lake Formation

We recommend that you start with the following sections:

- [AWS Lake Formation: How It Works \(p. 3\)](#) — Learn about essential terminology and how the various components interact.
- [Setting Up AWS Lake Formation \(p. 6\)](#) — Get information about prerequisites, and complete important setup tasks.
- [Getting Started with AWS Lake Formation \(p. 13\)](#) — Follow step-by-step tutorials to learn how to use Lake Formation.
- [Security in AWS Lake Formation \(p. 62\)](#) — Understand how you can help secure access to data in Lake Formation.

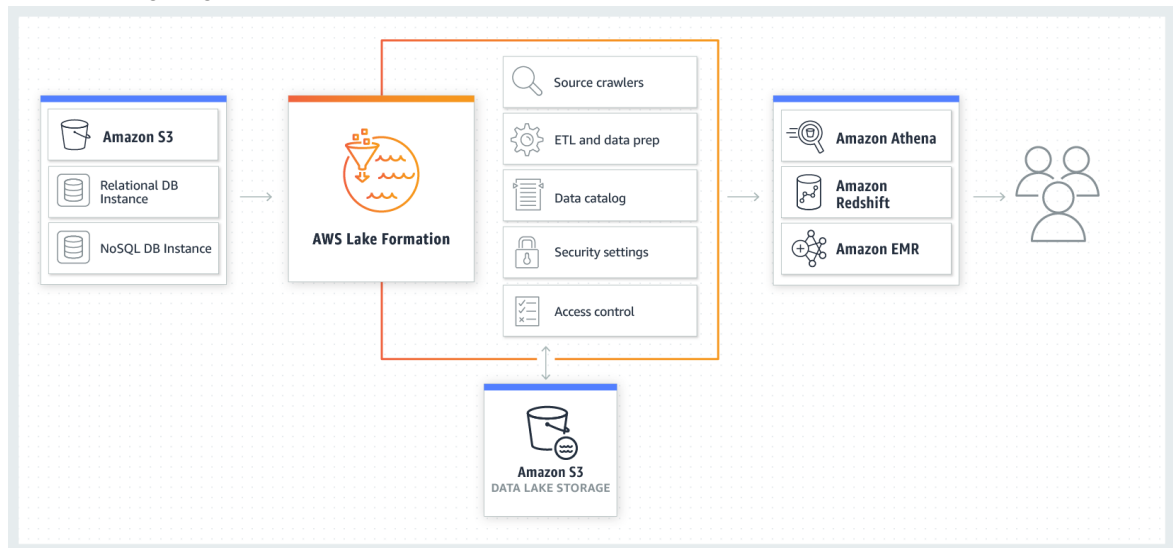


# AWS Lake Formation: How It Works

AWS Lake Formation makes it easier for you to build, secure, and manage data lakes. Lake Formation helps you do the following, either directly or through other AWS services:

- Register the Amazon Simple Storage Service (Amazon S3) buckets and paths where your data lake will reside.
- Orchestrate data flows that ingest, cleanse, transform, and organize the raw data.
- Create and manage a Data Catalog containing metadata about data sources and data in the data lake.
- Define granular data access policies to the metadata and data through a grant/revoke permissions model.

The following diagram illustrates how data is loaded and secured in Lake Formation.



As the diagram shows, Lake Formation manages AWS Glue crawlers, AWS Glue ETL jobs, the Data Catalog, security settings, and access control. After the data is securely stored in the data lake, users can access the data through their choice of analytics services, including Amazon Athena, Amazon Redshift, and Amazon EMR.

## Topics

- [Lake Formation Terminology \(p. 3\)](#)
- [Lake Formation Components \(p. 5\)](#)

## 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 analytics 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.

## 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](#) (p. 106). For information about designating a user as a data lake administrator, see [Create a Data Lake Administrator](#) (p. 8).

## 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.

# Setting Up AWS Lake Formation

Complete the following tasks to get set up to use Lake Formation:

1. [Sign Up for AWS](#) (p. 6)
2. [Create an Administrator IAM User](#) (p. 6)
3. [Create an IAM Role for Workflows](#) (p. 7)
4. [Create a Data Lake Administrator](#) (p. 8)
5. [Change Data Catalog Settings](#) (p. 11)
6. the section called “(Optional) Allow Data Filtering on Amazon EMR Clusters” (p. 12)
7. the section called “(Optional) Grant Access to the Data Catalog Encryption Key” (p. 12)

## Sign Up for AWS

When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including Lake Formation. You are charged only for the services that you use.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

### To create an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Note your AWS account number, because you'll need it for the next task.

## Create an Administrator IAM User

Services in AWS, such as Lake Formation, require that you provide credentials when you access them, so that the service can determine whether you have permission to access its resources. We don't recommend that you access AWS using the credentials for your AWS account. Instead, we recommend that you use AWS Identity and Access Management (IAM). You can create an IAM user, and then add the user to an IAM group with administrative permissions, or grant this user administrative permissions. You can then access AWS using the credentials for the IAM user.

If you signed up for AWS but have not created an administrative IAM user for yourself, you can create one using the IAM console. If you aren't familiar with using the console, see [Working with the AWS Management Console](#) for an overview.

### To create an administrator user for yourself and add the user to an administrators group (console)

1. Sign in to the [IAM console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

### Note

We strongly recommend that you adhere to the best practice of using the **Administrator** IAM user that follows and securely lock away the root user credentials. Sign in as the root user only to perform a few [account and service management tasks](#).

2. In the navigation pane, choose **Users** and then choose **Add user**.
3. For **User name**, enter **Administrator**.
4. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.
5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
6. Choose **Next: Permissions**.
7. Under **Set permissions**, choose **Add user to group**.
8. Choose **Create group**.
9. In the **Create group** dialog box, for **Group name** enter **Administrators**.
10. Choose **Filter policies**, and then select **AWS managed - job function** to filter the table contents.
11. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.

### Note

You must activate IAM user and role access to Billing before you can use the **AdministratorAccess** permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in [step 1 of the tutorial about delegating access to the billing console](#).

12. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
13. Choose **Next: Tags**.
14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see [Tagging IAM entities](#) in the *IAM User Guide*.
15. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see [Access management](#) and [Example policies](#).

## Create an IAM Role for Workflows

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. 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 the IAM administrator user that you created in [Create an Administrator IAM User \(p. 6\)](#) or as an IAM 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:Permissions**.
4. Search for the **AWSGlueServiceRole** managed policy, and select the check box next to the policy name in the list. Then complete the **Create role** wizard, naming the role `LakeFormationWorkflowRole`. To finish, choose **Create role**.
5. Back on the **Roles** page, search for `LakeFormationWorkflowRole` and choose the role name.
6. On the role **Summary** page, under the **Permissions** tab, choose **Add inline policy**, 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess",
        "lakeformation:GrantPermissions"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": ["iam:PassRole"],
      "Resource": [
        "arn:aws:iam::<account-id>: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, 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.

## 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 \(p. 106\)](#).

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` IAM user has these permissions implicitly.

- lakeformation:PutDataLakeSettings
- lakeformation:GetDataLakeSettings

### To create a data lake administrator (console)

1. If the IAM user who is to be a data lake administrator does not yet exist, use the IAM console to create it. Otherwise, view the existing IAM 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.
AWSGlueConsoleFullAccess, CloudWatchLogsReadOnlyAccess	Optional	Attach these policies if the data lake administrator will be troubleshooting workflows created from Lake Formation blueprints. These policies enable the data lake administrator to view troubleshooting information in the AWS Glue console and the Amazon CloudWatch Logs console. For information about workflows, see <a href="#">Importing Data Using Workflows</a> (p. 58).
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 <a href="#">the section called "Cross-Account Access"</a> (p. 81).
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"](#) (p. 150).

#### 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.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PassRolePermissions",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::<account-id>: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 [the section called "Cross-Account Access" \(p. 81\)](#).

A suggested name for the policy is `RAMAccess`.

```

{
  "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 the IAM Administrator user that you created in [Create an Administrator IAM User \(p. 6\)](#) or as any IAM administrative user.
6. Do one of the following:
  - If a welcome message appears, choose **Add administrators**.
  - In the navigation pane, under **Permissions**, choose **Admins and database creators**. Then under **Data lake administrators**, choose **Grant**.
7. In the **Manage data lake administrators** dialog box, for **IAM users and roles**, choose the IAM user that you created or selected in Step 1, and then choose **Save**.

## Change Data Catalog Settings

Lake Formation starts with the "Use only IAM access control" settings enabled for compatibility with existing AWS Glue Data Catalog behavior. We recommend that you disable these settings to enable fine-grained access control with Lake Formation permissions.

For more information, see [the section called "Changing the Default Security Settings for Your Data Lake" \(p. 147\)](#).

### Important

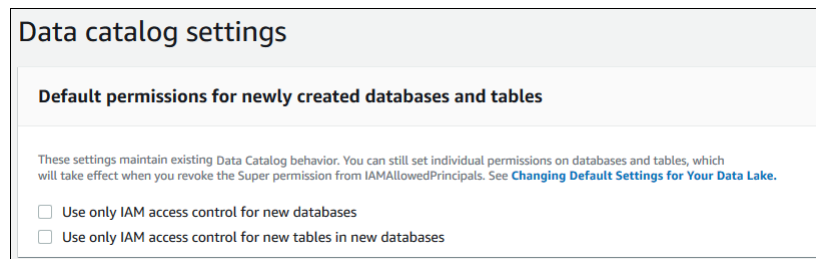
If you have existing AWS Glue Data Catalog databases and tables, do not follow the instructions in this section. Instead, follow the instructions in [Upgrading AWS Glue Data Permissions to the Lake Formation Model \(p. 156\)](#).

### 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" \(p. 133\)](#).

### 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 the IAM administrator user that you created in [Create an Administrator IAM User \(p. 6\)](#) or as an IAM user with the AdministratorAccess AWS managed policy.
2. In the navigation pane, under **Data catalog**, choose **Settings**.
3. Clear both check boxes and choose **Save**.



**Data catalog settings**

**Default permissions for newly created databases and tables**

These settings maintain existing Data Catalog behavior. You can still set individual permissions on databases and tables, which will take effect when you revoke the Super permission from IAMAllowedPrincipals. See [Changing Default Settings for Your Data Lake](#).

☐ Use only IAM access control for new databases

☐ Use only IAM access control for new tables in new databases

4. Sign out of the Lake Formation console and sign back in as the data lake administrator.
5. In the navigation pane, under **Permissions**, choose **Admins and database creators**.
6. Under **Database creators**, select the IAMAllowedPrincipals group, and choose **Revoke**.

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

7. Choose **Revoke**.

## (Optional) Allow Data Filtering on Amazon EMR Clusters

If you intend to analyze and process data in your data lake with Amazon EMR, you must opt in to allow Amazon EMR clusters to access data managed by Lake Formation. If you don't opt in, Amazon EMR clusters 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 analytics 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. 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.

By opting in to allow data filtering on the EMR cluster, you are certifying that you have properly secured the cluster.

### To opt in to allow data filtering on Amazon EMR clusters (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 [Create an Administrator IAM User \(p. 6\)](#) has this permission.
2. In the navigation pane, under **Permissions**, choose **External data filtering**.
3. On the **External data filtering** page, do the following:
  - a. Turn on **Allow Amazon EMR clusters to filter data managed by Lake Formation**.
  - b. For **AWS account IDs**, enter the account IDs of AWS accounts with Amazon EMR clusters that are to perform data filtering. Press **Enter** after each account ID.
  - c. Choose **Save**.

## (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*.

# Getting Started with AWS Lake Formation

To learn about Lake Formation, go through one of tutorials provided in this guide.

- [Tutorial: Creating a Data Lake from an AWS CloudTrail Source \(p. 13\)](#)

In this tutorial, you use your own CloudTrail logs as a data source.

- [Tutorial: Creating a Data Lake from a JDBC Source in Lake Formation \(p. 21\)](#)

In this tutorial, you use one of your JDBC-accessible data stores, such as a relational database, as a data source.

## Note

You can go through both tutorials. However, some steps, such as creating users, are duplicated, and can be skipped in the second tutorial. You can use the users that you created in the first tutorial in the second tutorial. The order in which you go through the tutorials is not important.

Before you begin, make sure that you've completed the steps in [Setting Up AWS Lake Formation \(p. 6\)](#).

The following are the general steps to create and use 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

- [Tutorial: Creating a Data Lake from an AWS CloudTrail Source \(p. 13\)](#)
- [Tutorial: Creating a Data Lake from a JDBC Source in Lake Formation \(p. 21\)](#)

## Tutorial: 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.

## Topics

- [About the Personas in This Tutorial \(p. 14\)](#)

- [AWS CloudTrail Tutorial Prerequisites \(p. 14\)](#)
- [Step 1: Create the IAM User to Be the Data Analyst \(p. 14\)](#)
- [Step 2: Add Permissions to Read AWS CloudTrail Logs to the Workflow Role \(p. 15\)](#)
- [Step 3: Create an Amazon S3 Bucket for the Data Lake \(p. 16\)](#)
- [Step 4: Register an Amazon S3 Path \(p. 16\)](#)
- [Step 5: Grant Data Location Permissions \(p. 16\)](#)
- [Step 6: Create a Database in the Data Catalog \(p. 16\)](#)
- [Step 7: Grant Data Permissions \(p. 17\)](#)
- [Step 8: Use a Blueprint to Create a Workflow \(p. 18\)](#)
- [Step 9: Run the Workflow \(p. 19\)](#)
- [Step 10: Grant SELECT on the Tables \(p. 20\)](#)
- [Step 11: Query the Data Lake Using Amazon Athena \(p. 20\)](#)

## About the Personas in This Tutorial

The following table lists the personas in this tutorial.

### Personas in This Tutorial

Persona	Description
IAM Administrator	User who can create IAM users and roles and Amazon S3 buckets. Has the AdministratorAccess AWS managed policy.
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.
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.

## AWS CloudTrail Tutorial Prerequisites

Before you begin:

- Ensure that you have completed the tasks in [Setting Up AWS Lake Formation \(p. 6\)](#).
- Know the location of your CloudTrail logs.

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

## Step 1: Create the IAM User to Be the Data Analyst

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 the IAM administrator user that you created in [Create an Administrator IAM User \(p. 6\)](#) or as an IAM 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`.

### Important

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

```
{
  "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 the IAM administrator user that you created in [Create an Administrator IAM User \(p. 6\)](#).
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 [How Do I Create an S3 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 \(p. 32\)](#).

## 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 \(p. 70\)](#).

## 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 navigation pane, under **Permissions**, choose **Data permissions**.
2. Choose **Grant**, and in the **Grant data permissions** dialog box, make these selections:
  - a. Under **Principals**, for **IAM user and roles**, choose `LakeFormationWorkflowRole`.
  - b. Under **Policy tags or catalog resources**, choose **Named data catalog resources**.
  - c. For **Databases**, choose the database that you created previously, `lakeformation_cloudtrail`.
  - d. Under **Database permissions**, select **Create table**, **Alter**, and **Drop**, and clear **Super** if it is selected.

Your **Grant data permissions** dialog box should now look like this screenshot.

**Grant data permissions**

**Principals**

☒ **IAM users and roles**  
Users or roles from this AWS account.

☐ **SAML users and groups**  
SAML users and group or QuickSight ARNs.

☐ **External accounts**  
AWS accounts or AWS organizations outside of this account.

**IAM users and roles**  
Add one or more IAM users or roles.

Choose IAM principals to add

**LakeFormationWorkflowRole** X  
Role

**Policy tags or catalog resources**

☐ **Resources matched by policy tags (recommended)**  
Manage permissions indirectly for resources or data matched by a specific set of policy tags.

☒ **Named data catalog resources**  
Manager permissions for specific databases or tables, in addition to fine-grained data access.

**Databases**  
Select one or more databases.

Choose databases

**lakeformation-cloudtrail** X

**Tables - optional**  
Select one or more tables.

Choose tables

**Database permissions**

**Database permissions**  
Choose specific access permissions to grant.

☒ Create table ☒ Alter ☒ Drop

☐ Describe

☐ Super  
This permission is the union of all the individual permissions to the left, and supersedes them.

**Grantable permissions**  
Choose the permission that may be granted to others.

☐ Create table ☐ Alter ☐ Drop

☐ Describe

☐ Super  
This permission allows the principal to grant any of the permissions to the left, and supersedes those grantable permissions.

3. Choose **Grant**.

For more information about granting Lake Formation permissions, see [Security and Access Control to Metadata and Data in Lake Formation](#) (p. 64).

## Step 8: Use a Blueprint to Create a Workflow

The 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. In the navigation pane, choose **Blueprints**, and then choose **Use blueprint**.
2. On the **Use a blueprint** page, under **Blueprint type**, choose **AWS CloudTrail**.



- Under **Import source**, choose a CloudTrail source and start date.
- Under **Import target**, specify these parameters:

<b>Target database</b>	lakeformation_cloudtrail
<b>Target storage location</b>	s3://<yourName>-datalake-cloudtrail
<b>Data format</b>	Parquet

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

<b>Workflow name</b>	lakeformationcloudtrailtest
<b>IAM role</b>	LakeFormationWorkflowRole
<b>Table prefix</b>	cloudtrailtest

**Note**  
Must be lower case.

- 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:

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

The workflow opens in the AWS Glue console.

- Ensure that the workflow is selected, and choose the **History** tab.
- 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 navigation pane, under **Permissions**, choose **Data permissions**.
2. Choose **Grant**, and in the **Grant data permissions** dialog box, make these selections:
  - a. Under **Principals**, for **IAM user and roles**, choose `datalake_user`.
  - b. Under **Policy tags or catalog resources**, choose **Named data catalog resources**.
  - c. For **Databases**, choose `lakeformation_cloudtrail`.  
  
The **Tables** list populates.
    - d. For **Tables**, choose `cloudtrailtest-cloudtrail`.
    - 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 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 pop-up menu beside the table `cloudtrailtest-cloudtrail`, choose **Preview table**.

The query runs and displays 10 rows of data.

### Note

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

# Tutorial: 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

- [About the Personas in This Tutorial \(p. 21\)](#)
- [JDBC Tutorial Prerequisites \(p. 22\)](#)
- [Step 1: Create the IAM User to Be the Data Analyst \(p. 22\)](#)
- [Step 2: Create a Connection in AWS Glue \(p. 23\)](#)
- [Step 3: Create an Amazon S3 Bucket for the Data Lake \(p. 23\)](#)
- [Step 4: Register an Amazon S3 Path \(p. 23\)](#)
- [Step 5: Grant Data Location Permissions \(p. 24\)](#)
- [Step 6: Create a Database in the Data Catalog \(p. 24\)](#)
- [Step 7: Grant Data Permissions \(p. 24\)](#)
- [Step 8: Use a Blueprint to Create a Workflow \(p. 24\)](#)
- [Step 9: Run the Workflow \(p. 25\)](#)
- [Step 10: Grant SELECT on the Tables \(p. 26\)](#)
- [Step 11: Query the Data Lake Using Amazon Athena \(p. 26\)](#)
- [Step 12: Query the Data in the Data Lake Using Amazon Redshift Spectrum \(p. 27\)](#)
- [Step 13: Grant or Revoke Lake Formation Permissions Using Amazon Redshift Spectrum \(p. 31\)](#)

## About the Personas in This Tutorial

The following table lists the personas that are used in this [AWS Lake Formation JDBC tutorial \(p. 21\)](#).

Persona	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.
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 \(p. 22\)](#).

## JDBC Tutorial Prerequisites

Before you begin the [AWS Lake Formation JDBC tutorial \(p. 21\)](#), ensure that you've done the following:

- Complete the tasks in [Setting Up AWS Lake Formation \(p. 6\)](#).
- 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 the IAM User to Be the Data Analyst" \(p. 22\)](#).

## Step 1: Create the IAM User to Be the Data Analyst

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 the IAM administrator user that you created in [Create an Administrator IAM User \(p. 6\)](#) or as an IAM 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 the IAM administrator user that you created in [Create an Administrator IAM User](#) (p. 6).
2. In the navigation pane, under **Data catalog**, choose **Connections**.
3. On the **Connections** page, choose **Add connection**.
4. On the **Set up your connection's properties** page, enter **datalake-tutorial** as the connection name, and choose **JDBC** as the connection type. Then choose **Next**.
5. Continue through the connection wizard and save the connection.

For help with creating a connection, see [Working with Connections on the AWS Glue Console](#) 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 the IAM administrator user that you created in [Create an Administrator IAM User](#) (p. 6).
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 Console 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 **Register and ingest**, 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](#) (p. 32).

## 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](#) (p. 70).

## 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 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 **Policy 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 [Security and Access Control to Metadata and Data in Lake Formation](#) (p. 64).

## 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:

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

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

<b>Workflow name</b>	lakeformationjdbctest
<b>IAM role</b>	LakeFormationWorkflowRole
<b>Table prefix</b>	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 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 **Policy 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 the IAM administrator user that you created in [Create an Administrator IAM User \(p. 6\)](#) (user name `Administrator`) or as an IAM 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": "*"
    }
  ]
}
```

- When you are finished, choose **Review** to review the policy. The policy validator reports any syntax errors.
- 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.
- In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
- For **Select type of trusted entity**, choose **AWS service**.
- Choose the Amazon Redshift service to assume this role.
- Choose the **Redshift Customizable** use case for your service. Then choose **Next: Permissions**.
- Search for the permissions policy that you created, **RedshiftLakeFormationPolicy**, and select the check box next to the policy name in the list.
- Choose **Next: Tags**.
- Choose **Next: Review**.
- For **Role name**, enter the name **RedshiftLakeFormationRole**.
- (Optional) For **Role description**, enter a description for the new role.
- Review the role, and then choose **Create role**.

### To grant **select** permissions on the table to be queried in the Lake Formation database

- Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Sign in as the data lake administrator.
- In the navigation pane, under **Permissions**, choose **Data permissions**, and then choose **Grant**.
- 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.
- Choose **Grant**.

### To set up Amazon Redshift Spectrum and run queries (new Amazon Redshift console)

#### Note

The following instructions are for the new Amazon Redshift console.

- Open the Amazon Redshift console at <https://console.aws.amazon.com/redshift>. Sign in as the user **Administrator**.
- Choose **Create cluster**.
- On the **Create cluster** page, under **DC2**, select **dc2.large**.
- Scroll down, and under **Cluster details**, enter or accept these parameters:

<b>Cluster identifier</b>	redshift-lakeformation-demo
<b>Database port</b>	5439
<b>Master user name</b>	awsuser
<b>Master user password</b>	(Choose a password)

5. Expand **Cluster permissions**, and for **Available IAM roles**, choose **RedshiftLakeFormationRole**. Then choose **Add IAM role**.
6. Choose **Create cluster**.

The **Clusters** page loads.

7. Wait until the cluster status becomes **Available**. Choose the refresh icon periodically.
8. 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`
9. Sign out of the Amazon Redshift console and sign back in as user `datalake_user`.
10. 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.

11. 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>';
```

12. 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 permissions.

13. 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.

### To set up Amazon Redshift Spectrum and run queries (original Amazon Redshift console)

#### Note

The following instructions are for the original Amazon Redshift console.

1. Open the Amazon Redshift console at <https://console.aws.amazon.com/redshift/>. Sign in as the user Administrator.
2. Choose **Quick launch cluster**.

3. On the **Quick launch** page, enter or accept these parameters:

<b>Cluster identifier</b>	redshift-lakeformation-demo
<b>Database name</b>	dev
<b>Database port</b>	5439
<b>Master user name</b>	awsuser
<b>Master user password</b>	(Choose and confirm a password)
<b>Available IAM roles</b>	RedshiftLakeFormationRole

For more information, see [Managing Clusters Using the Console](#) in the *Amazon Redshift Cluster Management Guide*.

4. Choose **Launch cluster**.
5. Wait until the cluster status becomes available.

To view cluster status, in the navigation pane, choose **Clusters** and check the **Cluster Status** column. Choose the refresh icon periodically.

6. 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`.
    - AmazonRedshiftReadOnlyAccess
    - AmazonRedshiftQueryEditor
7. Sign out of Amazon Redshift console and sign back in as user `datalake_user`.
8. Open the query editor and connect to the cluster. If prompted for credentials, 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**.

#### Note

If you are not prompted for credentials and another cluster is already selected in the query editor, choose the cluster name to open the credentials dialog box.

9. 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>';
```

10. In the **Schema** list, choose `redshift_jdbc`.

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

11. Choose the eye icon next to one of the tables.

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*.

# Adding an Amazon S3 Location to Your Data Lake

To add an Amazon Simple Storage Service (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.

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](#) (p. 69).

## Topics

- [Requirements for Roles Used to Register Locations](#) (p. 32)
- [Registering an Amazon S3 Location](#) (p. 34)
- [Registering an Encrypted Amazon S3 Location](#) (p. 35)
- [Registering an Amazon S3 Location in Another AWS Account](#) (p. 37)
- [Registering an Encrypted Amazon S3 Location Across AWS Accounts](#) (p. 39)
- [Deregistering an Amazon S3 Location](#) (p. 41)

## 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.

The simplest way to register the location is to use the Lake Formation service-linked role. This role grants the required permissions on the location. However, you might want to use a user-defined role to register the location.

### Important

If you plan to access the location using Amazon EMR, you must use a user-defined role and not the Lake Formation service-linked role to register the location. If you already registered a location with the service-linked role and now 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”](#) (p. 41).

The following are the requirements for a user-defined role:

- When creating a new role, on the IAM console, on the **Create role** page, choose **AWS service**, and then under **Choose a use case**, choose **Glue**.
- The role must have trust relationships with the following entities:
  - `glue.amazonaws.com`
  - `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.

```
{
  "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"
      ]
    }
  ]
}
```

- 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PassRolePermissions",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::<account-id>:role/<role-name>"
      ]
    }
  ]
}
```

For more information about the Lake Formation service-linked role, see [the section called “Using Service-Linked Roles” \(p. 150\)](#).

## 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.

The simplest way to register the location is to use the Lake Formation service-linked role. This role grants the required permissions on the location. You may also use a custom role to register the location, provided that it meets the requirements in [the section called “Requirements for Roles Used to Register Locations” \(p. 32\)](#). For more information about the service-linked role, see [Service-Linked Role Permissions for Lake Formation \(p. 151\)](#).

You can use the AWS Lake Formation console, Lake Formation API, or AWS Command Line Interface (AWS CLI) to register an Amazon S3 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 **Register and Ingest**, choose **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) 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.

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” \(p. 32\)](#).
6. Choose **Register location**.

### To register a location (AWS CLI)

- Enter the following CLI command. Replace `<s3-path>` with a valid Amazon S3 path.

```
aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-path> --use-  
service-linked-role
```

This command uses the service-linked role to register the location. You can use the `--role-arn` argument instead to supply your own role.



For more information, see [RegisterResource Action \(Python: register\\_resource\)](#) (p. 179).

## 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 customer master keys (CMKs) and AWS managed CMKs are supported. Client-side encryption/decryption is not supported.

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 CMK, or the CMK 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.

The simplest way to register the location is to use the Lake Formation service-linked role. This role grants the required read/write permissions on the location. You may also use a custom role to register the location, provided that it meets the requirements in [the section called "Requirements for Roles Used to Register Locations"](#) (p. 32).

### Important

If you used an AWS managed CMK (`aws/s3`) 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 CMK or an AWS managed CMK.

- [Registering a location encrypted with a customer managed CMK](#) (p. 35)
- [Registering a location encrypted with an AWS managed CMK](#) (p. 36)

### To register an Amazon S3 location encrypted with a customer managed CMK

#### Note

If the CMK 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"](#) (p. 39) 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 CMK used to encrypt the location.
2. In the navigation pane, choose **Customer managed keys**, and then choose the name of the desired CMK.
3. On the CMK 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 CMK 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 **Register and Ingest**, 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"](#) (p. 32).
9. Choose **Register location**.

For more information about the service-linked role, see [Service-Linked Role Permissions for Lake Formation](#) (p. 151).

### To register an Amazon S3 location encrypted with an AWS managed CMK

**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"](#) (p. 39) 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"](#) (p. 32).

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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "<AWS managed key ARN>"
    }
  ]
}
```

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 **Register and Ingest**, 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 cannot use the Lake Formation service-linked role to register a location in another account. You must use a custom role instead. The role must meet the requirements in [the section called "Requirements for Roles Used to Register Locations" \(p. 32\)](#). For more information about the service-linked role, see [Service-Linked Role Permissions for Lake Formation \(p. 151\)](#).

## 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" \(p. 39\)](#) 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" \(p. 32\)](#). 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.

```
{
  "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 **Register and ingest**, choose **Data lake locations**.
11. 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 customer master keys (CMKs) and AWS managed CMKs 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 CMK created in AWS KMS.
- The Amazon S3 location is not in the same AWS account as the AWS Glue Data Catalog.
- The CMK 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 CMK 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 CMK.

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 CMK. As a result, you don't have to grant permissions on the CMK 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 custom role instead. The role must meet the requirements in [the section called "Requirements for Roles Used to Register Locations" \(p. 32\)](#). For more information about the service-linked role, see [Service-Linked Role Permissions for Lake Formation \(p. 151\)](#).

### To register an encrypted Amazon S3 location across AWS accounts

1. In the same AWS account as the Data Catalog, 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" \(p. 32\)](#). Ensure that the role includes a policy that grants Amazon S3 permissions on the location.
3. If the CMK is not in the same account as the Data Catalog, add to the role an inline policy that grants the required permissions on the CMK. The following is an example policy. Replace *<cmk-region>* and *<cmk-account-id>* with the region and account number of the CMK. Replace *<key-id>* with the key ID.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:<cmk-region>:<cmk-account-id>: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 *<catalog-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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<catalog-account-id>:role/<role-name>"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::<bucket-name>"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<catalog-account-id>: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 CMK.

- a. Open the AWS KMS console at <https://console.aws.amazon.com/kms>. Then, sign in as an IAM administrator or as a user who can modify the key policy of the CMK used to encrypt the location.
- b. In the navigation pane, choose **Customer managed keys**, and then choose the name of the CMK.
- c. On the CMK 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 CMK](#) in the *AWS Key Management Service Developer Guide*.

6. Open the AWS Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Sign in to the Data Catalog AWS account as the data lake administrator.
7. In the navigation pane, under **Register and ingest**, 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 **Register and Ingest**, choose **Data lake locations**.
3. Select a location, and on the **Actions** menu, choose **Remove**.
4. When prompted for confirmation, choose **Remove**.



# Managing Data Catalog Tables and Databases

AWS 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 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 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.

## Topics

- [Creating a Database \(p. 43\)](#)
- [Creating Tables \(p. 44\)](#)
- [Searching for Tables \(p. 44\)](#)
- [Sharing Data Catalog Tables and Databases Across AWS Accounts \(p. 45\)](#)
- [Accessing and Viewing Shared Data Catalog Tables and Databases \(p. 45\)](#)
- [Creating Resource Links \(p. 49\)](#)

## 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 \(p. 70\)](#).

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 **Admins and database creators** 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 Security Settings for Your Data Lake” \(p. 147\)](#).

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 [Importing Data Using Workflows](#) (p. 58).
- Create the table manually using the Lake Formation console, AWS Glue API, or AWS Command Line Interface (AWS CLI).
- Create a resource link to a table in an external account. See [the section called "Creating Resource Links"](#) (p. 49).

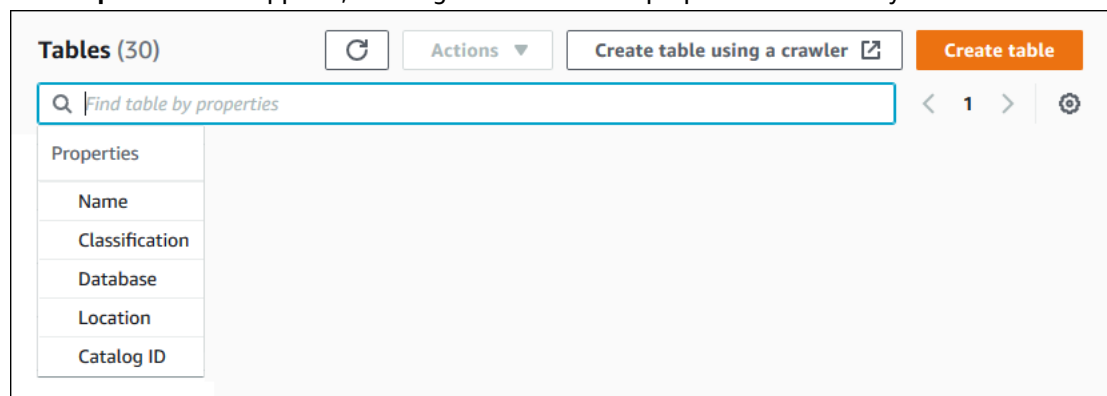
## Searching for Tables

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 at the top of the page. The field has the placeholder text *Find table by properties*.

The **Properties** menu appears, showing the various table properties to search by.

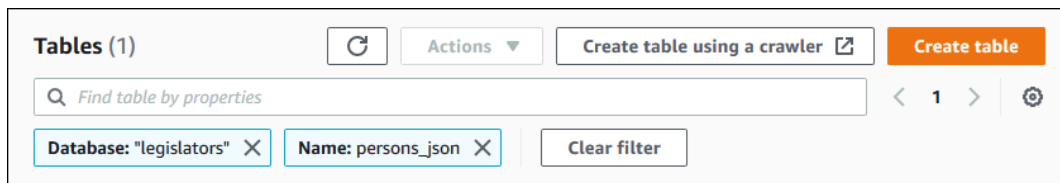


4. Do one of the following:
  - Search by containing database.
    1. Choose **Database** from the **Properties** menu, and then either choose a database from the **Databases** menu that appears or type a database name and press **Enter**.

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.



The screenshot shows the AWS Lake Formation console interface. At the top, there's a header with 'Tables (1)', a refresh button, an 'Actions' dropdown, a 'Create table using a crawler' button with an external link icon, and a 'Create table' button. Below this is a search bar with the placeholder text 'Find table by properties'. Under the search bar, there are two filter tiles: 'Database: "legislators"' and 'Name: persons\_json', each with a close button (X). To the right of these tiles is a 'Clear filter' button. On the far right, there are navigation controls showing '< 1 >' and a settings gear icon.

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.

## 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 Access: How It Works](#) (p. 81) and [Granting and Revoking Data Catalog Permissions in Lake Formation](#) (p. 112).

### See Also:

- [Accessing and Viewing Shared Data Catalog Tables and Databases](#) (p. 45)
- [Cross-Account Access Prerequisites](#) (p. 82)

## 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. The tag-based access method (TBAC) of sharing resources does not use AWS RAM. Therefore, resources that are shared across accounts by using the TBAC method are available immediately.

### See Also:

- [Sharing Data Catalog Tables and Databases Across AWS Accounts \(p. 45\)](#)
- [Cross-Account Access: How It Works \(p. 81\)](#)
- [Accessing the Underlying Data of a Shared Table \(p. 85\)](#)
- [Metadata Access Control \(p. 67\)](#) (for information about the named resource method versus the TBAC method for sharing resources.)

### Topics

- [Accepting a Resource Share Invitation from AWS RAM \(p. 46\)](#)
- [Viewing Shared Data Catalog Tables and Databases \(p. 48\)](#)

## 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" \(p. 194\)](#).

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” \(p. 194\)](#).

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.

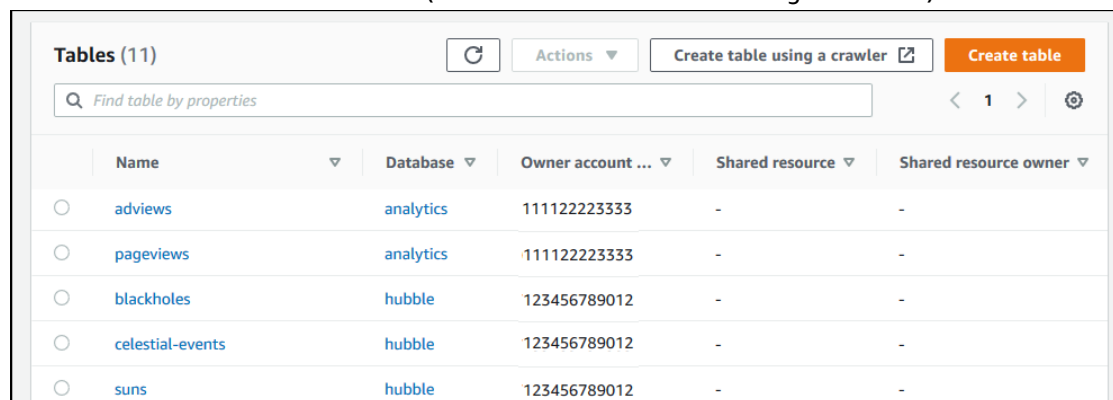
### To view shared resources using the Lake Formation console

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**.

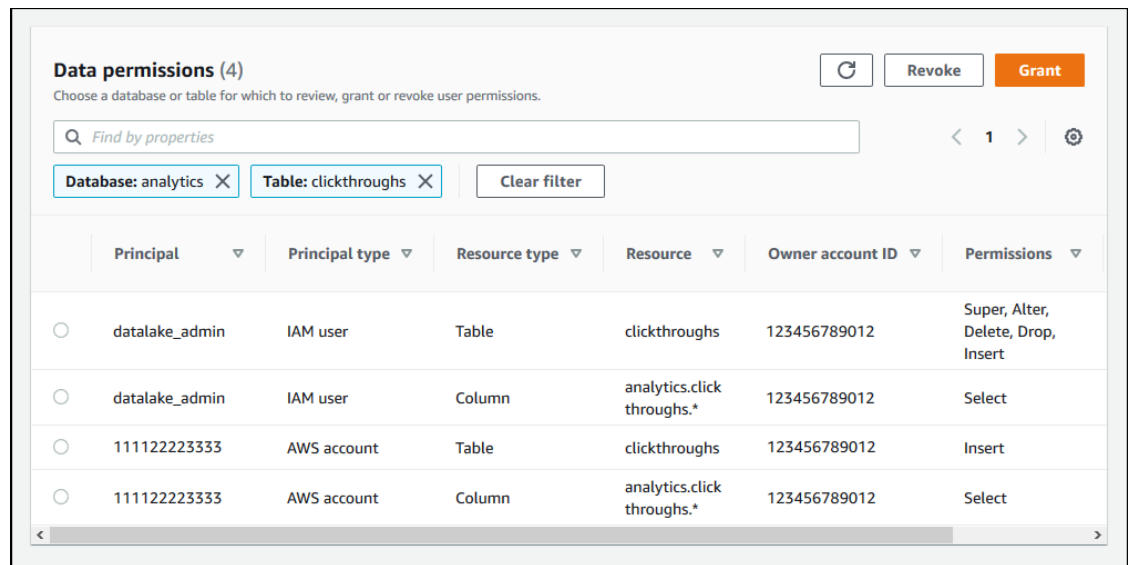
The console displays a list of databases or tables both in your account and shared with your account. For resources that are shared with your account, the console displays the owner's AWS account ID under the **Owner account ID** column (the third column in the following screenshot).



Tables (11)					
<input type="text" value="Find table by properties"/>					
	Name	Database	Owner account ...	Shared resource	Shared resource owner
<input type="radio"/>	advIEWS	analytics	111122223333	-	-
<input type="radio"/>	pageviews	analytics	111122223333	-	-
<input type="radio"/>	blackholes	hubble	123456789012	-	-
<input type="radio"/>	celestial-events	hubble	123456789012	-	-
<input type="radio"/>	suns	hubble	123456789012	-	-

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.



The screenshot shows the 'Data permissions (4)' interface in the AWS Lake Formation console. It includes a search bar with the text 'Find by properties', filter buttons for 'Database: analytics' and 'Table: clickthroughs', and a 'Clear filter' button. Below the filters is a table with columns: Principal, Principal type, Resource type, Resource, Owner account ID, and Permissions. The table lists four permissions for the 'analytics.clickthroughs.\*' resource.

	Principal	Principal type	Resource type	Resource	Owner account ID	Permissions
<input type="radio"/>	datalake_admin	IAM user	Table	clickthroughs	123456789012	Super, Alter, Delete, Drop, Insert
<input type="radio"/>	datalake_admin	IAM user	Column	analytics.clickthroughs.*	123456789012	Select
<input type="radio"/>	11112223333	AWS account	Table	clickthroughs	123456789012	Insert
<input type="radio"/>	11112223333	AWS account	Column	analytics.clickthroughs.*	123456789012	Select

### 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>.
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.

### Topics

- [How Resource Links Work in Lake Formation \(p. 49\)](#)
- [Creating a Resource Link to a Shared Data Catalog Table \(p. 51\)](#)
- [Creating a Resource Link to a Shared Data Catalog Database \(p. 53\)](#)
- [Resource Link Handling in AWS Glue APIs \(p. 54\)](#)

## 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.
- 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.

Tables (30)					
<input type="text" value="Find table by properties"/>		<a href="#">Actions</a>	<a href="#">Create table using a crawler</a>	<a href="#">Create table</a>	
Name	Database	Owner account ...	Shared resource	Shared resource owner	
<a href="#"><i>inventory-link</i></a>	retail	123456789012	<a href="#">inventory</a>	111122223333	
<a href="#"><i>incidents-link</i></a>	issues-local	123456789012	<a href="#">incidents</a>	111122223333	
<a href="#">site-logs</a>	logs	123456789012	-	-	
<a href="#">alexa-logs</a>	logs	123456789012	-	-	

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 [the section called "Lake Formation Permissions Reference" \(p. 133\)](#).

### 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 and Revoking Data Catalog Permissions in Lake Formation](#) (p. 112)

## Creating a Resource Link to a Shared Data Catalog Table

You can create a resource link to a shared table 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**, and then choose **Create table**.
3. On the **Create table** page, choose the **Resource Link** tile, and then 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**

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 [the section called "Accepting an AWS RAM Resource Share Invitation"](#) (p. 46).

### 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.

**Table details**  
Create a table in the Data Catalog.

☐ Table  
Create a table in my account

☒ Resource Link  
Create a resource link to a shared table

Resource link name  
clickthroughs-link  
Name may contain letters (A-Z), numbers (0-9), hyphens (-), or underscores (\_), and must be less than 256 characters long.

Database  
Resource link will be contained in this database.  
adtrack

Shared table  
Enter or choose a shared table.  
clickthroughs

Shared table's database  
Enter the database containing the shared table.  
analytics

Shared table owner ID  
Enter the account ID of the shared table owner.

Cancel Create

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 to principals that must be able to view the link and access the link target through the link.

### To create a resource link to a shared table (AWS CLI)

1. Enter a command similar to the following.

```
aws glue create-table --database-name myissues --  
table-input '{"Name":"mycustomers","TargetTable":  
{ "CatalogId":"111122223333","DatabaseName":"issues","Name":"customers"}}'
```

This command creates a resource link named `mycustomers` 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 to principals that must be able to view the link and access the link target through the link.

**See Also:**

- [the section called “How Resource Links Work” \(p. 49\)](#)
- [the section called “DESCRIBE” \(p. 140\)](#)

## 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 database**.
3. On the **Create database** page, choose the **Resource Link** tile, and then 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.

#### 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 [the section called “Accepting an AWS RAM Resource Share Invitation” \(p. 46\)](#).

#### 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.

**Database details**  
Create a database in the Data Catalog.

☐ Database  
Create a database in my account

☒ Resource Link  
Create a resource link to a shared database

Resource link name  
analytics-db-link  
Name may contain letters (A-Z), numbers (0-9), hyphens (-), or underscores (\_), and must be less than 256 characters long.

Shared database  
Enter or choose a shared database.  
analytics

Shared database owner ID  
Enter the account ID of the shared database owner.

Cancel Create

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 to principals that must be able to view the link and access the link target through the link.

### To create a resource link to a shared database (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 that must be able to view the link and access the link target through the link.

### See Also:

- [the section called “How Resource Links Work” \(p. 49\)](#)
- [the section called “DESCRIBE” \(p. 140\)](#)

## 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 [the section called “DESCRIBE” \(p. 140\)](#).

### 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 <code>ResourceShareType = ALL</code> 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 designated 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
<code>SearchTables</code>	Returns tables and table resource links. It doesn't follow links to return target table properties. You must specify <code>ResourceShareType = ALL</code> to see tables shared with your account.
<code>GetTableVersion</code>	Same as <code>GetTable</code> .
<code>GetTableVersions</code>	Same as <code>GetTable</code> .
<code>DeleteTableVersion</code>	Same as <code>DeleteTable</code> .
<code>BatchDeleteTableVersion</code>	Same as <code>DeleteTable</code> .

### Partition API Operations

API Operation	Resource Link Handling
<code>CreatePartition</code>	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.
<code>BatchCreatePartition</code>	Same as <code>CreatePartition</code> .
<code>UpdatePartition</code>	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.
<code>DeletePartition</code>	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 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.
<code>BatchDeletePartition</code>	Same as <code>DeletePartition</code> .
<code>GetPartition</code>	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.
<code>GetPartitions</code>	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.
<code>BatchGetPartition</code>	Same as <code>GetPartition</code> .

### 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:

- [the section called “How Resource Links Work” \(p. 49\)](#)

# 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 \(p. 58\)](#)
- [Creating a workflow \(p. 59\)](#)
- [Running a workflow \(p. 61\)](#)

## 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.

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...
<ul style="list-style-type: none"><li>• Schema evolution is flexible. (Columns are re-named, previous columns are deleted, and new columns are added in their place.)</li></ul>	<ul style="list-style-type: none"><li>• Schema evolution is incremental. (There is only successive addition of columns.)</li></ul>



Use database snapshot when...	Use incremental database when...
<ul style="list-style-type: none"> <li>Complete consistency is needed between the source and the destination.</li> </ul>	<ul style="list-style-type: none"> <li>Only new rows are added; previous rows are not updated.</li> </ul>

## 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 [Create an IAM Role for Workflows](#) (p. 7) and the section called “Granting Lake Formation Permissions” (p. 104).

### 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](#) (p. 193).
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 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`.

**Incremental data**  
Enter tables in the data source to import along with bookmark columns to determine previously imported data.

Table name	Bookmark keys	Bookmark order	Partitioning scheme - optional	
<input type="text" value="Enter a table name"/>	<input type="text" value="Enter a bookmark key"/> <small>Comma-delimited list of bookmark columns.</small>	<input type="text" value="Choose a sort. ▼"/>	<input type="text" value="Type partitioning"/>	<input type="button" value="Remove"/>
<input type="button" value="Add"/>				

For more information, see [Tracking Processed Data Using Job Bookmarks](#) in the *AWS Glue Developer Guide*.

- 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.

- Choose an import frequency.

You can specify a `cron` expression with the **Custom** option.

- Under **Import options**:
  - Enter a workflow name.
  - For role, choose the role `LakeFormationWorkflowRole`, which you created in [Create an IAM Role for Workflows \(p. 7\)](#).
  - Optionally specify a table prefix. The prefix is prepended to the names of Data Catalog tables that the workflow creates.
- 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:**

- [the section called “Blueprints and Workflows” \(p. 58\)](#)

## 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 \(p. 193\)](#).
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:**

- [the section called “Blueprints and Workflows” \(p. 58\)](#)

# 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](#) (p. 62)
- [Infrastructure Security in Lake Formation](#) (p. 63)
- [Security and Access Control to Metadata and Data in Lake Formation](#) (p. 64)
- [Security Event Logging in AWS Lake Formation](#) (p. 150)
- [Using Service-Linked Roles for Lake Formation](#) (p. 150)

## Data Protection in Lake Formation

The AWS [shared responsibility model](#) applies to data protection in AWS Lake Formation. 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. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with 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 recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- 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 personal data that is stored in Amazon S3.

- If you require FIPS 140-2 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 \(FIPS\) 140-2](#).

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Lake Formation or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Lake Formation or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't 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](#) (p. 35).

## Infrastructure Security in 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.

# Security and Access Control to Metadata and Data in Lake Formation

AWS Lake Formation provides a permissions model that is based on a simple grant/revoke mechanism. Lake Formation permissions combine with AWS Identity and Access Management (IAM) permissions to control access to data stored in data lakes and to the metadata that describes that data.

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).
- 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 metadata is organized as 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 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 an IAM user or role, an Amazon QuickSight 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 [the section called “AWS Service Integrations with Lake Formation” \(p. 1\)](#).

## Topics

- [Lake Formation Access Control Overview \(p. 64\)](#)
- [Managing Policy Tags for Metadata Access Control \(p. 92\)](#)
- [Granting Lake Formation Permissions \(p. 104\)](#)
- [Viewing Database and Table Permissions in Lake Formation \(p. 144\)](#)
- [AWS Managed Policies for Lake Formation \(p. 147\)](#)
- [Changing the Default Security Settings for Your Data Lake \(p. 147\)](#)
- [Permissions Example Scenario \(p. 149\)](#)

## Lake Formation Access Control Overview

Access control in AWS Lake Formation is divided into the following two areas:

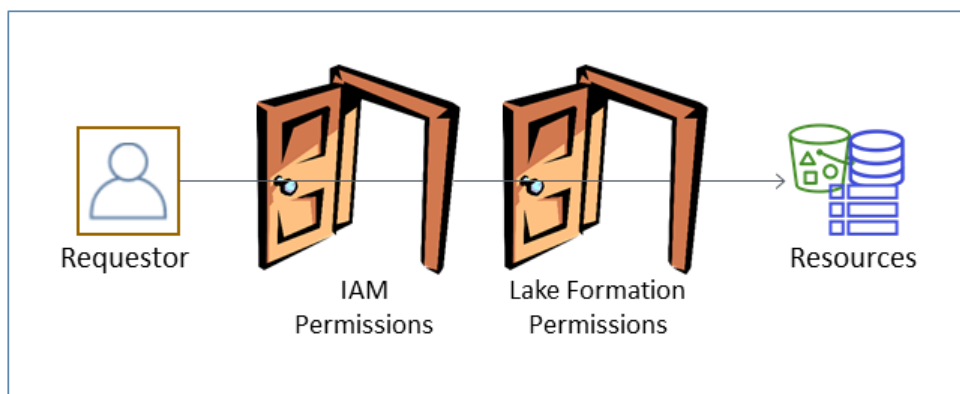
- **Metadata access control** – Permissions on Data Catalog resources (*Data Catalog permissions*).

These permissions enable principals to create, read, update, and delete metadata databases and tables in the Data Catalog.

- **Underlying data access control** – Permissions on locations in Amazon Simple Storage Service (Amazon S3) (*data access permissions* and *data location permissions*).
  - Data access 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.

**Topics**

- [Methods for Fine-Grained Access Control \(p. 66\)](#)
- [Metadata Access Control \(p. 67\)](#)
- [Underlying Data Access Control \(p. 69\)](#)
- [Tag-Based Access Control in Lake Formation \(p. 73\)](#)
- [Cross-Account Access in Lake Formation \(p. 81\)](#)

## 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	<p><b>This is the default method</b> for backward compatibility with AWS Glue.</p> <ul style="list-style-type: none"> <li><i>Open</i> means that the special permission <code>Super</code> is granted to the group <code>IAMAllowedPrincipals</code>, where <code>IAMAllowedPrincipals</code> 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 <code>Super</code> 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 Catalog resources and Amazon S3 locations to be controlled solely by IAM policies. For more information, see <a href="#">the section called "Changing the Default Security Settings for Your Data Lake"</a> (p. 147) and <a href="#">Upgrading AWS Glue Data Permissions to the Lake Formation Model</a> (p. 156).</li> <li><i>Fine-grained</i> means that IAM policies control all access to Data Catalog resources and to individual Amazon S3 buckets.</li> </ul> <p>On the Lake Formation console, this method appears as <b>Use only IAM access control</b>.</p>
Method 2	Fine-grained	Coarse-grained	<p><b>This is the recommended method.</b></p> <ul style="list-style-type: none"> <li><i>Fine-grained</i> access means granting limited Lake Formation permissions to individual principals on Data Catalog resources, Amazon S3 locations, and the underlying data in those locations.</li> <li><i>Coarse-grained</i> means broader permissions on individual operations and on access to Amazon S3 locations. For example, a coarse-grained IAM policy might include <code>"glue:*"</code> or <code>"glue:Create*"</code> rather than <code>"glue:CreateTables"</code>, 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</li> </ul>



Method	Lake Formation Permissions	IAM Permissions	Comments
			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 [the section called “Changing the Default Security Settings for Your Data Lake” \(p. 147\)](#).
- Data lake administrators and database creators have implicit Lake Formation permissions that you must understand. For more information, see [Implicit Lake Formation Permissions \(p. 106\)](#).

## 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 *policy tags* to Data Catalog databases, tables, and columns, and grant permissions on one or more tags to principals. Each policy tag is a key-value pair, such as `department=sales`. A principal that has tags that match the 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 [Overview of Tag-Based Access Control \(p. 73\)](#).

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*

Catalog	Database	Table
		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 [the section called "Lake Formation Permissions Reference" \(p. 133\)](#).

### 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.

```
{
  "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.

```
{
  "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.

```
{
  "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](#) (p. 193).

## 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 or alter 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 *Amazon S3 location* [with grant option]

With the grant option, you can allow the grantee to 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`.

```
{
  "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"
      ]
    }
  ]
}
```

Lake Formation provides a service-linked role that you can use during registration to automatically create policies like this. For more information, see [the section called “Using Service-Linked Roles” \(p. 150\)](#).

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 following IAM permission:

`lakeformation:GetDataAccess`

With this permission, Lake Formation grants the request for temporary credentials to access the data.

### Note

Only Amazon Athena requires the user to have the `lakeformation:GetDataAccess` permission. For other integrated services, the assumed role must have the permission.

This permission is included in the suggested policies in the [Lake Formation Personas and IAM Permissions Reference \(p. 193\)](#).

To summarize, to enable Lake Formation principals to read and write underlying data with access controlled by Lake Formation permissions:

- The Amazon S3 locations that contain the data must be registered 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.

### Note

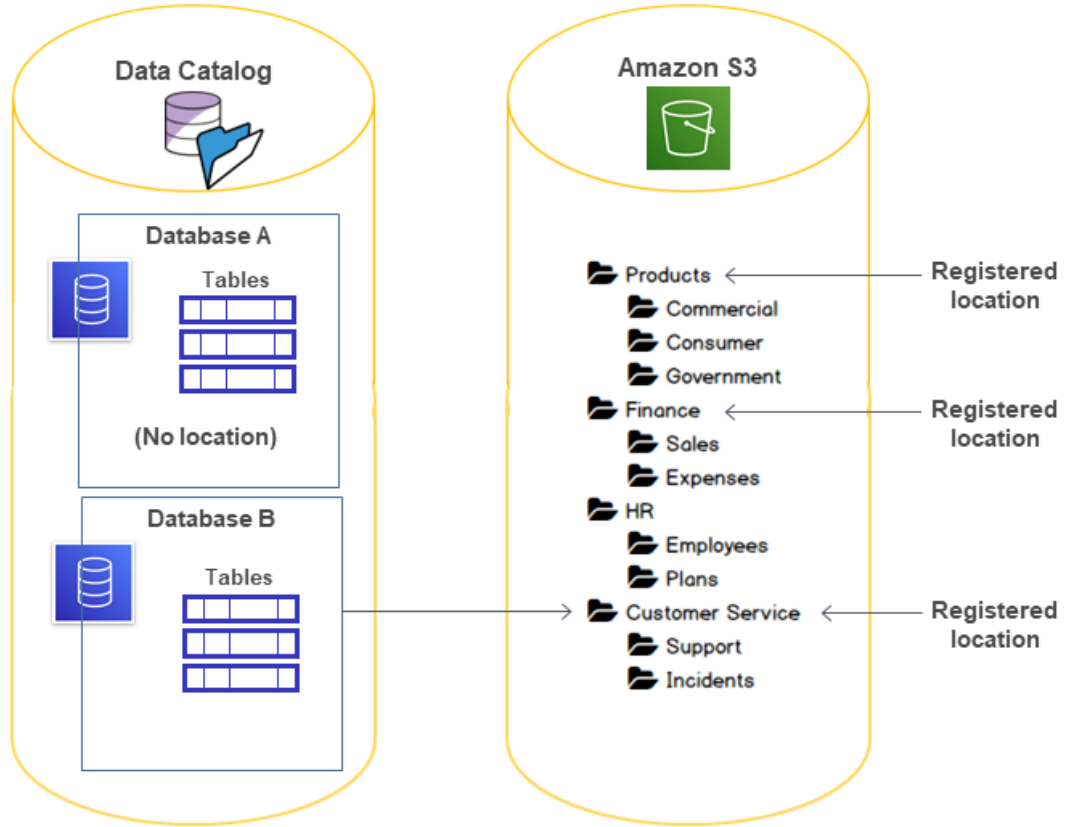
The Lake Formation permissions model doesn't prevent access to Amazon S3 locations through the Amazon S3 API or console if you have access to them through IAM or Amazon S3 policies. You can attach IAM policies to principals to block this access.

### 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.

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 <code>Finance/Sales</code>	Fails	No data location permission
Database A at <code>Products</code>	Succeeds	Has data location permission
Database A at <code>HR/Plans</code>	Succeeds	Location is not registered

Database and Location	Succeeds or Fails	Reason
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 \(p. 32\)](#)
- [the section called “Lake Formation Permissions Reference” \(p. 133\)](#)
- [Lake Formation Personas and IAM Permissions Reference \(p. 193\)](#)

## Tag-Based Access Control in Lake Formation

Tag-based access control (TBAC) is the recommended method to use to grant Lake Formation permissions when there is a large number of Data Catalog resources. TBAC is more scalable than the named resource method and requires less permission management overhead.

### Topics

- [Overview of Tag-Based Access Control \(p. 73\)](#)
- [Tag-Based Access Control Permissions Model \(p. 78\)](#)
- [Tag-Based Access Control Notes and Restrictions \(p. 80\)](#)

### See Also

- [Managing Policy Tags for Metadata Access Control \(p. 92\)](#)
- [Granting, Revoking, and Listing Policy Tag Permissions \(p. 107\)](#)
- [Granting Data Catalog Permissions Using the TBAC Method \(p. 122\)](#)

## Overview of Tag-Based Access Control

With tag-based access control (TBAC), *policy tags* are assigned to Data Catalog resources (databases, tables, and columns) and are granted to principals. A principal with tags that match the tags of a resource can access that resource.

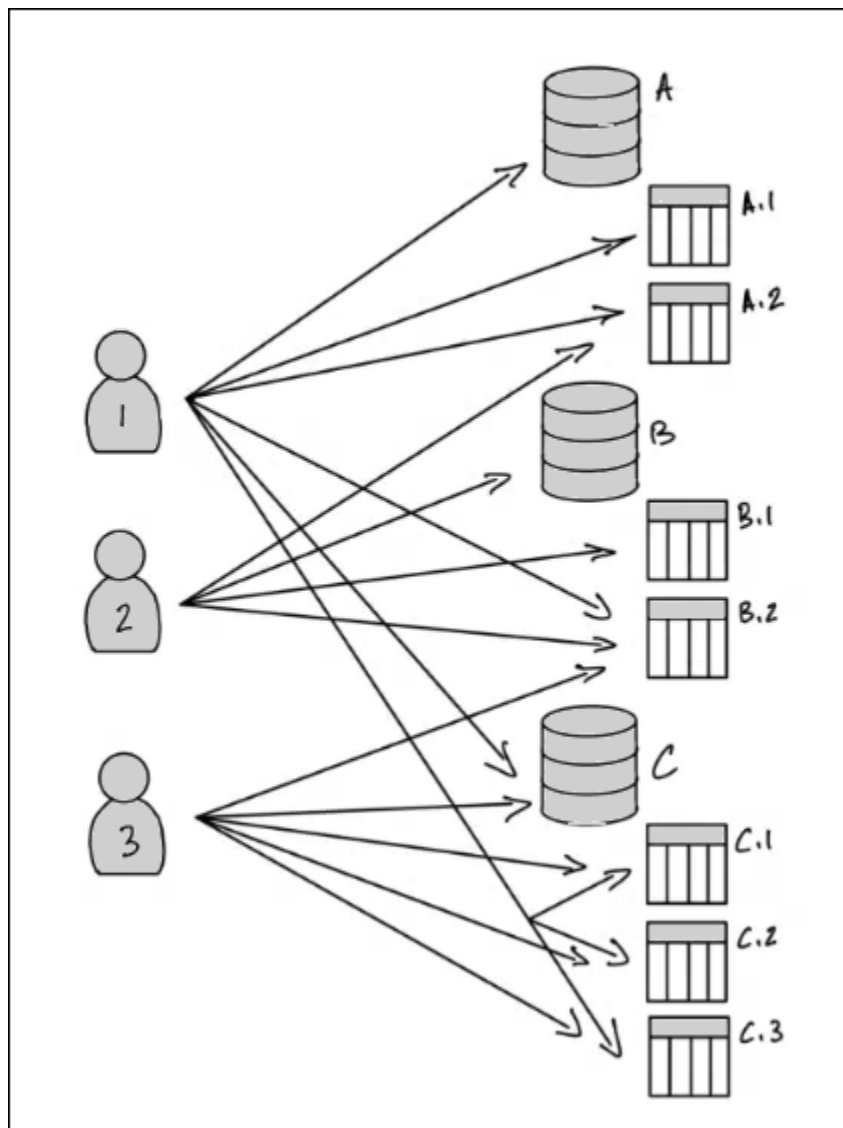
Each policy tag is a key-value pair, such as `department=sales` or `classification=restricted`. A key can have multiple defined values, such as `department=sales,marketing,engineering,finance`.

To use the TBAC method, data lake administrators and data engineers perform the following tasks.

Task	Task Details
1. Define an ontology of policy tags.	-
2. Create the policy tags in Lake Formation.	<a href="#">Creating Policy Tags (p. 93)</a>
3. Assign policy tags to Data Catalog resources.	<a href="#">Assigning Policy Tags to Data Catalog Resources (p. 97)</a>
4. Grant permissions to other principals to assign policy tags to resources, optionally with the grant option.	<a href="#">Granting, Revoking, and Listing Policy Tag Permissions (p. 107)</a>

Task	Task Details
5. Grant policy tag expressions to principals, optionally with the grant option.	<a href="#">Granting Data Catalog Permissions Using the TBAC Method (p. 122)</a>
6. (Recommended) After verifying that principals have access to the correct resources through the TBAC method, revoke permissions that were granted by using the named resource method.	-

Consider the case where a data lake administrator 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, the data lake administrator would have to make 17 grants, as follows (in pseudo-code).

```
GRANT CREATE_TABLE ON Database A TO PRINCIPAL 1
```



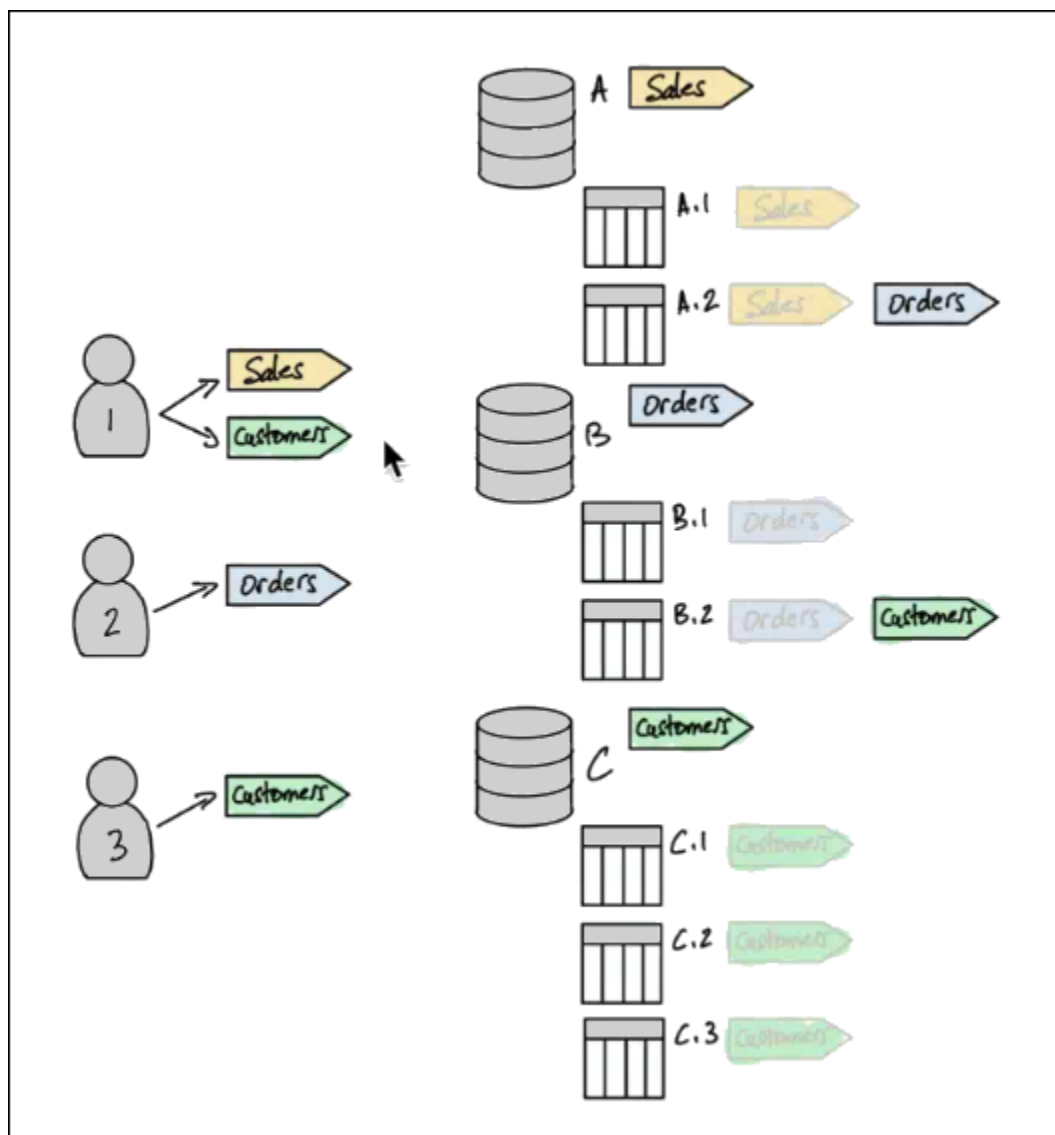
```
GRANT SELECT, INSERT ON Table A.1 TO PRINCIPAL 1
GRANT SELECT, INSERT ON Table A.2 TO PRINCIPAL 1
GRANT SELECT, INSERT ON Table B.2 TO PRINCIPAL 1
...
GRANT SELECT, INSERT ON Table A.2 TO PRINCIPAL 2
GRANT CREATE_TABLE ON Database B TO PRINCIPAL 2
...
GRANT SELECT, INSERT ON Table C.3 TO PRINCIPAL 3
```

Now consider how the data lake administrator would grant permissions by using TBAC. The following diagram indicates that the data lake administrator has assigned policy tags to databases and tables, and has granted permissions on policy tags to principals.

In this example, the tags represent areas of the data lake that contain analytics for different modules of an enterprise resource planning (ERP) application suite. The data lake administrator wants to control access to the analytics data for the various modules. All tags have the key `module` and possible values `Sales`, `Orders`, and `Customers`. An example tag looks like this:

```
module=Sales
```

The diagram shows only the tag values.



### Tag Assignments to Data Catalog Resources and Inheritance

Tables inherit tags from databases and columns inherit tags from tables. Inherited values can be overridden. In the preceding diagram, dimmed tags are inherited.

Because of inheritance, the data lake administrator needs to make only the five following 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 policy tags to the databases and tables, the data lake administrator must make only four grants of 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` tag can access Data Catalog resources with the `module=Sales` tag (for example, database A), a principal with the `module=Customers` tag can access resources with the `module=Customers` tag, and so on.

The preceding grant commands are incomplete. This is because although they indicate through 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 policy 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 Principal Permissions on Resources

Given the policy tags assigned to the databases and tables in the preceding diagram, and the policy 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 Policy Tags
Principal 1	<ul style="list-style-type: none"><li>• <code>CREATE_TABLE</code> on database A</li><li>• <code>SELECT, INSERT</code> on table A.1</li><li>• <code>SELECT, INSERT</code> on table A.2</li><li>• <code>SELECT, INSERT</code> on table B.2</li><li>• <code>CREATE_TABLE</code> on database C</li><li>• <code>SELECT, INSERT</code> on table C.1</li><li>• <code>SELECT, INSERT</code> on table C.2</li><li>• <code>SELECT, INSERT</code> on table C.3</li></ul>
Principal 2	<ul style="list-style-type: none"><li>• <code>SELECT, INSERT</code> on table A.2</li><li>• <code>CREATE_TABLE</code> on database B</li><li>• <code>SELECT, INSERT</code> on table B.1</li><li>• <code>SELECT, INSERT</code> on table B.2</li></ul>
Principal 3	<ul style="list-style-type: none"><li>• <code>SELECT, INSERT</code> on table B.2</li><li>• <code>CREATE_TABLE</code> on database C</li><li>• <code>SELECT, INSERT</code> on table C.1</li><li>• <code>SELECT, INSERT</code> on table C.2</li><li>• <code>SELECT, INSERT</code> on table C.3</li></ul>

### 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 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 TBAC method, using a single tag, the total of the number of grants to principals and assignments to resources is  $n(P) + n(R)$ .

### See Also

- [Managing Policy Tags for Metadata Access Control \(p. 92\)](#)
- [Granting Data Catalog Permissions Using the TBAC Method \(p. 122\)](#)

## Tag-Based Access Control Permissions Model

The following are the rules and permissions that you must understand to effectively use the tag-based access control (TBAC) method for securing your data lake.

- All policy tags must be predefined before they can be assigned to Data Catalog resources or granted to principals.

Data engineers and analysts decide on an ontology for tags. The data lake administrator then creates and maintains the tags in Lake Formation. Only the data lake administrator can perform create, update, and delete operations on tags.

- You can assign multiple 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 tags to resources when you create the resource. You can only add tags to existing resources.
- You can grant tag expressions, not just single tags, to a principal.

A tag expression looks something like the following (in pseudo-code).

```
module=sales AND division=(consumer OR commercial)
```

A principal that is granted this 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 tag expression consists of just one tag, such as `module=sales`.

- A principal that is granted permissions on a tag with multiple values can access Data Catalog resources with either of those values. For example, if a user is granted a tag with `key=module` and `values=orders, customers`, the user has access to resources that are assigned either `module=orders` or `module=customers`.
- At first, only the data lake administrator can assign tags to Data Catalog resources. The data lake administrator can grant the `DESCRIBE` and `ASSOCIATE` permissions on tags to principals so that those principals can view and assign tags. The following table describes these permissions.

Permission	Description
DESCRIBE	A principal with this permission on a tag can view the tag and its values when they assign tags to resources or grant permissions on tags. You can grant DESCRIBE on all key values or on specific values.
ASSOCIATE	A principal with this permission on a tag can assign the tag to a Data Catalog resource. Granting ASSOCIATE 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.

- At first, the data lake administrator is the only principal who can grant permissions on Data Catalog resources (data permissions) by using the TBAC method. If the data lake administrator grants data permissions with TBAC to a principal in their account with the grant option, the grant recipient can then grant data permissions on the resources in one of two ways:
  - Using the named resource method.
  - Using the TBAC method, but only using the same 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 TBAC method, but only with the complete tag expression `module=customers, region=west,south`.

- Although data lake administrators have implicit Lake Formation permissions to create, update, and delete tags, to assign tags to resources, and to grant tags to principals, data lake administrators also need the following TBAC-related AWS Identity and Access Management (IAM) permissions.

```
"lakeformation:AddLFTagsToResource",  
"lakeformation:RemoveLFTagsFromResource",  
"lakeformation:GetResourceLFTags",  
"lakeformation:ListLFTags",  
"lakeformation:CreateLFTag",  
"lakeformation:GetLFTag",  
"lakeformation:UpdateLFTag",  
"lakeformation>DeleteLFTag",  
"lakeformation:SearchTablesByLFTags",  
"lakeformation:SearchDatabasesByLFTags"
```

Principals who assign tags to resources and grant tags to principals must have the same permissions, except for the `CreateLFTag`, `UpdateLFTag`, and `DeleteLFTag` permissions.

For more information, see [Lake Formation Personas and IAM Permissions Reference \(p. 193\)](#).

- If a principal is granted permissions on a resource with both the 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 tags across accounts, and granting permissions on Data Catalog resources across accounts using the TBAC method. In both cases, the principal is an AWS account ID.

**Note**

Currently, TBAC cross-account grants to organizations and organizational units are not supported.

For more information, see [Cross-Account Access in Lake Formation \(p. 81\)](#).

### Example – Life Cycle of a Policy Tag

1. The data lake administrator Michael creates a tag `module=Customers`.
2. Michael grants `ASSOCIATE` on the 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 tags to the table. For more information, see [Assigning Policy Tags to Data Catalog Resources \(p. 97\)](#).
4. Eduardo assigns the 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 \(p. 67\)](#)

## Tag-Based Access Control Notes and Restrictions

The following are notes and restrictions for tag-based access control:

- Using tag-based access control (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 [Tag-Based Access Control Cross-Account Prerequisites \(p. 83\)](#).
- Tag keys can't exceed 128 characters in length. Each tag value can be up to 256 characters in length.
- The maximum number of tags that can be assigned to a Data Catalog resource is 50.
- The following limits are soft limits:
  - The maximum number of tags that can be created is 1000.
  - The maximum number of values that can be defined for a tag is 15.
- Tags keys and values are converted to all lower case when they are stored.
- Only one value for a tag can be assigned to a particular resource.
- If multiple tags are granted to a principal with a single grant, the principal can access only Data Catalog resources that have all of the tags.
- AWS Glue ETL jobs require full table access, so only tags on databases and tables are applicable. If any tables processed by the job have policy tags on columns, the job fails.
- If a 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.
- Granting cross-account permissions on tags is supported, but only for granting to external AWS accounts, not to organizations or organizational units.

## Cross-Account Access in Lake Formation

AWS Lake Formation allows cross-account access to 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.

### Topics

- [Cross-Account Access: How It Works \(p. 81\)](#)
- [Cross-Account Access Prerequisites \(p. 82\)](#)
- [Tag-Based Access Control Cross-Account Prerequisites \(p. 83\)](#)
- [Cross-Account Best Practices and Limitations \(p. 84\)](#)
- [Accessing the Underlying Data of a Shared Table \(p. 85\)](#)
- [Cross-Account CloudTrail Logging \(p. 86\)](#)
- [Managing Cross-Account Permissions Using Both AWS Glue and Lake Formation \(p. 89\)](#)
- [Viewing All Cross-Account Grants Using the GetResourcePolicies API Operation \(p. 91\)](#)

### Related Topics

- [Sharing Data Catalog Tables and Databases Across AWS Accounts \(p. 45\)](#)
- [Granting Lake Formation Permissions \(p. 104\)](#)
- [Accessing and Viewing Shared Data Catalog Tables and Databases \(p. 45\)](#)
- [Creating Resource Links \(p. 49\)](#)
- [Troubleshooting Cross-Account Access \(p. 199\)](#)

## Cross-Account Access: How It Works

To enable cross-account access, you grant Lake Formation permissions with the grant option on Data Catalog tables and databases (Data Catalog resources) to an external AWS account, organization, or organizational unit. The grant operation automatically shares those resources.

You don't share resources with specific principals in external AWS accounts—you share the resources only with the accounts. Granting Lake Formation permissions to an organization or organizational unit is equivalent to granting the permission to every AWS account in that organization or organizational unit.

### Note

Granting cross-account permissions to organizations and organizational units is not supported for tag-based access control (TBAC).

When you use the named resource method to grant Lake Formation permissions on a Data Catalog resource to an external 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.

### Note

The tag-based access control (TBAC) method of granting Data Catalog permissions does not use AWS RAM for cross-account grants. Therefore, cross-account grants are available immediately. For more information, see [Tag-Based Access Control in Lake Formation \(p. 73\)](#).

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)

In each account that accesses a shared resource:

- At least one user must be a data lake administrator. For information on how to create a data lake administrator, see [Create a Data Lake Administrator](#) (p. 8).
- The data lake administrator can view shared resources and 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. Because the data lake administrator must grant permissions on shared resources to the principals in the grantee account, cross-account permissions must always be granted with the grant option.
- 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 were 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.
- Principals can create a *resource link* in their Data Catalog to a shared resource from another AWS account. Integrated services such as Amazon Athena and Amazon Redshift Spectrum require resource links to be able to include shared resources in queries. For more information about resource links, see [the section called "How Resource Links Work"](#) (p. 49).
- When the underlying data of a shared resource is accessed, AWS 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 [the section called "Cross-Account CloudTrail Logging"](#) (p. 86).

#### See also:

- [Sharing Data Catalog Tables and Databases Across AWS Accounts](#) (p. 45)
- [Accessing and Viewing Shared Data Catalog Tables and Databases](#) (p. 45)
- [Granting and Revoking Data Catalog Permissions in Lake Formation](#) (p. 112)
- [Creating Resource Links](#) (p. 49)
- [What is AWS Organizations](#) in the *AWS Organizations User Guide*

## Cross-Account Access Prerequisites

Before your AWS account can share Data Catalog databases and tables (Data Catalog resources), and before you can access resources shared with your account, the following prerequisites must be met:

- If you're currently using an AWS Glue Data Catalog resource policy and you want to grant cross-account permissions using the named resource method, you must either remove the policy or add new permissions to it that are required for cross-account grants. If you intend to use the tag-based access control (TBAC) method, you must have a Data Catalog resource policy that enables TBAC. For more information, see [Managing Cross-Account Permissions Using Both AWS Glue and Lake Formation](#) (p. 89) and [Tag-Based Access Control Cross-Account Prerequisites](#) (p. 83).
- Before granting cross-account permissions on a Data Catalog resource, you must revoke all Lake Formation permissions from the `IAMAllowedPrincipals` group for the resource.
- For Data Catalog 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.

```
aws glue update-database --name <database> --database-input
'{"Name": "<database>", "CreateTableDefaultPermissions": []}'
```

Also, for databases where you want external accounts to create tables, ensure that this setting is off.

- If you want to use the named resources method to share Data Catalog resources with your organization 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.

- You can't grant cross-account access to a database or table that is encrypted—that was created in an encrypted Data Catalog—if you don't have permissions on the Data Catalog encryption key.
- Users who want to use the named resources method to grant cross-account permissions must have the required AWS Identity and Access Management (IAM) permissions on AWS Glue and the AWS Resource Access Manager (AWS RAM) service. The AWS managed policy `AWSLakeFormationCrossAccountManager` grants the required permissions.

Data lake administrators in accounts that receive resources that were shared with the named resource method 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.

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

- There is an additional requirement when you use an AWS Glue crawler to crawl an Amazon S3 location in another account and save the resulting tables in a database in the other account. The S3 bucket must have a bucket policy that grants permissions on the bucket to the crawler role.

### See Also

- [Tag-Based Access Control Cross-Account Prerequisites](#) (p. 83)

## Tag-Based Access Control Cross-Account Prerequisites

Before you can use the tag-based access control (TBAC) method to grant cross-account access to Data Catalog resources, you must add the following JSON permissions object to your AWS Glue Data Catalog resource policy. You must add this code for each AWS account that you are granting permissions to.

To add this code, you can use the **Settings** page on the AWS Glue console, or the `glue:PutResourcePolicy` API operation.

Replace `<recipient-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 your AWS account ID.

```
{
  "Effect": "Allow",
  "Action": [
    "glue:*"
  ],
  "Principal": {
    "AWS": [
      "<recipient-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
    }
  }
}
```

#### Note

All code in the resource policy must be within a `Statement`.

```
{
  "Version": "2012-10-17",
  "Statement": [ ]
}
```

#### Important

If you are currently also granting cross-account permissions by using the named resource method, 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 \(p. 89\)](#).

#### See Also

- [Managing Cross-Account Permissions Using Both AWS Glue and Lake Formation \(p. 89\)](#)
- [Metadata Access Control \(p. 67\)](#)

## Cross-Account Best Practices and Limitations

The following are best practices and limitations of cross-account access:

- There is no limit to the number of Lake Formation permission grants that you can make to principals in your own AWS account. However, the maximum number of cross-account grants that your account can make with the named resource method is 1,600. To avoid reaching this limit, follow these best practices for the named resource method:
- 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](#) (p. 45).

- 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 and Revoking Data Catalog Permissions in Lake Formation](#) (p. 112).

**Note**

The limit of 1,600 grants is soft limit. To exceed this limit, request a higher limit for the number of resource shares in AWS Resource Access Manager (AWS RAM). For more information, 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 [the section called "Creating Resource Links"](#) (p. 49).

- 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**

- [Tag-Based Access Control Notes and Restrictions](#) (p. 80)
- [CREATE\\_TABLE](#) (p. 138) in the [the section called "Lake Formation Permissions Reference"](#) (p. 133) for more cross-account access rules and limitations.

## 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 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.

```
{
  "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:<region>:<account-id-A>:table/<database>/<table>",
        "arn:aws:glue:<region>:<account-id-A>:database/<database>",
        "arn:aws:glue:<region>:<account-id-A>:catalog"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "arn:aws:lakeformation:<region>:<account-id-A>:catalog:<account-id-A>"
      ],
      "Condition": {
        "StringEquals": {
          "lakeformation:GlueARN": "arn:aws:glue:<region>:<account-id-A>:table/<database>/<table>"
        }
      }
    }
  ]
}
```

#### See Also:

- [the section called “Accepting an AWS RAM Resource Share Invitation” \(p. 46\)](#)

## 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 \(p. 87\)](#)
- [Querying CloudTrail Logs for Amazon S3 Cross-Account Access \(p. 88\)](#)

## 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 AdministratorAccess 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="GetObject"` or `eventName="PutObject"`.

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": "GetObject"
```

```
.....  
.....  
"principalId": "AROQSOX5XXUR7D6RMYLR:AWSLF-00-GL-111122223333-B8JSAjo5QA",  
"arn": "arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForLakeFormationDataAccess/  
AWSLF-00-GL-111122223333-B8JSAjo5QA",  
"sessionContext": {  
  "sessionIssuer": {  
    "type": "Role",  
    "principalId": "AROQSOX5XXUR7D6RMYLR",  
    "arn": "arn:aws:iam::111122223333:role/aws-service-role/lakeformation.amazonaws.com/  
AWSServiceRoleForLakeFormationDataAccess",  
    "accountId": "111122223333",  
    "userName": "AWSServiceRoleForLakeFormationDataAccess"  
  },  
  .....  
  .....  
}
```

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 `GrantPermissions` 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, for grants made by using the named resource method, 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 has made no cross-account grants using the named resources method, which 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` member of the `PutResourcePolicyRequest` structure to `'TRUE'` (type = string). The following code example shows how to do this in Python.

```
import boto3
import json

REGION = 'us-east-2'
PRODUCER_ACCOUNT_ID = '123456789012'
CONSUMER_ACCOUNT_IDS = ['111122223333']

glue = glue_client = boto3.client('glue')

policy = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowConsumerFullCatalogAccess",
            "Effect": "Allow",
            "Action": [
                "glue:*"
            ],
            "Principal": {
                "AWS": CONSUMER_ACCOUNT_IDS
            },
            "Resource": [
                f"arn:aws:glue:{REGION}:{PRODUCER_ACCOUNT_ID}:catalog",
                f"arn:aws:glue:{REGION}:{PRODUCER_ACCOUNT_ID}:database/*",
                f"arn:aws:glue:{REGION}:{PRODUCER_ACCOUNT_ID}:table/*/*"
            ]
        }
    ]
}

policy_str = json.dumps(policy)
glue.put_resource_policy(PolicyInJson=policy_str, EnableHybrid='TRUE')
```

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, 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.

```
{
  "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

If you are also using the tag-based access control (TBAC) method to make cross-account grants, you must have a Data Catalog resource policy with at least the permissions specified in [Tag-Based Access Control Cross-Account Prerequisites](#) (p. 83).

### See Also:

- [Metadata Access Control](#) (p. 67) (for a discussion of the named resource method versus the tag-based access control (TBAC) method).
- [Viewing Shared Data Catalog Tables and Databases](#) (p. 48)
- [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 GetResourcePolicies 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:GetResourcePolicies` 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:GetResourcePolicies` API operation.

### Note

This operation also returns the Data Catalog resource policy. However, if you enabled metadata 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetTable",
        "glue:GetTableVersion",
        "glue:GetTableVersions",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:BatchGetPartition",
        "glue:GetTables",
        "glue:SearchTables"
      ],
      "Principal": { "AWS": [
        "111122223333"
      ] },
      "Resource": [
        "arn:aws:glue:<region>:111122223333:table/db1/t"
      ]
    }
  ]
}
```

**See Also:**

- [GetResourcePolicies Action \(Python: `get\_resource\_policies`\)](#) in the *AWS Glue Developer Guide*

## Managing Policy Tags for Metadata Access Control

To use the tag-based access control (TBAC) method to secure Data Catalog resources (databases, tables, and columns), you create policy tags, assign them to resources, and grant them to principals.

Only data lake administrators can create, update, and delete policy tags. At first, only data lake administrators can assign policy tags to Data Catalog resources. A data lake administrator can grant permissions to other principals to assign tags to resources.

You can manage policy tags by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

**Topics**

- [Creating Policy Tags \(p. 93\)](#)
- [Updating Policy Tags \(p. 93\)](#)
- [Deleting Policy Tags \(p. 94\)](#)
- [Listing Policy Tags \(p. 95\)](#)
- [Assigning Policy Tags to Data Catalog Resources \(p. 97\)](#)
- [Viewing Policy Tags Assigned to a Resource \(p. 101\)](#)
- [Viewing the Resources That a Policy Tag Is Assigned To \(p. 103\)](#)

**See Also**

- [Granting, Revoking, and Listing Policy Tag Permissions \(p. 107\)](#)
- [Granting Data Catalog Permissions Using the TBAC Method \(p. 122\)](#)

- [Tag-Based Access Control in Lake Formation \(p. 73\)](#)

## Creating Policy Tags

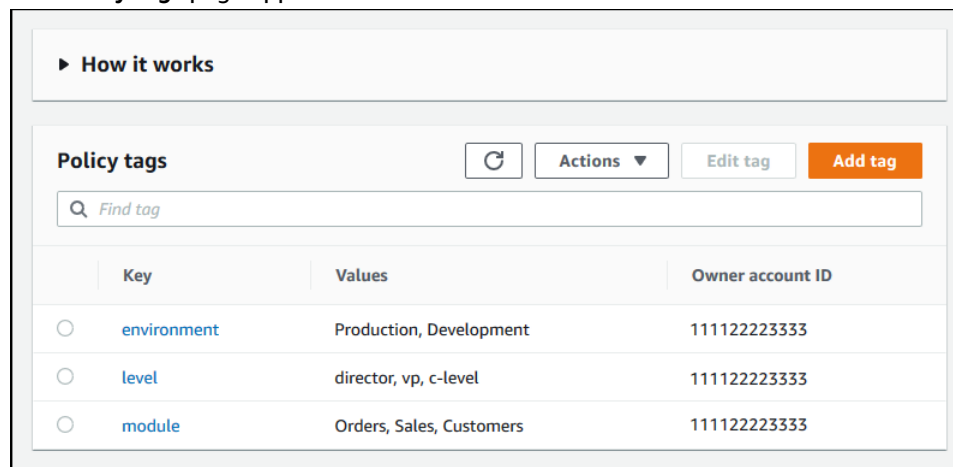
All policy tags must be defined in Lake Formation before they can be used. A policy tag consists of a key and one or more possible values for the key. Only data lake administrators can create policy tags.

You can create policy tags by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

### To create a policy 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**, choose **Policy tags**.

The **Policy tags** page appears.



3. Choose **Add tag**.
4. In the **Add policy 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, or enter one value at a time and press **Enter** or choose **Add** after each one. The maximum number of values permitted is 15.
5. Choose **Save**.

### To create a policy tag (AWS CLI)

- Enter a `create-lf-tag` command.

The following example creates a tag with key `module` and values `Customers` and `Orders`.

```
aws lakeformation create-lf-tag --tag-key module --tag-values Customers Orders
```

## Updating Policy Tags

You update a policy tag by adding or deleting permitted key values. You can't change the tag key. To change the key, delete the tag and add one with the required key.

When you delete a tag value, no check is performed for the presence of that tag value on any Data Catalog resource. If the deleted 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 tag value, you can optionally use the `remove-lf-tags-from-resource` command (p. 101) to remove the 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 can update a tag.

You can update a policy tag by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

### To update a policy 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**, choose **Policy tags**.
3. On the **Policy tags** page, select a tag, and then choose **Edit tag**.
4. In the **Edit policy tag** dialog box, add or remove tag values.

To add multiple values, in the **Values** field, either enter a comma-delimited list, or enter one value at a time and press **Enter** or choose **Add** after each one.

5. Choose **Save**.

### To update a policy 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 tag key `level`.

```
aws lakeformation update-lf-tag --tag-key level --tag-values-to-add vice-president --tag-values-to-delete vp
```

## Deleting Policy Tags

You can delete policy tags that are no longer in use. No check is performed for the presence of the tag on a Data Catalog resource. If the deleted tag is associated with a resource, it is no longer visible for the resource, and any principals that were granted permissions on that tag no longer have the permissions.

Before deleting a tag, you can optionally use the `remove-lf-tags-from-resource` command (p. 101) to remove the tag from all resources.

Only data lake administrators can delete a tag.

You can delete a tag by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

### To delete a policy 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**, choose **Policy tags**.
3. On the **Policy tags** page, select a tag, and then on the **Actions** menu, choose **Delete**.
4. In the **Delete tag?** dialog box, to confirm the deletion, enter the tag key in the designated field and press **Enter**. Then choose **Delete**.

### To delete a policy tag (AWS CLI)

- Enter a `delete-lf-tag` command. Provide the key of the tag to delete.

#### Example

The following example deletes the tag with the key `region`.

```
aws lakeformation delete-lf-tag --tag-key region
```

## Listing Policy Tags

You can list the policy tags that you have the `DESCRIBE` or `ASSOCIATE` permissions on. The values listed with each tag key are the values that you have permissions on.

Data lake administrators can see all tags that are defined in the local AWS account and all 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 tags.

You can list policy tags by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

### To list policy tags (console)

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.  
Sign in as a data lake administrator or as a principal that has been granted permissions on policy tags and that has the `lakeformation:ListLFTags` IAM permission.
2. In the navigation pane, under **Permissions**, choose **Policy tags**.

The **Policy tags** page appears.

► How it works

Policy tags

↺

Actions ▼

Edit tag

Add tag

🔍 Find tag

	Key	Values	Owner account ID
<input type="radio"/>	environment	Production, Development	111122223333
<input type="radio"/>	level	director, vp, c-level	111122223333
<input type="radio"/>	module	Orders, Sales, Customers	111122223333

Check the **Owner account ID** column to determine the tags that were shared with your account from an external account.

### To list policy tags (AWS CLI)

- Run the following command as a data lake administrator or as a principal that has been granted permissions on policy 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 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 tags and tags that were granted from external accounts. 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"
      ]
    }
  ]
}
```

### See Also

- [Lake Formation Personas and IAM Permissions Reference \(p. 193\)](#)

## Assigning Policy Tags to Data Catalog Resources

You can assign policy tags to Data Catalog resources (databases, tables, and columns) to control access to those resources. Only principals that are granted matching policy tags (and principals that are granted access with the named resource method) can access the resources.

If a table inherits a tag from a database or a column inherits a tag from a table, you can override the inherited value by assigning a new value to the tag key.

The maximum number of tags that you can assign to a resource is 50.

To assign a tag to a Data Catalog resource, you must:

- Have the Lake Formation `ASSOCIATE` permission on the tag.
- Have the IAM `lakeformation:AddLFTagsToResource` permission.
- 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: `CREATE_TABLE`, `ALTER`, and `DROP`
  - For databases in an external account: `CREATE_TABLE` and `ALTER`
  - For tables (and columns): `ALTER`, `DROP`, `INSERT`, `SELECT`, and `DELETE`

In addition, the policy tag and the resource that it is being assigned to must be in the same AWS account.

To remove a tag from a Data Catalog resource, you must meet these requirements, and also have the `lakeformation:RemoveLFTagsFromResource` IAM permission.

### To assign policy tags to a Data Catalog database or table (console)

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, do one of the following:
  - To assign policy tags to databases, choose **Databases**.
  - To assign policy tags to tables, choose **Tables**.
3. Choose a database or table, and on the **Actions** menu, choose **Edit tags**.

The **Edit policy tags <resource-name>** dialog box appears.

**Edit policy tags: inventory** [X]

**Policy tags**  
Once associated with catalog resources, policy tags allow you to create scalable access control policies.  
There are no inherited policy tags associated with the resource.

Assigned keys	Values	
<input type="text" value="module"/>	<input type="text" value="Enter policy tag value"/>	<input type="button" value="Remove"/>
<input type="button" value="Assign new policy tag"/>	orders	
<small>You can add 49 more tags</small>	customers	
	sales	

If a table has tags that are inherited from its containing database, the inherited tags are shown.



**Edit policy tags: inventory** [X]

**Policy tags**  
Once associated with catalog resources, policy tags allow you to create scalable access control policies.

**Inherited keys**      **Values**

Q region      midwest (inherited) ▼

---

**Assigned keys**      **Values**

Q module X      customers ▲      Remove

Assign new policy tag      orders

You can add 49 more tags      customers (selected)

sales

Cancel      Save

4. (Optional) If a table has inherited tags, for the **Values** list next to an **Inherited keys** field, choose a value to override the inherited value.
5. If you want to assign new policy tags, perform these steps:
  - a. Choose **Assign new policy tag**.
  - b. For **Assigned keys**, choose a tag key, and for **Values**, choose a value.
  - c. (Optional) Choose **Assign new policy tag** again to assign an additional tag.
6. Choose **Save**.

### To assign policy 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 tags**.

#### Note

If you intend to add or delete columns and save a new version, do that first. Then edit the tags.

The **Edit policy tags** dialog box appears, and displays any tags that are inherited from the table.

**Edit policy tags: 2 columns**

**Policy tags**  
Once associated with catalog resources, policy tags allow you to create scalable access control policies.

**Inherited keys**  
module

**Values**  
orders (inherited)

**Assigned keys**  
environment

**Values**  
Enter policy tag value  
development  
test  
production

**Assign new policy tag**  
You can add 49 more tags

**Remove**

**Cancel** **Save**

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 policy tag**. Then for **Assigned keys**, choose a key, and for **Values**, choose a value for the key.
8. (Optional) Choose **Assign new policy tag** again to add another tag.
9. Choose **Save**.

### To assign policy tags to a Data Catalog resource (AWS CLI)

- Run the `add-lf-tags-to-resource` command.

The following example assigns the 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 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 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
```

### To update a policy tag for a Data Catalog resource (AWS CLI)

- Use the `add-lf-tags-to-resource` command, as described in the previous procedure.

Adding a policy tag with the same key as an existing tag but with a different value updates the existing value.

### To remove a tag from a Data Catalog resource (AWS CLI)

- Run the `remove-lf-tags-from-resource` command.

If a table has a tag value that overrides the value that is inherited from the parent database, removing that 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 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
```

### See Also

- [Lake Formation Personas and IAM Permissions Reference \(p. 193\)](#)

## Viewing Policy Tags Assigned to a Resource

You can view the policy tags that are assigned to a Data Catalog resource. You must have the `DESCRIBE` or `ASSOCIATE` permission on a tag to view it.

### To view the policy 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, do one of the following:
  - To view policy tags assigned to a database, choose **Databases**.
  - To view policy 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 **Policy tags** section.

The following screenshot shows the tags assigned to a `customers` table, which is contained in the `issues` database. The `team` tag is inherited from the database. The `tier` column has the `level=vp` tag assigned.

Policy tags (4)				Edit tags
Tags are key-value pairs that you can assign to data catalog resources, such as databases, tables, and columns. You can then grant permissions to principals based on these tags to control access to the resources. Table columns inherit all policy tags that are assigned to the table. <a href="#">Learn More</a>				
<input type="text" value="Find tags"/>				
< 1 > ⚙				
Resource	Key	Value	Inherited from	
customers (table)	team	support	issues	
customers (table)	environment	Production	-	
customers (table)	module	Customers	-	
tier (column)	level	vp	-	

### To view the policy 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 tags that are explicitly assigned, not inherited. If you want to see all tags on all columns, including inherited tags, omit the `--show-assigned-lf-tags` option.

## Viewing the Resources That a Policy Tag Is Assigned To

You can view all the Data Catalog resources that a particular policy tag key is assigned to. To do so, you need the following Lake Formation permissions:

- `DESCRIBE` or `ASSOCIATE` on the 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`

### To view the resources that a policy 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 **Administrative roles**, choose **Policy tags**.
3. Choose a policy tag key (not the option button next to the key name).

The tag details page displays a list of resources that the tag has been assigned to.

**team**

**Policy tag** Delete Edit

Key	Values
team	marketing, field service, support, sales

**Associated data catalog resources (6)**

Key	Values	Resource type	Resource
team	support	DATABASE	<a href="#">issues</a>
team	support	TABLE	<a href="#">customers</a>
team	support	COLUMN	<a href="#">customers.custno</a>
team	support	COLUMN	<a href="#">customers.tier</a>
team	support	COLUMN	<a href="#">customers.company</a>
team	support	COLUMN	<a href="#">customers.region</a>

### To view the resources that a policy tag is assigned to (AWS CLI)

- 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` policy tag assigned. For each table and column listed, all assigned 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
```

### See Also

- [Lake Formation Personas and IAM Permissions Reference \(p. 193\)](#)

## Granting Lake Formation Permissions

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.

### Topics

- [IAM Permissions Required to Grant or Revoke Lake Formation Permissions \(p. 104\)](#)
- [Implicit Lake Formation Permissions \(p. 106\)](#)
- [Granting, Revoking, and Listing Policy Tag Permissions \(p. 107\)](#)
- [Granting and Revoking Data Catalog Permissions in Lake Formation \(p. 112\)](#)
- [Granting Data Location Permissions \(p. 128\)](#)
- [Lake Formation Permissions Reference \(p. 133\)](#)

### See Also:

- [Lake Formation Access Control Overview \(p. 64\)](#) (for an introduction to Lake Formation permissions)
- [Lake Formation Permissions Reference \(p. 133\)](#) (for details on each Lake Formation permission)

## 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` or `glue:GetDatabase` for a table or database that you're granting permissions on with 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.

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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lakeformation:ListPermissions",
        "lakeformation:GrantPermissions",
        "lakeformation:BatchGrantPermissions",
        "lakeformation:RevokePermissions",
        "lakeformation:BatchRevokePermissions",
        "glue:GetDatabases",
        "glue:SearchTables",
        "glue:GetTables",
        "glue:GetDatabase",
        "glue:GetTable",
        "iam:ListUsers",
        "iam:ListRoles"
      ],
      "Resource": "*"
    }
  ]
}
```

All of the `glue:` and `iam:` permissions in this policy are available in the AWS managed policy `AWSGlueConsoleFullAccess`.

To grant permissions by using tag-based access control (TBAC), principals need additional IAM permissions. For more information, see [Tag-Based Access Control Permissions Model \(p. 78\)](#) and [Lake Formation Personas and IAM Permissions Reference \(p. 193\)](#).

### 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 [the section called "Data Lake Administrator Permissions" \(p. 194\)](#).

### The Administrative User

A principal with IAM 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.

```
{
  "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.

### See Also

- [the section called “Cross-Account Access” \(p. 81\)](#)

## 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 full read access to all resources in the Data Catalog. 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 [the section called “Create a Data Lake Administrator” \(p. 8\)](#).

### 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 [the section called “Creating a Database” \(p. 43\)](#).

**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.

## Granting, Revoking, and Listing Policy Tag Permissions

You can grant the `DESCRIBE` and `ASSOCIATE` Lake Formation permissions on policy tags to principals so that they can view the policy tags and assign them to Data Catalog resources (databases, tables, and columns). When policy tags are assigned to Data Catalog resources, you can use the tag-based access control (TBAC) method to secure those resources. For more information, see [Tag-Based Access Control in Lake Formation \(p. 73\)](#).

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 [Tag-Based Access Control Permissions Model \(p. 78\)](#).

You can grant the `DESCRIBE` and `ASSOCIATE` permissions on a 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 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 policy tags by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

**Topics**

- [Listing Policy Tag Permissions Using the Console \(p. 107\)](#)
- [Granting Policy Tag Permissions Using the Console \(p. 108\)](#)
- [Granting, Revoking, and Listing Policy Tag Permissions Using the AWS CLI \(p. 110\)](#)

**See Also**

- [Managing Policy Tags for Metadata Access Control \(p. 92\)](#)
- [Tag-Based Access Control in Lake Formation \(p. 73\)](#)

## Listing Policy Tag Permissions Using the Console

You can use the Lake Formation console to view the permissions granted on policy tags. You must be a data lake administrator or have the `DESCRIBE` or `ASSOCIATE` permission on a tag to see it.

**To list policy tag permissions (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 to whom the `ASSOCIATE` or `DESCRIBE` permissions on policy tags have been granted.

2. In the navigation pane, under **Permissions**, choose **Tag permissions**.

The **Tag permissions** page appears.

Tag permissions						
<input type="text" value="Find permissions by tag"/>						
	Principal	Principal type	Keys	values	Permissions	Grantable
<input type="radio"/>	arn:aws:iam::11112223333:user/datalake_admin	IAM User	environment	All values	DESCRIBE	DESCRIBE
<input type="radio"/>	arn:aws:iam::11112223333:user/datalake_admin	IAM User	environment	All values	ASSOCIATE	ASSOCIATE
<input type="radio"/>	arn:aws:iam::11112223333:user/datalake_user1	IAM User	module	Orders, Sales	ASSOCIATE	
<input type="radio"/>	arn:aws:iam::11112223333:user/datalake_admin	IAM User	module	All values	DESCRIBE	DESCRIBE
<input type="radio"/>	arn:aws:iam::11112223333:user/datalake_admin	IAM User	module	All values	ASSOCIATE	ASSOCIATE

## Granting Policy Tag Permissions Using the Console

The following steps explain how to grant permissions on tags by using the **Grant tag permissions** page on the Lake Formation console. The page is divided into these sections:

- **Principals** – The users, roles, or AWS accounts to grant permissions to.
- **Policy tags** – The tags to grant permissions on.
- **Permissions** – The permissions to grant.

### Open the Grant tag permissions page

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.

Sign in as a data lake administrator or as a user to whom the `ASSOCIATE` or `DESCRIBE` permissions on policy tags have been granted with the grant option.

2. In the navigation pane, under **Permissions**, choose **Tag permissions**.
3. Choose **Grant**.

### Specify the Principals

In the **Principals** section, choose a principal type and specify principals to grant permissions to.

#### Principals

☒ **IAM users and roles**  
Users or roles from this AWS account.

☐ **SAML users and groups**  
SAML users and group or QuickSight ARNs.

☐ **External accounts**  
AWS accounts or AWS organizations outside of this account.

**IAM users and roles**  
Add one or more IAM users or roles.

### 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 QuickSight users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon QuickSight 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 \(p. 133\)](#).

#### Note

Lake Formation integration with Amazon QuickSight is supported for Amazon QuickSight Enterprise Edition only.

### External accounts

For **AWS account**, enter one or more valid AWS account IDs. Press **Enter** after each ID.

#### Note

Currently, granting tag permissions to organizations and organizational units isn't supported for TBAC.

## Specify the Policy Tags

In the **Policy tags** section, specify the policy tags to grant permissions on.

▼ **Policy tags**

Tag permission scope  
Choose to grant permissions on all or a subset of policy tags.

Key	Values	
<input type="text" value="module"/>	<input type="text" value="Choose tag values"/>	<input type="button" value="Remove"/>
	<input type="button" value="Orders"/> <input type="button" value="Sales"/>	
<input type="text" value="Enter a tag key"/>	<input type="text" value="Choose tag values"/>	<input type="button" value="Remove"/>

1. Choose **Add policy tag** to reveal the first row of fields for specifying a tag.
2. Position the cursor in the **Key** field, optionally start typing to narrow down the selection list, and select a 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 tag.

4. To add another tag, choose **Add policy tag** again, and repeat the previous two steps.

## Specify the Permissions

In the **Permissions** section, select permissions and grantable permissions.

**▼ Permissions**

**Tag permissions**  
Select the specific access permissions to grant.

☐ Describe ☐ Associate

**Grantable permissions**  
Select the permissions that the grant recipient can grant to other principals.

☐ Describe ☐ Associate

1. Under **Tag permissions**, select the permissions to grant.  
Granting **Associate** implicitly grants **Describe**.
2. (Optional) Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account.
3. Choose **Grant**.

## Granting, Revoking, and Listing Policy Tag Permissions Using the AWS CLI

You can grant, revoke, and list permissions on policy tags by using the AWS Command Line Interface (AWS CLI).

### To list policy tag permissions (AWS CLI)

- Enter a `list-permissions` command. You must be a data lake administrator or have the `DESCRIBE` or `ASSOCIATE` permission on a tag to see it.

The following command requests all 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 tags granted to all principals. Non-administrative users see only tags granted to them. 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 tag key. The following command returns all permissions granted on the tag module.

```
aws lakeformation list-permissions --resource-type LF_TAG --resource '{ "LFTag":
{"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

You can also list tag values granted to a specific principal for a specific 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 tag key. The following command shows how to do this for the principal `datalake_user1` and the 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 policy tags (AWS CLI)

- Enter a command similar to the following. This example grants to user `datalake_user1` the `ASSOCIATE` permission on the 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 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"]}}'

```

### To revoke permissions on policy tags (AWS CLI)

- Enter a command similar to the following. This example revokes the `ASSOCIATE` permission on the 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 and Revoking Data Catalog Permissions in Lake Formation

You can grant Data Catalog 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 Catalog permissions on both metadata databases and metadata tables. When you grant permissions on tables, you can limit access to specific table columns for even more fine-grained access control.

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.

You can grant permissions by using either the named resource access control method or the tag-based access control (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 resources that you own with those accounts or organizations. Principals in those accounts or organizations can then access Data Catalog resources that you own and the underlying data.

**Note**

Currently, the TBAC method supports granting cross-account permissions to AWS accounts only, not to organizations or organizational units.

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 resources until the administrator grants permissions on the shared resources 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).

**Topics**

- [Granting Data Catalog Permissions Using the Named Resource Method \(p. 113\)](#)
- [Granting Data Catalog Permissions Using the TBAC Method \(p. 122\)](#)
- [Granting Resource Link Permissions \(p. 126\)](#)
- [Granting Permissions on a Database or Table Shared with Your Account \(p. 127\)](#)

**See Also:**

- [Sharing Data Catalog Tables and Databases Across AWS Accounts \(p. 45\)](#)
- [Metadata Access Control \(p. 67\)](#)
- [Lake Formation Permissions Reference \(p. 133\)](#)

## Granting Data Catalog Permissions Using the Named Resource Method

You can use the named resource method to grant Lake Formation permissions on specific Data Catalog databases and tables. You can grant permissions by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

**Topics**

- [Granting Database Permissions Using the Lake Formation Console and the Named Resource Method \(p. 113\)](#)
- [Granting Database Permissions Using the AWS CLI and the Name Resources Method \(p. 116\)](#)
- [Granting Table Permissions Using the Lake Formation Console and the Named Resource Method \(p. 117\)](#)
- [Granting Table Permissions Using the AWS CLI and the named resource Method \(p. 121\)](#)

### Granting Database Permissions Using the Lake Formation Console and the Named Resource Method

The following steps explain how to grant database permissions by using the named resource method and the **Grant Permissions** page on the Lake Formation console. The page is divided into the following sections:

- **Principals** – The users, roles, AWS accounts, organizations, or organizational units to grant permissions to.
- **Policy 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 database resource link, see [Granting Resource Link Permissions \(p. 126\)](#).

### 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 a user who has been granted Lake Formation permissions on the database with the grant option.
2. Do one of the following:
  - In the navigation pane, choose **Data permissions**. Then choose **Grant**.
  - In the navigation pane, choose **Databases**. Then, on the **Databases** page, choose a database, and on 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 \(p. 49\)](#).

### Specify the Principals

In the **Principals** section, choose a principal type and then specify principals to grant permissions to.

**Principals**

☒ **IAM users and roles**  
Users or roles from this AWS account.

☐ **SAML users and groups**  
SAML users and group or QuickSight ARNs.

☐ **External accounts**  
AWS accounts or AWS organizations outside of this account.

**IAM users and roles**  
Add one or more IAM users or roles.

#### 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 QuickSight users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon QuickSight 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 \(p. 133\)](#).

#### Note

Lake Formation integration with Amazon QuickSight is supported for Amazon QuickSight Enterprise Edition only.



## External accounts

For **AWS account** or **AWS organization**, enter one or more valid AWS account IDs, organization IDs, or organizational unit IDs. 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 \(p. 45\)](#)

## Specify the Databases

In the **Policy tags or catalog resources** section, choose one or more databases to grant permissions on.

1. Choose **Named data catalog resources**.

**Policy tags or catalog resources**

☐ Resources matched by policy tags  
(recommended)  
Manage permissions indirectly for resources or data matched by a specific set of policy tags.

☒ Named data catalog resources  
Manage permissions for specific databases or tables, in addition to fine-grained data access.

**Database**  
Add one or more databases.

Choose databases ▼

retail X

**Table- optional**  
Add one or more tables.

Choose tables ▼

2. Choose one or more databases from the **Database** list.

## Specify the Permissions

In the **Permissions** section, select permissions and grantable permissions.

**Permissions**  
Select the permissions to grant.

☒ **Database permissions**  
Grant resource-wide permissions.

☐ **Column-based permissions**  
Grant data access to specific columns.

---

**Database permissions**  
Choose specific access permissions to grant.

☐ Create Table    ☐ Alter    ☐ Drop    ☐ Describe

☐ **Super**  
This permission is the union of the individual permissions above and supercedes them. [Learn More](#)

**Grantable permissions**  
Choose the permission that may be granted to others.

☐ Create Table    ☐ Alter    ☐ Drop    ☐ Describe

☐ **Super**  
This permission is the union of the individual permissions above and supercedes them. [Learn More](#)

1. 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 \(p. 128\)](#).

2. (Optional) Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account.
3. Choose **Grant**.

**See Also**

- [Lake Formation Permissions Reference \(p. 133\)](#)
- [Granting Permissions on a Database or Table Shared with Your Account \(p. 127\)](#)

## Granting Database Permissions Using the AWS CLI and the Name Resources Method

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" } }'
```

### 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 \(p. 128\)](#).

### See Also

- [Lake Formation Permissions Reference \(p. 133\)](#)
- [Granting Permissions on a Database or Table Shared with Your Account \(p. 127\)](#)

## Granting Table Permissions Using the Lake Formation Console and the Named Resource Method

You can use the Lake Formation console and the named resource method 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.

The following steps explain how to grant table permissions by using the named resource method and the **Grant Permissions** page on the Lake Formation console. The page is divided into these sections:

- **Principals** – The users, roles, AWS accounts, organizations, or organizational units to grant permissions to.
- **Policy 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 \(p. 126\)](#).

### 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 table creator, or a user who has been granted permissions on the table with the grant option.
2. Do one of the following:
  - In the navigation pane, choose **Data 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 \(p. 49\)](#).

### Specify the Principals

In the **Principals** section, choose a principal type and specify principals to grant permissions to.

**Principals**

☒ **IAM users and roles**  
Users or roles from this AWS account.

☐ **SAML users and groups**  
SAML users and group or QuickSight ARNs.

☐ **External accounts**  
AWS accounts or AWS organizations outside of this account.

**IAM users and roles**  
Add one or more IAM users or roles.

Choose IAM principals to add ▼

#### 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 QuickSight users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon QuickSight 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 \(p. 133\)](#).

**Note**

Lake Formation integration with Amazon QuickSight is supported for Amazon QuickSight Enterprise Edition only.

## External accounts

For **AWS account** or **AWS organization**, enter one or more valid AWS account IDs, organization IDs, or organizational unit IDs. 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 \(p. 45\)](#)

## Specify the Tables

In the **Policy tags or catalog resources** section, choose a database. Then choose one or more tables, or **All tables**.

**Policy tags or catalog resources**

☐ Resources matched by policy tags (recommended)  
Manage permissions indirectly for resources or data matched by a specific set of policy tags.

☒ Named data catalog resources  
Manage permissions for specific databases or tables, in addition to fine-grained data access.

**Database**  
Add one or more databases.

Choose databases ▼

retail ✕

**Table- optional**  
Add one or more tables.

Choose tables ▼

inventory ✕  
No description available

## Specify the Permissions

In the **Permissions** section, select permissions and grantable permissions.

There are two types of permissions that you can grant:

- **Table permissions** – Table permissions include **Alter**, **Insert**, **Drop**, **Delete**, **Describe**, **Select**, and **Super**. The **Select** permission grants read access to all columns in the table.

**Permissions**  
Select the permissions to grant.

☒ **Table permissions**  
Grant resource-wide permissions.

☐ **Column-based permissions**  
Grant data access to specific columns.

---

**Table permissions**  
Choose specific access permissions to grant.

☐ Alter   ☐ Insert   ☐ Drop   ☐ Delete   ☐ Select   ☐ Describe

☐ **Super**  
This permission is the union of the individual permissions above and supercedes them. [Learn More](#)

**Grantable permissions**  
Choose the permission that may be granted to others.

☐ Alter   ☐ Insert   ☐ Drop   ☐ Delete   ☐ Select   ☐ Describe

☐ **Super**  
This permission is the union of the individual permissions above and supercedes them. [Learn More](#)

- **Column-based permissions** – Column permissions grant access to only a subset of table columns ("column filtering"). You can specify the subset with an inclusion list or an exclusion list. If you include the grant option, the grant recipient can grant permissions only on the columns that you grant to them. The only permission that you can grant when granting column-based permissions is the `Select` permission.

**Permissions**  
Select the permissions to grant.

☐ **Table permissions**  
Grant resource-wide permissions.

☒ **Column-based permissions**  
Grant data access to specific columns.

---

**Choose permission filter**  
Choose whether to include or exclude columns.

☒ **Include columns**  
Grant permissions to access specific columns.

☐ **Exclude columns**  
Grant permissions to access all but specific columns.

**Select columns**

**Grantable permissions**  
Choose the permission that may be granted to others.

☐ **Select**

The following are the rules for granting these permissions using the Lake Formation console:

- To grant `Select` on all columns, or to grant other table permissions such as `Insert`, in the **Permissions** section, choose the **Table permissions** option. Then, do the following:
  1. Under **Table permissions**, select the permissions to grant.

2. (Optional) Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account.
3. Choose **Grant**.

**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 \(p. 128\)](#).

- To grant only `Select` with column filtering, in the **Permissions** section, choose **Column-based permissions** and specify the columns to include or exclude. Then choose **Grant**.

**Important**

If you grant column permissions, don't grant the `Select` permission under the **Table permissions** option. That permission grants access to all columns.

**See Also**

- [Lake Formation Permissions Reference \(p. 133\)](#)
- [Granting Permissions on a Database or Table Shared with Your Account \(p. 127\)](#)

## Granting Table Permissions Using the AWS CLI and the named resource Method

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

The following example grants `SELECT` and `ALTER` to 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" "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 \(p. 128\)](#).

#### Example – Grant on All Tables with the Grant Option

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 Column Filtering

This next example grants `SELECT` on a subset of columns in the table `persons`.

```
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" ] } }'
```

### See Also

- [Lake Formation Permissions Reference \(p. 133\)](#)
- [Granting Permissions on a Database or Table Shared with Your Account \(p. 127\)](#)

## Granting Data Catalog Permissions Using the TBAC Method

You can use the tag-based access control (TBAC) method to grant Lake Formation permissions on Data Catalog databases, tables, and columns.

You can grant permissions by using the AWS Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

### Topics

- [Granting Data Catalog Permissions Using the Lake Formation Console and the TBAC Method \(p. 122\)](#)
- [Granting Data Catalog Permissions Using the AWS CLI and the TBAC Method \(p. 125\)](#)

### See Also

- [Granting, Revoking, and Listing Policy Tag Permissions \(p. 107\)](#)
- [Managing Policy Tags for Metadata Access Control \(p. 92\)](#)
- [Tag-Based Access Control in Lake Formation \(p. 73\)](#)

## Granting Data Catalog Permissions Using the Lake Formation Console and the TBAC Method

The following steps explain how to grant permissions by using the tag-based access control (TBAC) method and the **Grant 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.
- **Policy tags or catalog resources** – The databases, tables, or resource links to grant permissions on.
- **Permissions** – The Lake Formation 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 or as a user who has been granted Lake Formation permissions on Data Catalog resources through TBAC with the grant option.
2. In the navigation pane, choose **Data permissions**. Then choose **Grant**.

### Specify the Principals

In the **Principals** section, choose a principal type and then specify principals to grant permissions to.



**Principals**

☒ **IAM users and roles**  
Users or roles from this AWS account.

☐ **SAML users and groups**  
SAML users and group or QuickSight ARNs.

☐ **External accounts**  
AWS accounts or AWS organizations outside of this account.

**IAM users and roles**  
Add one or more IAM users or roles.

Choose IAM principals to add ▼

### **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 QuickSight users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML, or ARNs for Amazon QuickSight 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](#) (p. 133).

#### **Note**

Lake Formation integration with Amazon QuickSight is supported for Amazon QuickSight Enterprise Edition only.

### **External accounts**

For **AWS account or AWS organization**, enter one or more valid AWS account IDs.

#### **Note**

Currently, the TBAC method supports granting cross-account permissions to AWS accounts only, not to organizations or organizational units.

#### **See Also**

- [Accessing and Viewing Shared Data Catalog Tables and Databases](#) (p. 45)

### **Specify the Policy Tags**

1. Ensure that the **Resources matched by policy tags** option is chosen.
2. Choose **Add policy tag**.
3. Choose a tag key and values.

If you choose more than one value, you are creating a tag expression with an OR operator. This means that if any of the tag values match a tag assigned to a Data Catalog resource, you are granted permissions on the resource.

**Policy tags or catalog resources**

☒ **Resources matched by policy tags (recommended)**  
Manage permissions indirectly for resources or data matched by a specific set of policy tags.

☐ **Named data catalog resources**  
Manage permissions for specific databases or tables, in addition to fine-grained data access.

**Key**  
Q module X

**Values**  
Choose tag values ▲

☐ orders  
☐ customers  
☐ sales

Add policy tag Remove

4. (Optional) Choose **Add policy tag** again to specify another policy tag.

If you specify more than one tag, you are creating a tag expression with an AND operator. The principal is granted permissions on a Data Catalog resource only if the resource was assigned a matching policy tag for each tag in the tag expression.

## Specify the Permissions

Specify the permissions that are granted to the principal on matching Data Catalog resources. Matching resources are those resources that were assigned policy tags that match one of the policy tag expressions granted to the principal.

You can specify the permissions to grant on matching databases, matching tables, or both.

**▼ Database permissions**

**Database permissions**  
Choose specific access permissions to grant.

☐ Create table ☐ Alter ☐ Drop ☐ Super  
This permission is the union of all the individual permissions to the left, and supersedes them.

☐ Describe

**Grantable permissions**  
Choose the permission that may be granted to others.

☐ Create table ☐ Alter ☐ Drop ☐ Super  
This permission allows the principal to grant any of the permissions to the left, and supersedes those grantable permissions.

☐ Describe

**▼ Table permissions**

**Table permissions**  
Choose specific access permissions to grant.

☐ Alter ☐ Insert ☐ Drop ☐ Super  
This permission is the union of all the individual permissions to the left, and supersedes them.

☐ Delete ☐ Select ☐ Describe

**Grantable permissions**  
Choose the permission that may be granted to others.

☐ Alter ☐ Insert ☐ Drop ☐ Super  
This permission allows the principal to grant any of the permissions to the left, and supersedes those grantable permissions.

☐ Delete ☐ Select ☐ Describe

1. Under **Database permissions**, select the database permissions to grant to the principal on matching databases.
2. Under **Table permissions**, select the table permissions to grant to the principal on matching tables.
3. Choose **Grant**.

### Granting Data Catalog Permissions Using the AWS CLI and the TBAC Method

You can use the AWS Command Line Interface (AWS CLI) and the tag-based access control (TBAC) method to grant Lake Formation permissions on Data Catalog databases, tables, and columns.

#### To grant Data Catalog permissions using TBAC (AWS CLI)

- Use the `grant-permissions` command.

##### Example

The following example grants the policy tag expression `"module=*" (all values of the 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 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 policy 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 policy 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 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"]}]]}'
```

## 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).

### To grant resource link permissions (console)

1. Do one of the following:
  - For database resource links, follow the steps in [Granting Database Permissions Using the Lake Formation Console and the Named Resource Method \(p. 113\)](#) to do the following:
    1. [Open the Grant Permissions Page \(p. 114\)](#).
    2. [Specify the Databases \(p. 115\)](#). Specify one or more database resource links.
    3. [Specify the Principals \(p. 114\)](#).
  - For table resource links, follow the steps in [Granting Table Permissions Using the Lake Formation Console and the Named Resource Method \(p. 117\)](#) to do the following:
    1. [Open the Grant Permissions Page \(p. 118\)](#).
    2. [Specify the Tables \(p. 119\)](#). Specify one or more table resource links.
    3. [Specify the Principals \(p. 118\)](#).
2. Under **Permissions**, select the permissions to grant. Optionally, select grantable permissions.

**Permissions**  
Select the permissions to grant.

☒ **Resource link permissions**  
Grant resource-wide permissions.

☐ **Column-based permissions**  
Grant data access to specific columns.

---

**Resource link permissions**  
Choose specific access permissions to grant.

☐ Drop ☐ Describe

☐ Super  
This permission is the union of the individual permissions above and supercedes them. [Learn More](#)

**Grantable permissions**  
Choose the permission that may be granted to others.

☐ Drop ☐ Describe

☐ Super  
This permission is the union of the individual permissions above and supercedes them. [Learn More](#)

3. Choose **Grant**.

### To grant resource link permissions (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 \(p. 49\)](#)
- [Lake Formation Permissions Reference \(p. 133\)](#)

## 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 Lake Formation Console and the Named Resource Method \(p. 113\)](#). In the **Database** list under **Policy 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 \(p. 46\)](#).

Also, for the `CREATE_TABLE` and `ALTER` permissions, follow the instructions in [Granting Data Location Permissions \(Same Account\) \(p. 129\)](#), 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 Lake Formation Console and the Named Resource Method \(p. 117\)](#). In the **Database** list under **Policy 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 \(p. 46\)](#).

Also, for the `ALTER` permission, follow the instructions in [Granting Data Location Permissions \(Same Account\)](#) (p. 129), and be sure to enter the owning account ID in the **Registered account location** field.

### To grant permissions on shared resources (TBAC method, console)

- Follow the instructions in [Granting Data Catalog Permissions Using the Lake Formation Console and the TBAC Method](#) (p. 122). In the **Policy tags or catalog resources** section, grant the exact tag expression that the external account granted to your account, or a subset of that expression.

For example, if an external account granted the 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 (e.g. `SELECT`, `ALTER`, etc.) that were granted on resources through the 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 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](#) (p. 70)

## Topics

- [Granting Data Location Permissions \(Same Account\) \(p. 129\)](#)
- [Granting Data Location Permissions \(External Account\) \(p. 131\)](#)
- [Granting Permissions on a Data Location Shared with Your Account \(p. 132\)](#)

## 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).

### To grant data location permissions (same account, 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 principal who has grant permissions on the desired data location.
2. In the navigation pane, 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 QuickSight users and groups**, enter one or more Amazon Resource Names (ARNs) for users or groups federated through SAML or ARNs for Amazon QuickSight users or groups.  
  
Enter one ARN at a time, and press **Enter** after each ARN. For information about how to construct the ARNs, see [the section called "Lake Formation Grant and Revoke AWS CLI Commands" \(p. 133\)](#).
  - 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**.

**Grant permissions** ✕

Add access permissions for specific storage locations.

☒ **My account**  
User or role from this AWS account.

☐ **External account**  
AWS account or AWS organization outside of my account.

**IAM users and roles**  
Add one or more IAM users or roles.

Choose IAM principals to add ▼

**datalake\_user** ✕  
User

**SAML and Amazon QuickSight users and groups**  
Enter a SAML user or group ARN or Amazon QuickSight ARN. Press Enter to add additional ARNs.

Ex: `arn:aws:iam::<AccountId>:saml-provider/<SamlProviderName>`

**Storage locations**  
Choose one or more data lake locations.

`s3://retail/transactions/2020q1` Browse

**Registered account location**  
The account where this storage location is registered in AWS Lake Formation.

`123456789012`

☐ Grantable

Cancel Grant

5. Choose **Grant**.

### To grant data location permissions (same account, AWS CLI)

- 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" } }'
```

#### See Also:

- [the section called “Lake Formation Permissions Reference” \(p. 133\)](#)



## 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 [the section called "Cross-Account Access Prerequisites"](#) (p. 82).

### 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, 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 "-" dash 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.

**Grant permissions** ×

Add access permissions for specific storage locations.

☐ My account  
User or role from this AWS account.

☒ External account  
AWS account or AWS organization outside of my account.

**AWS account ID or AWS organization ID**

111122223333 ×  
Account

Enter one or more AWS account IDs or AWS organization IDs. Press Enter after each ID.

**Storage locations**  
Choose one or more data lake locations.

Browse

☒ Grantable

Cancel Grant

5. Select **Grantable**.

## 6. Choose **Grant**.

### 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":"s3: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":"s3: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.

### See Also:

- [the section called “Lake Formation Permissions Reference” \(p. 133\)](#)

## 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 \(Same Account\) \(p. 129\)](#).

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>" } }'
```

## 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 “Lake Formation Access Control Overview” \(p. 64\)](#). 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 and Revoking Data Catalog Permissions” \(p. 112\)](#) and [the section called “Granting Data Location Permissions” \(p. 128\)](#).

#### 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 “Granting Lake Formation Permissions” \(p. 104\)](#).

#### Topics

- [Lake Formation Grant and Revoke AWS CLI Commands \(p. 133\)](#)
- [ALTER \(p. 136\)](#)
- [CREATE\\_DATABASE \(p. 137\)](#)
- [CREATE\\_TABLE \(p. 138\)](#)
- [DATA\\_LOCATION\\_ACCESS \(p. 139\)](#)
- [DELETE \(p. 139\)](#)
- [DESCRIBE \(p. 140\)](#)
- [DROP \(p. 141\)](#)
- [INSERT \(p. 141\)](#)
- [SELECT \(p. 142\)](#)
- [Super \(p. 143\)](#)

## 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 QuickSight user or group
- For cross-account permissions, the ARN for an AWS account ID, organization ID, or organizational unit ID

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 QuickSight 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 QuickSight 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
```

## ALTER

Permission	Granted on This Resource	Grantee Also Needs
ALTER	DATABASE	glue:UpdateDatabase
ALTER	TABLE	glue:UpdateTable

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.	<code>CREATE_DATABASE</code> 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.



Create Table Use Case	Permissions Needed
The specified location is managed by Lake Formation (is registered), and the containing 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 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 \(p. 70\)](#).

## DELETE

Permission	Granted on This Resource	Grantee Also Needs
DELETE	TABLE	(No additional IAM permissions are needed if the location is registered.)

A principal with this permission can delete 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	Resource link	glue:GetTable to grant <code>DESCRIBE</code> on a table resource link, and glue:GetDatabase to grant <code>DESCRIBE</code> on a database resource link.
DESCRIBE	DATABASE	glue:GetDatabase to grant <code>DESCRIBE</code> on a database.
DESCRIBE	TABLE	glue:GetTable to grant <code>DESCRIBE</code> on a table.

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	Resource link	glue:DeleteDatabase to drop a database resource link, and glue:DeleteTable to drop a table resource link.

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	<ul style="list-style-type: none"> <li>TABLE</li> </ul>	(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.

Principal	Principal type	Resource type	Resource	Owner account ID	Permissions
datalake_user3	IAM user	Table	inventory	111122223333	Insert
datalake_user3	IAM user	Column	retail.inventory.*	111122223333	Select
datalake_user2	AD user	Table	inventory	111122223333	Delete, Insert
datalake_user2	AD user	Column	retail.inventory.*	111122223333	Select

## Super

Permission	Granted on This Resource	Grantee Also Needs
Super	DATABASE	glue:*Database*
Super	TABLE	glue:*Table*, glue:*Partition*

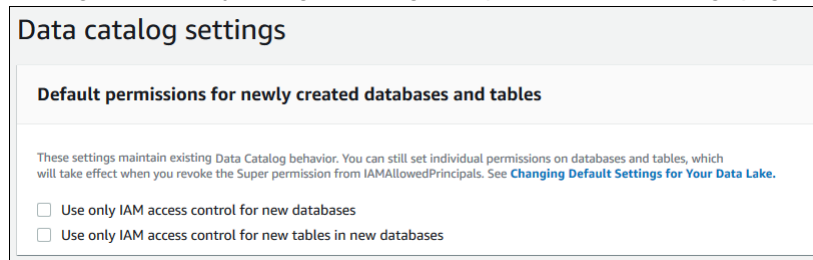
This permission allows a principal to perform every supported Lake Formation operation on the database or table. You can't grant `Super` on a database to an external account.

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 cause 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 Security Settings for Your Data Lake” \(p. 147\)](#)
- [Upgrading AWS Glue Data Permissions to the Lake Formation Model \(p. 156\)](#)

## 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 `glue: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 tiles 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 at the top of the page, 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 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**.

Principal	Principal type	Resource type	Resource	Owner account ID	Permissions	Grantable
Administrator	IAM user	Table	alexa-logs	111122223333	Super	Super

### 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 at the top of the page, 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" } }'
```

## 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.

Principal	AWS Managed Policy	Comments
Lake Formation user, including the data lake administrator	<code>AWSGlueConsoleFullAccess</code>	Allows the principal to conduct a variety of operations on the Lake Formation console.
Data lake administrators	<code>AWSLakeFormationDataAdmin</code>	Allows the data lake administrator to conduct administrative operations and view AWS CloudTrail logs.
Lake Formation user, including the data lake administrator	<code>AWSLakeFormationCrossAccountManager</code>	Allows the principal to grant Lake Formation permissions to external AWS accounts, to organizations, or to organizational units.

### 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 [the section called "Create a Data Lake Administrator" \(p. 8\)](#) and [the section called "Using Service-Linked Roles" \(p. 150\)](#).

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.

## Changing the Default Security 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 [the section called “Change Data Catalog Settings” \(p. 11\)](#).
2. Change the settings for existing Data Catalog resources. For instructions, see [Upgrading AWS Glue Data Permissions to the AWS Lake Formation Model \(p. 156\)](#).

### Changing the Default Security Settings Using the Lake Formation `PutDataLakeSettings` API Operation

You can also change default security settings by using the Lake Formation [the section called “PutDataLakeSettings \(put\\_data\\_lake\\_settings\)” \(p. 177\)](#). This action takes as arguments an optional catalog ID and a [the section called “DataLakeSettings” \(p. 176\)](#).

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`.

## 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.

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])
```

## 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 \(p. 153\)](#).

## 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

## 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.

Try this: 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`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LakeFormationDataAccessPermissionsForS3",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3::my-kinesis-test/logs/*"
      ]
    },
    {
      "Sid": "LakeFormationDataAccessPermissionsForS3ListBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3::my-kinesis-test"
      ]
    }
  ]
}
```

The following permissions are required to be able to register locations with this service-linked role:

- `iam:CreateServiceLinkedRole`
- `iam:PutRolePolicy`

The data lake administrator typically has these permissions.

# AWS Glue Features in Lake Formation

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*.

# 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": "8d2cce0-55f3-42d3-9ede-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": "AROAGGFTBBBGOBWV2EMZA: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"  
  },  
...  
}
```

### See Also

- [Cross-Account CloudTrail Logging \(p. 86\)](#)

# 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. It restricts the data that your users and those services can access via Lake Formation functionality. By comparison, the AWS Glue model grants data access via [fine-grained access control](#) IAM permissions. To make the switch, follow the steps in this guide.

For more information, see [the section called “Lake Formation Access Control Overview” \(p. 64\)](#).

## Topics

- [About Upgrading to the Lake Formation Permissions Model \(p. 156\)](#)
- [Step 1: List Users' and Roles' Existing Permissions \(p. 157\)](#)
- [Step 2: Set Up Equivalent Lake Formation Permissions \(p. 158\)](#)
- [Step 3: Give Users IAM Permissions to Use Lake Formation \(p. 159\)](#)
- [Step 4: Switch Your Data Stores to the Lake Formation Permissions Model \(p. 159\)](#)
- [Step 5: Secure New Data Catalog Resources \(p. 161\)](#)
- [Step 6: Give Users a New IAM Policy for Future Data Lake Access \(p. 162\)](#)
- [Step 7: Clean Up Existing IAM Policies \(p. 162\)](#)

## About Upgrading to the Lake Formation Permissions Model

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 start using Lake Formation to manage access to your data by registering the locations of existing Data Catalog resources in Lake Formation. 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. However, they can't query the data itself unless you register the Amazon S3 location with Lake Formation.

Follow the steps in this section to upgrade to the Lake Formation permissions model. Start with [the section called "Step 1: List Existing Permissions" \(p. 157\)](#).

## 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 [Setting Up AWS Lake Formation \(p. 6\)](#).

### Topics

- [Using the API \(p. 157\)](#)
- [Using the AWS Management Console \(p. 158\)](#)
- [Using AWS CloudTrail \(p. 158\)](#)

## Using the API

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](#) (p. 158).

## Step 2: Set Up Equivalent Lake Formation Permissions

Using the information collected in [Step 1: List Users' and Roles' Existing Permissions](#) (p. 157), grant AWS Lake Formation permissions to match the AWS Glue permissions. Use any of the following methods to perform the grants:

- Use the Lake Formation console or the AWS CLI.

See [the section called "Granting and Revoking Data Catalog Permissions"](#) (p. 112).

- Use the `GrantPermissions` or `BatchGrantPermissions` API operations.

See [Permissions APIs](#) (p. 165).

For more information, see [Granting Lake Formation Permissions](#) (p. 104).

After setting up Lake Formation permissions, proceed to [Step 3: Give Users IAM Permissions to Use Lake Formation](#) (p. 159).

## 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`.

```
{
  "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](#) (p. 159).

## 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](#) (p. 159)
- [Secure Existing Data Catalog Resources](#) (p. 160)
- [Turn On Lake Formation Permissions for Your Amazon S3 Location](#) (p. 161)

## 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" \(p. 133\)](#).

### 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 \(p. 32\)](#). Or, use the `RegisterResource` API operation as described in [Credential Vending API \(p. 178\)](#).

#### 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 \(p. 161\)](#).

## 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" \(p. 133\)](#).

#### 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](#) (p. 162).

## 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`.

```
{
  "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 [Lake Formation Personas and IAM Permissions Reference](#) (p. 193).

Next, proceed to [Step 7: Clean Up Existing IAM Policies](#) (p. 162).

## 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 API

## Contents

- [Permissions APIs \(p. 165\)](#)
  - [Data Types \(p. 165\)](#)
  - [Resource Structure \(p. 166\)](#)
  - [DatabaseResource Structure \(p. 166\)](#)
  - [TableResource Structure \(p. 167\)](#)
  - [TableWithColumnsResource Structure \(p. 167\)](#)
  - [DataLocationResource Structure \(p. 168\)](#)
  - [DataLakePrincipal Structure \(p. 168\)](#)
  - [ResourcePermissions Structure \(p. 168\)](#)
  - [ResourcePermissionsError Structure \(p. 168\)](#)
  - [PrincipalResourcePermissions Structure \(p. 169\)](#)
  - [DetailsMap Structure \(p. 169\)](#)
  - [PrincipalResourcePermissionsError Structure \(p. 169\)](#)
  - [ColumnWildcard Structure \(p. 170\)](#)
  - [BatchPermissionsRequestEntry Structure \(p. 170\)](#)
  - [BatchPermissionsFailureEntry Structure \(p. 170\)](#)
  - [PrincipalPermissions Structure \(p. 170\)](#)
  - [Operations \(p. 171\)](#)
  - [GrantPermissions Action \(Python: grant\\_permissions\) \(p. 171\)](#)
  - [RevokePermissions Action \(Python: revoke\\_permissions\) \(p. 172\)](#)
  - [BatchGrantPermissions Action \(Python: batch\\_grant\\_permissions\) \(p. 172\)](#)
  - [BatchRevokePermissions Action \(Python: batch\\_revoke\\_permissions\) \(p. 173\)](#)
  - [GetEffectivePermissionsForPath Action \(Python: get\\_effective\\_permissions\\_for\\_path\) \(p. 174\)](#)
  - [ListPermissions Action \(Python: list\\_permissions\) \(p. 174\)](#)
- [Data Lake Settings APIs \(p. 175\)](#)
  - [Data Types \(p. 175\)](#)
  - [DataLakeSettings Structure \(p. 176\)](#)
  - [Operations \(p. 177\)](#)
  - [GetDataLakeSettings Action \(Python: get\\_data\\_lake\\_settings\) \(p. 177\)](#)
  - [PutDataLakeSettings Action \(Python: put\\_data\\_lake\\_settings\) \(p. 177\)](#)
- [Credential Vending API \(p. 178\)](#)
  - [Data Types \(p. 178\)](#)
  - [FilterCondition Structure \(p. 178\)](#)
  - [ColumnNames list \(p. 179\)](#)
  - [ResourceInfo Structure \(p. 179\)](#)
  - [Operations \(p. 179\)](#)
  - [RegisterResource Action \(Python: register\\_resource\) \(p. 179\)](#)
  - [DeregisterResource Action \(Python: deregister\\_resource\) \(p. 180\)](#)

- [ListResources Action \(Python: list\\_resources\) \(p. 180\)](#)
- [Tagging API \(p. 181\)](#)
  - [Data Types \(p. 181\)](#)
  - [LFTagKeyResource Structure \(p. 181\)](#)
  - [LFTagPolicyResource Structure \(p. 182\)](#)
  - [TaggedTable Structure \(p. 182\)](#)
  - [TaggedDatabase Structure \(p. 183\)](#)
  - [LFTag Structure \(p. 183\)](#)
  - [LFTagPair Structure \(p. 183\)](#)
  - [TagError Structure \(p. 183\)](#)
  - [ColumnLFTag Structure \(p. 184\)](#)
  - [Operations \(p. 184\)](#)
  - [AddLFTagsToResource Action \(Python: add\\_lf\\_tags\\_to\\_resource\) \(p. 184\)](#)
  - [RemoveLFTagsFromResource Action \(Python: remove\\_lf\\_tags\\_from\\_resource\) \(p. 185\)](#)
  - [GetResourceLFTags Action \(Python: get\\_resource\\_lf\\_tags\) \(p. 186\)](#)
  - [ListLFTags Action \(Python: list\\_lf\\_tags\) \(p. 186\)](#)
  - [CreateLFTag Action \(Python: create\\_lf\\_tag\) \(p. 187\)](#)
  - [GetLFTag Action \(Python: get\\_lf\\_tag\) \(p. 188\)](#)
  - [UpdateLFTag Action \(Python: update\\_lf\\_tag\) \(p. 189\)](#)
  - [DeleteLFTag Action \(Python: delete\\_lf\\_tag\) \(p. 189\)](#)
  - [SearchTablesByLFTags Action \(Python: search\\_tables\\_by\\_lf\\_tags\) \(p. 190\)](#)
  - [SearchDatabasesByLFTags Action \(Python: search\\_databases\\_by\\_lf\\_tags\) \(p. 191\)](#)
- [Common Data Types \(p. 192\)](#)
  - [ErrorDetail Structure \(p. 192\)](#)
  - [String Patterns \(p. 192\)](#)

## Permissions APIs

The Permissions API describes data types and operations having to do with granting and revoking permissions in AWS Lake Formation.

### Data Types

- [Resource Structure \(p. 166\)](#)
- [DatabaseResource Structure \(p. 166\)](#)
- [TableResource Structure \(p. 167\)](#)
- [TableWithColumnsResource Structure \(p. 167\)](#)
- [DataLocationResource Structure \(p. 168\)](#)
- [DataLakePrincipal Structure \(p. 168\)](#)
- [ResourcePermissions Structure \(p. 168\)](#)
- [ResourcePermissionsError Structure \(p. 168\)](#)
- [PrincipalResourcePermissions Structure \(p. 169\)](#)
- [DetailsMap Structure \(p. 169\)](#)
- [PrincipalResourcePermissionsError Structure \(p. 169\)](#)

- [ColumnWildcard Structure \(p. 170\)](#)
- [BatchPermissionsRequestEntry Structure \(p. 170\)](#)
- [BatchPermissionsFailureEntry Structure \(p. 170\)](#)
- [PrincipalPermissions Structure \(p. 170\)](#)

## Resource Structure

A structure for the resource.

### Fields

- **Catalog** – An empty-structure named `CatalogResource`.

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **Database** – A [DatabaseResource \(p. 166\)](#) object.

The database for the resource. Unique to the Data Catalog. A database is a set of associated table definitions organized into a logical group. You can Grant and Revoke database permissions to a principal.

- **Table** – A [TableResource \(p. 167\)](#) object.

The table for the resource. A table is a metadata definition that represents your data. You can Grant and Revoke table privileges to a principal.

- **TableWithColumns** – A [TableWithColumnsResource \(p. 167\)](#) object.

The table with columns for the resource. A principal with permissions to this resource can select metadata from the columns of a table in the Data Catalog and the underlying data in Amazon S3.

- **DataLocation** – A [DataLocationResource \(p. 168\)](#) object.

The location of an Amazon S3 path where permissions are granted or revoked.

- **LFTag** – A [LFTagKeyResource \(p. 181\)](#) object.

The tag key and values attached to a resource.

- **LFTagPolicy** – A [LFTagPolicyResource \(p. 182\)](#) object.

A list of tag conditions that define a resource's tag policy.

## DatabaseResource Structure

A structure for the database object.

### Fields

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, it is the account ID of the caller.

- **Name** – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The name of the database resource. Unique to the Data Catalog.

## TableResource Structure

A structure for the table object. A table is a metadata definition that represents your data. You can Grant and Revoke table privileges to a principal.

### Fields

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, it is the account ID of the caller.

- **DatabaseName** – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The name of the database for the table. Unique to a Data Catalog. A database is a set of associated table definitions organized into a logical group. You can Grant and Revoke database privileges to a principal.

- **Name** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The name of the table.

- **TableWildcard** – An empty-structure named TableWildcard.

A wildcard object representing every table under a database.

At least one of TableResource\$Name or TableResource\$TableWildcard is required.

## TableWithColumnsResource Structure

A structure for a table with columns object. This object is only used when granting a SELECT permission.

This object must take a value for at least one of ColumnsNames, ColumnsIndexes, or ColumnsWildcard.

### Fields

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, it is the account ID of the caller.

- **DatabaseName** – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The name of the database for the table with columns resource. Unique to the Data Catalog. A database is a set of associated table definitions organized into a logical group. You can Grant and Revoke database privileges to a principal.

- **Name** – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The name of the table resource. A table is a metadata definition that represents your data. You can Grant and Revoke table privileges to a principal.

- **ColumnNames** – An array of UTF-8 strings.

The list of column names for the table. At least one of ColumnNames or ColumnWildcard is required.

- **ColumnWildcard** – A [ColumnWildcard \(p. 170\)](#) object.

A wildcard specified by a `ColumnWildcard` object. At least one of `ColumnNames` or `ColumnWildcard` is required.

## DataLocationResource Structure

A structure for a data location object where permissions are granted or revoked.

### Fields

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog where the location is registered with AWS Lake Formation. By default, it is the account ID of the caller.

- `ResourceArn` – *Required:* UTF-8 string.

The Amazon Resource Name (ARN) that uniquely identifies the data location resource.

## DataLakePrincipal Structure

The AWS Lake Formation principal. Supported principals are IAM users or IAM roles.

### Fields

- `DataLakePrincipalIdentifier` – UTF-8 string, not less than 1 or more than 255 bytes long.

An identifier for the AWS Lake Formation principal.

## ResourcePermissions Structure

The permissions granted or revoked on a resource.

### Fields

- `Resource` – A [Resource \(p. 166\)](#) object.

The resource to be granted or revoked permissions.

- `Permissions` – An array of UTF-8 strings.

The permissions to be granted or revoked on the resource.

- `PermissionsWithGrantOption` – An array of UTF-8 strings.

Indicates whether to grant the ability to grant permissions (as a subset of permissions granted).

## ResourcePermissionsError Structure

A structure representing an error from granting or revoking permissions on the resource.

### Fields

- `ResourcePermissions` – A [ResourcePermissions \(p. 168\)](#) object.

A list of resource permissions that had errors.

- **Error** – An [ErrorDetail](#) (p. 192) object.

The error associated with the attempt to grant or revoke permissions on the resource.

## PrincipalResourcePermissions Structure

The permissions granted or revoked on a resource.

### Fields

- **Principal** – A [DataLakePrincipal](#) (p. 168) object.

The Data Lake principal to be granted or revoked permissions.

- **Resource** – A [Resource](#) (p. 166) object.

The resource where permissions are to be granted or revoked.

- **Permissions** – An array of UTF-8 strings.

The permissions to be granted or revoked on the resource.

- **PermissionsWithGrantOption** – An array of UTF-8 strings.

Indicates whether to grant the ability to grant permissions (as a subset of permissions granted).

- **AdditionalDetails** – A [DetailsMap](#) (p. 169) object.

This attribute can be used to return any additional details of `PrincipalResourcePermissions`. Currently returns only as a RAM resource share ARN.

## DetailsMap Structure

A structure containing the additional details to be returned in the `AdditionalDetails` attribute of `PrincipalResourcePermissions`.

If a catalog resource is shared through AWS Resource Access Manager (AWS RAM), then there will exist a corresponding RAM resource share ARN.

### Fields

- **ResourceShare** – An array of UTF-8 strings.

A resource share ARN for a catalog resource shared through AWS Resource Access Manager (AWS RAM).

## PrincipalResourcePermissionsError Structure

A structure representing an error in granting or revoking permissions to the principal.

### Fields

- **PrincipalResourcePermissions** – A [PrincipalResourcePermissions](#) (p. 169) object.

The principal permissions that were to be granted or revoked.

- **Error** – An [ErrorDetail](#) (p. 192) object.

The error message for the attempt to grant or revoke permissions.

## ColumnWildcard Structure

A wildcard object, consisting of an optional list of excluded column names or indexes.

### Fields

- **ExcludedColumnNames** – An array of UTF-8 strings.  
Excludes column names. Any column with this name will be excluded.

## BatchPermissionsRequestEntry Structure

A permission to a resource granted by batch operation to the principal.

### Fields

- **Id** – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long.  
A unique identifier for the batch permissions request entry.
- **Principal** – A [DataLakePrincipal \(p. 168\)](#) object.  
The principal to be granted a permission.
- **Resource** – A [Resource \(p. 166\)](#) object.  
The resource to which the principal is to be granted a permission.
- **Permissions** – An array of UTF-8 strings.  
The permissions to be granted.
- **PermissionsWithGrantOption** – An array of UTF-8 strings.  
Indicates if the option to pass permissions is granted.

## BatchPermissionsFailureEntry Structure

A list of failures when performing a batch grant or batch revoke operation.

### Fields

- **RequestEntry** – A [BatchPermissionsRequestEntry \(p. 170\)](#) object.  
An identifier for an entry of the batch request.
- **Error** – An [ErrorDetail \(p. 192\)](#) object.  
An error message that applies to the failure of the entry.

## PrincipalPermissions Structure

Permissions granted to a principal.

### Fields

- **Principal** – A [DataLakePrincipal \(p. 168\)](#) object.  
The principal who is granted permissions.



- **Permissions** – An array of UTF-8 strings.

The permissions that are granted to the principal.

## Operations

- [GrantPermissions Action \(Python: `grant\_permissions`\)](#) (p. 171)
- [RevokePermissions Action \(Python: `revoke\_permissions`\)](#) (p. 172)
- [BatchGrantPermissions Action \(Python: `batch\_grant\_permissions`\)](#) (p. 172)
- [BatchRevokePermissions Action \(Python: `batch\_revoke\_permissions`\)](#) (p. 173)
- [GetEffectivePermissionsForPath Action \(Python: `get\_effective\_permissions\_for\_path`\)](#) (p. 174)
- [ListPermissions Action \(Python: `list\_permissions`\)](#) (p. 174)

## GrantPermissions Action (Python: `grant_permissions`)

Grants permissions to the principal to access metadata in the Data Catalog and data organized in underlying data storage such as Amazon S3.

For information about permissions, see [Security and Access Control to Metadata and Data](#).

### Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern](#) (p. 192).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **Principal** – *Required:* A [DataLakePrincipal](#) (p. 168) object.

The principal to be granted the permissions on the resource. Supported principals are IAM users or IAM roles, and they are defined by their principal type and their ARN.

Note that if you define a resource with a particular ARN, then later delete, and recreate a resource with that same ARN, the resource maintains the permissions already granted.

- **Resource** – *Required:* A [Resource](#) (p. 166) object.

The resource to which permissions are to be granted. Resources in AWS Lake Formation are the Data Catalog, databases, and tables.

- **Permissions** – *Required:* An array of UTF-8 strings.

The permissions granted to the principal on the resource. AWS Lake Formation defines privileges to grant and revoke access to metadata in the Data Catalog and data organized in underlying data storage such as Amazon S3. AWS Lake Formation requires that each principal be authorized to perform a specific task on AWS Lake Formation resources.

- **PermissionsWithGrantOption** – An array of UTF-8 strings.

Indicates a list of the granted permissions that the principal may pass to other users. These permissions may only be a subset of the permissions granted in the `Privileges`.

### Response

- *No Response parameters.*

## Errors

- `ConcurrentModificationException`
- `EntityNotFoundException`
- `InvalidInputException`

## RevokePermissions Action (Python: `revoke_permissions`)

Revokes permissions to the principal to access metadata in the Data Catalog and data organized in underlying data storage such as Amazon S3.

### Request

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- `Principal` – *Required:* A [DataLakePrincipal \(p. 168\)](#) object.

The principal to be revoked permissions on the resource.

- `Resource` – *Required:* A [Resource \(p. 166\)](#) object.

The resource to which permissions are to be revoked.

- `Permissions` – *Required:* An array of UTF-8 strings.

The permissions revoked to the principal on the resource. For information about permissions, see [Security and Access Control to Metadata and Data](#).

- `PermissionsWithGrantOption` – An array of UTF-8 strings.

Indicates a list of permissions for which to revoke the grant option allowing the principal to pass permissions to other principals.

### Response

- *No Response parameters.*

## Errors

- `ConcurrentModificationException`
- `EntityNotFoundException`
- `InvalidInputException`

## BatchGrantPermissions Action (Python: `batch_grant_permissions`)

Batch operation to grant permissions to the principal.

### Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **Entries** – *Required:* An array of [BatchPermissionsRequestEntry \(p. 170\)](#) objects.

A list of up to 20 entries for resource permissions to be granted by batch operation to the principal.

### Response

- **Failures** – An array of [BatchPermissionsFailureEntry \(p. 170\)](#) objects.

A list of failures to grant permissions to the resources.

### Errors

- `InvalidInputException`
- `OperationTimeoutException`

## BatchRevokePermissions Action (Python: batch\_revoke\_permissions)

Batch operation to revoke permissions from the principal.

### Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **Entries** – *Required:* An array of [BatchPermissionsRequestEntry \(p. 170\)](#) objects.

A list of up to 20 entries for resource permissions to be revoked by batch operation to the principal.

### Response

- **Failures** – An array of [BatchPermissionsFailureEntry \(p. 170\)](#) objects.

A list of failures to revoke permissions to the resources.

### Errors

- `InvalidInputException`
- `OperationTimeoutException`

## GetEffectivePermissionsForPath Action (Python: get\_effective\_permissions\_for\_path)

Returns the Lake Formation permissions for a specified table or database resource located at a path in Amazon S3. `GetEffectivePermissionsForPath` will not return databases and tables if the catalog is encrypted.

### Request

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- `ResourceArn` – *Required:* UTF-8 string.

The Amazon Resource Name (ARN) of the resource for which you want to get permissions.

- `NextToken` – UTF-8 string.

A continuation token, if this is not the first call to retrieve this list.

- `MaxResults` – Number (integer), not less than 1 or more than 1000.

The maximum number of results to return.

### Response

- `Permissions` – An array of [PrincipalResourcePermissions \(p. 169\)](#) objects.

A list of the permissions for the specified table or database resource located at the path in Amazon S3.

- `NextToken` – UTF-8 string.

A continuation token, if this is not the first call to retrieve this list.

### Errors

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`

## ListPermissions Action (Python: list\_permissions)

Returns a list of the principal permissions on the resource, filtered by the permissions of the caller. For example, if you are granted an ALTER permission, you are able to see only the principal permissions for ALTER.

This operation returns only those permissions that have been explicitly granted.

For information about permissions, see [Security and Access Control to Metadata and Data](#).

## Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern](#) (p. 192).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **Principal** – A [DataLakePrincipal](#) (p. 168) object.

Specifies a principal to filter the permissions returned.

- **ResourceType** – UTF-8 string (valid values: CATALOG | DATABASE | TABLE | DATA\_LOCATION | LF\_TAG | LF\_TAG\_POLICY | LF\_TAG\_POLICY\_DATABASE | LF\_TAG\_POLICY\_TABLE).

Specifies a resource type to filter the permissions returned.

- **Resource** – A [Resource](#) (p. 166) object.

A resource where you will get a list of the principal permissions.

This operation does not support getting privileges on a table with columns. Instead, call this operation on the table, and the operation returns the table and the table w columns.

- **NextToken** – UTF-8 string.

A continuation token, if this is not the first call to retrieve this list.

- **MaxResults** – Number (integer), not less than 1 or more than 1000.

The maximum number of results to return.

## Response

- **PrincipalResourcePermissions** – An array of [PrincipalResourcePermissions](#) (p. 169) objects.

A list of principals and their permissions on the resource for the specified principal and resource types.

- **NextToken** – UTF-8 string.

A continuation token, if this is not the first call to retrieve this list.

## Errors

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

# Data Lake Settings APIs

The Data Lake Settings API describes data types and operations for managing the data lake administrators.

## Data Types

- [DataLakeSettings Structure](#) (p. 176)

## DataLakeSettings Structure

A structure representing a list of AWS Lake Formation principals designated as data lake administrators and lists of principal permission entries for default create database and default create table permissions.

### Fields

- **DataLakeAdmins** – An array of [DataLakePrincipal](#) (p. 168) objects, not more than 10 structures.

A list of AWS Lake Formation principals. Supported principals are IAM users or IAM roles.

- **CreateDatabaseDefaultPermissions** – An array of [PrincipalPermissions](#) (p. 170) objects.

Specifies whether access control on newly created database is managed by Lake Formation permissions or exclusively by IAM permissions. You can override this default setting when you create a database.

A null value indicates access control by Lake Formation permissions. A value that assigns ALL to IAM\_ALLOWED\_PRINCIPALS indicates access control by IAM permissions. This is referred to as the setting "Use only IAM access control," and is for backward compatibility with the AWS Glue permission model implemented by IAM permissions.

The only permitted values are an empty array or an array that contains a single JSON object that grants ALL to IAM\_ALLOWED\_PRINCIPALS.

```
[
    {
        "Principal": {
            "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
        },
        "Permissions": [
            "ALL"
        ]
    }
]
```

For more information, see [Changing the Default Security Settings for Your Data Lake](#).

- **CreateTableDefaultPermissions** – An array of [PrincipalPermissions](#) (p. 170) objects.

Specifies whether access control on newly created table is managed by Lake Formation permissions or exclusively by IAM permissions.

A null value indicates access control by Lake Formation permissions. A value that assigns ALL to IAM\_ALLOWED\_PRINCIPALS indicates access control by IAM permissions. This is referred to as the setting "Use only IAM access control," and is for backward compatibility with the AWS Glue permission model implemented by IAM permissions.

The only permitted values are an empty array or an array that contains a single JSON object that grants ALL to IAM\_ALLOWED\_PRINCIPALS.

```
[
    {
        "Principal": {
            "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
        },
        "Permissions": [
            "ALL"
        ]
    }
]
```

]

For more information, see [Changing the Default Security Settings for Your Data Lake](#).

- **TrustedResourceOwners** – An array of UTF-8 strings.

A list of the resource-owning account IDs that the caller's account can use to share their user access details (user ARNs). The user ARNs can be logged in the resource owner's AWS CloudTrail log.

You may want to specify this property when you are in a high-trust boundary, such as the same team or company.

## Operations

- [GetDataLakeSettings Action \(Python: `get\_data\_lake\_settings`\) \(p. 177\)](#)
- [PutDataLakeSettings Action \(Python: `put\_data\_lake\_settings`\) \(p. 177\)](#)

## GetDataLakeSettings Action (Python: `get_data_lake_settings`)

Retrieves the list of the data lake administrators of a Lake Formation-managed data lake.

### Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

### Response

- **DataLakeSettings** – A [DataLakeSettings \(p. 176\)](#) object.

A structure representing a list of AWS Lake Formation principals designated as data lake administrators.

### Errors

- `InternalServiceException`
- `InvalidInputException`
- `EntityNotFoundException`

## PutDataLakeSettings Action (Python: `put_data_lake_settings`)

Sets the list of data lake administrators who have admin privileges on all resources managed by Lake Formation. For more information on admin privileges, see [Granting Lake Formation Permissions](#).

This API replaces the current list of data lake admins with the new list being passed. To add an admin, fetch the current list and add the new admin to that list and pass that list in this API.

### Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **DataLakeSettings** – *Required:* A [DataLakeSettings \(p. 176\)](#) object.

A structure representing a list of AWS Lake Formation principals designated as data lake administrators.

### Response

- *No Response parameters.*

### Errors

- `InternalServiceException`
- `InvalidInputException`

## Credential Vending API

The Credential Vending API describes the data types and API related to working with the AWS Lake Formation service to vend credentials and to register and manage a data lake resource.

## Data Types

- [FilterCondition Structure \(p. 178\)](#)
- [ColumnNames list \(p. 179\)](#)
- [ResourceInfo Structure \(p. 179\)](#)

## FilterCondition Structure

This structure describes the filtering of columns in a table based on a filter condition.

### Fields

- **Field** – UTF-8 string (valid values: `RESOURCE_ARN` | `ROLE_ARN` | `LAST_MODIFIED`).

The field to filter in the filter condition.

- **ComparisonOperator** – UTF-8 string (valid values: `EQ` | `NE` | `LE` | `LT` | `GE` | `GT` | `CONTAINS` | `NOT_CONTAINS` | `BEGINS_WITH` | `IN` | `BETWEEN`).

The comparison operator used in the filter condition.

- **StringValueList** – An array of UTF-8 strings.

A string with values used in evaluating the filter condition.



## ColumnNames list

A list of column names in a table.

An array of UTF-8 strings.

A list of column names in a table.

## ResourceInfo Structure

A structure containing information about an AWS Lake Formation resource.

### Fields

- **ResourceArn** – UTF-8 string.  
The Amazon Resource Name (ARN) of the resource.
- **RoleArn** – UTF-8 string, matching the [Custom string pattern #5](#) (p. 192).  
The IAM role that registered a resource.
- **LastModified** – Timestamp.  
The date and time the resource was last modified.

## Operations

- [RegisterResource Action](#) (Python: `register_resource`) (p. 179)
- [DeregisterResource Action](#) (Python: `deregister_resource`) (p. 180)
- [ListResources Action](#) (Python: `list_resources`) (p. 180)

## RegisterResource Action (Python: `register_resource`)

Registers the resource as managed by the Data Catalog.

To add or update data, Lake Formation needs read/write access to the chosen Amazon S3 path. Choose a role that you know has permission to do this, or choose the `AWSServiceRoleForLakeFormationDataAccess` service-linked role. When you register the first Amazon S3 path, the service-linked role and a new inline policy are created on your behalf. Lake Formation adds the first path to the inline policy and attaches it to the service-linked role. When you register subsequent paths, Lake Formation adds the path to the existing policy.

The following request registers a new location and gives AWS Lake Formation permission to use the service-linked role to access that location.

```
ResourceArn = arn:aws:s3:::my-bucket UseServiceLinkedRole = true
```

If `UseServiceLinkedRole` is not set to true, you must provide or set the `RoleArn`:

```
arn:aws:iam::12345:role/my-data-access-role
```

### Request

- **ResourceArn** – *Required*: UTF-8 string.  
The Amazon Resource Name (ARN) of the resource that you want to register.

- `UseServiceLinkedRole` – Boolean.

Designates an AWS Identity and Access Management (IAM) service-linked role by registering this role with the Data Catalog. A service-linked role is a unique type of IAM role that is linked directly to Lake Formation.

For more information, see [Using Service-Linked Roles for Lake Formation](#).

- `RoleArn` – UTF-8 string, matching the [Custom string pattern #5](#) (p. 192).

The identifier for the role that registers the resource.

## Response

- *No Response parameters.*

## Errors

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `AlreadyExistsException`

# DeregisterResource Action (Python: deregister\_resource)

Deregisters the resource as managed by the Data Catalog.

When you deregister a path, Lake Formation removes the path from the inline policy attached to your service-linked role.

## Request

- `ResourceArn` – *Required:* UTF-8 string.

The Amazon Resource Name (ARN) of the resource that you want to deregister.

## Response

- *No Response parameters.*

## Errors

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

# ListResources Action (Python: list\_resources)

Lists the resources registered to be managed by the Data Catalog.

### Request

- `FilterConditionList` – An array of [FilterCondition \(p. 178\)](#) objects, not less than 1 or more than 20 structures.

Any applicable row-level and/or column-level filtering conditions for the resources.

- `MaxResults` – Number (integer), not less than 1 or more than 1000.

The maximum number of resource results.

- `NextToken` – UTF-8 string.

A continuation token, if this is not the first call to retrieve these resources.

### Response

- `ResourceInfoList` – An array of [ResourceInfo \(p. 179\)](#) objects.

A summary of the data lake resources.

- `NextToken` – UTF-8 string.

A continuation token, if this is not the first call to retrieve these resources.

### Errors

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

## Tagging API

The Tagging API describes the data types and API related to an authorization strategy that defines a permissions model on attributes or key-value pair tags.

### Data Types

- [LFTagKeyResource Structure \(p. 181\)](#)
- [LFTagPolicyResource Structure \(p. 182\)](#)
- [TaggedTable Structure \(p. 182\)](#)
- [TaggedDatabase Structure \(p. 183\)](#)
- [LFTag Structure \(p. 183\)](#)
- [LFTagPair Structure \(p. 183\)](#)
- [TagError Structure \(p. 183\)](#)
- [ColumnLFTag Structure \(p. 184\)](#)

### LFTagKeyResource Structure

A structure containing a tag key and values for a resource.

### Fields

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **TagKey** – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The key-name for the tag.

- **TagValues** – *Required:* An array of UTF-8 strings, not less than 1 or more than 50 strings.

A list of possible values an attribute can take.

## LFTagPolicyResource Structure

A structure containing a list of tag conditions that apply to a resource's tag policy.

### Fields

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **ResourceType** – *Required:* UTF-8 string (valid values: DATABASE | TABLE).

The resource type for which the tag policy applies.

- **Expression** – *Required:* An array of [LFTag \(p. 183\)](#) objects, not less than 1 or more than 5 structures.

A list of tag conditions that apply to the resource's tag policy.

## TaggedTable Structure

A structure describing a table resource with tags.

### Fields

- **Table** – A [TableResource \(p. 167\)](#) object.

A table that has tags attached to it.

- **LFTagOnDatabase** – An array of [LFTagPair \(p. 183\)](#) objects, not less than 1 or more than 50 structures.

A list of tags attached to the database where the table resides.

- **LFTagsOnTable** – An array of [LFTagPair \(p. 183\)](#) objects, not less than 1 or more than 50 structures.

A list of tags attached to the table.

- **LFTagsOnColumns** – An array of [ColumnLFTag \(p. 184\)](#) objects.

A list of tags attached to columns in the table.

## TaggedDatabase Structure

A structure describing a database resource with tags.

### Fields

- **Database** – A [DatabaseResource](#) (p. 166) object.  
A database that has tags attached to it.
- **LFTags** – An array of [LFTagPair](#) (p. 183) objects, not less than 1 or more than 50 structures.  
A list of tags attached to the database.

## LFTag Structure

A structure that allows an admin to grant user permissions on certain conditions. For example, granting a role access to all columns not tagged 'PII' of tables tagged 'Prod'.

### Fields

- **TagKey** – *Required:* UTF-8 string, not less than 1 or more than 128 bytes long.  
The key-name for the tag.
- **TagValues** – *Required:* An array of UTF-8 strings, not less than 1 or more than 50 strings.  
A list of possible values an attribute can take.

## LFTagPair Structure

A structure containing a tag key-value pair.

### Fields

- **CatalogId** – *Required:* Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern](#) (p. 192).  
The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.
- **TagKey** – *Required:* UTF-8 string, not less than 1 or more than 128 bytes long.  
The key-name for the tag.
- **TagValues** – *Required:* An array of UTF-8 strings, not less than 1 or more than 50 strings.  
A list of possible values an attribute can take.

## TagError Structure

A structure containing an error related to a `TagResource` or `UntagResource` operation.

### Fields

- **Tag** – A [LFTagPair](#) (p. 183) object.

The key-name of the tag.

- **Error** – An [ErrorDetail](#) (p. 192) object.

An error that occurred with the attachment or detachment of the tag.

## ColumnLFTag Structure

A structure containing the name of a column resource and the tags attached to it.

### Fields

- **Name** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern](#) (p. 192).

The name of a column resource.

- **LFTags** – An array of [LFTagPair](#) (p. 183) objects, not less than 1 or more than 50 structures.

The tags attached to a column resource.

## Operations

- [AddLFTagsToResource Action](#) (Python: `add_lf_tags_to_resource`) (p. 184)
- [RemoveLFTagsFromResource Action](#) (Python: `remove_lf_tags_from_resource`) (p. 185)
- [GetResourceLFTags Action](#) (Python: `get_resource_lf_tags`) (p. 186)
- [ListLFTags Action](#) (Python: `list_lf_tags`) (p. 186)
- [CreateLFTag Action](#) (Python: `create_lf_tag`) (p. 187)
- [GetLFTag Action](#) (Python: `get_lf_tag`) (p. 188)
- [UpdateLFTag Action](#) (Python: `update_lf_tag`) (p. 189)
- [DeleteLFTag Action](#) (Python: `delete_lf_tag`) (p. 189)
- [SearchTablesByLFTags Action](#) (Python: `search_tables_by_lf_tags`) (p. 190)
- [SearchDatabasesByLFTags Action](#) (Python: `search_databases_by_lf_tags`) (p. 191)

## AddLFTagsToResource Action (Python: `add_lf_tags_to_resource`)

Attaches one or more tags to an existing resource.

### Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern](#) (p. 192).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **Resource** – *Required:* A [Resource](#) (p. 166) object.

The resource to which to attach a tag.

- **LFTags** – *Required:* An array of [LFTagPair \(p. 183\)](#) objects, not less than 1 or more than 50 structures.

The tags to attach to the resource.

### Response

- **Failures** – An array of [TagError \(p. 183\)](#) objects.

A list of failures to tag the resource.

### Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `AccessDeniedException`
- `ConcurrentModificationException`

## RemoveLFTagsFromResource Action (Python: `remove_lf_tags_from_resource`)

Removes a tag from the resource. Only database, table, or tableWithColumns resource are allowed. To tag columns, use the column inclusion list in `tableWithColumns` to specify column input.

### Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **Resource** – *Required:* A [Resource \(p. 166\)](#) object.

The resource where you want to remove a tag.

- **LFTags** – *Required:* An array of [LFTagPair \(p. 183\)](#) objects, not less than 1 or more than 50 structures.

The tags to be removed from the resource.

### Response

- **Failures** – An array of [TagError \(p. 183\)](#) objects.

A list of failures to untag a resource.

### Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`

- `OperationTimeoutException`
- `GlueEncryptionException`
- `AccessDeniedException`
- `ConcurrentModificationException`

## GetResourceLFTags Action (Python: `get_resource_lf_tags`)

Returns the tags applied to a resource.

### Request

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern](#) (p. 192).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- `Resource` – *Required:* A [Resource](#) (p. 166) object.

The resource for which you want to return tags.

- `ShowAssignedLFTags` – Boolean.

Indicates whether to show the assigned tags.

### Response

- `LFTagOnDatabase` – An array of [LFTagPair](#) (p. 183) objects, not less than 1 or more than 50 structures.

A list of tags applied to a database resource.

- `LFTagsOnTable` – An array of [LFTagPair](#) (p. 183) objects, not less than 1 or more than 50 structures.

A list of tags applied to a table resource.

- `LFTagsOnColumns` – An array of [ColumnLFTag](#) (p. 184) objects.

A list of tags applied to a column resource.

### Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `AccessDeniedException`

## ListLFTags Action (Python: `list_lf_tags`)

Lists tags that the requester has permission to view.



## Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **ResourceShareType** – UTF-8 string (valid values: `FOREIGN` | `ALL`).

If resource share type is `ALL`, returns both in-account tags and shared tags that the requester has permission to view. If resource share type is `FOREIGN`, returns all share tags that the requester can view. If no resource share type is passed, lists tags in the given catalog ID that the requester has permission to view.

- **MaxResults** – Number (integer), not less than 1 or more than 1000.

The maximum number of results to return.

- **NextToken** – UTF-8 string.

A continuation token, if this is not the first call to retrieve this list.

## Response

- **LFTags** – An array of [LFTagPair \(p. 183\)](#) objects, not less than 1 or more than 50 structures.

A list of tags that the requested has permission to view.

- **NextToken** – UTF-8 string.

A continuation token, present if the current list segment is not the last.

## Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

# CreateLFTag Action (Python: create\_lf\_tag)

Creates a tag with the specified name and values.

## Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **TagKey** – *Required:* UTF-8 string, not less than 1 or more than 128 bytes long.

The key-name for the tag.

- **TagValues** – *Required:* An array of UTF-8 strings, not less than 1 or more than 50 strings.

A list of possible values an attribute can take.

### Response

- *No Response parameters.*

### Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `AccessDeniedException`

## GetLFTag Action (Python: `get_lf_tag`)

Returns a tag definition.

### Request

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- `TagKey` – *Required:* UTF-8 string, not less than 1 or more than 128 bytes long.

The key-name for the tag.

### Response

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- `TagKey` – UTF-8 string, not less than 1 or more than 128 bytes long.

The key-name for the tag.

- `TagValues` – An array of UTF-8 strings, not less than 1 or more than 50 strings.

A list of possible values an attribute can take.

### Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`

- `OperationTimeoutException`
- `AccessDeniedException`

## UpdateLFTag Action (Python: `update_lf_tag`)

Updates the list of possible values for the specified tag key. If the tag does not exist, the operation throws an `EntityNotFoundException`. The values in the delete key values will be deleted from list of possible values. If any value in the delete key values is attached to a resource, then API errors out with a 400 Exception - "Update not allowed". Untag the attribute before deleting the tag key's value.

### Request

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern](#) (p. 192).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- `TagKey` – *Required*: UTF-8 string, not less than 1 or more than 128 bytes long.

The key-name for the tag for which to add or delete values.

- `TagValuesToDelete` – An array of UTF-8 strings, not less than 1 or more than 50 strings.

A list of tag values to delete from the tag.

- `TagValuesToAdd` – An array of UTF-8 strings, not less than 1 or more than 50 strings.

A list of tag values to add from the tag.

### Response

- *No Response parameters.*

### Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `AccessDeniedException`

## DeleteLFTag Action (Python: `delete_lf_tag`)

Deletes the specified tag key name. If the attribute key does not exist or the tag does not exist, then the operation will not do anything. If the attribute key exists, then the operation checks if any resources are tagged with this attribute key, if yes, the API throws a 400 Exception with the message "Delete not allowed" as the tag key is still attached with resources. You can consider untagging resources with this tag key.

### Request

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **TagKey** – *Required:* UTF-8 string, not less than 1 or more than 128 bytes long.

The key-name for the tag to delete.

### Response

- *No Response parameters.*

### Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `AccessDeniedException`

## SearchTablesByLFTags Action (Python: `search_tables_by_lf_tags`)

This operation allows a search on `TABLE` resources by `LFTags`. This will be used by admins who want to grant user permissions on certain `LFTags`. Before making a grant, the admin can use `SearchTablesByLFTags` to find all resources where the given `LFTags` are valid to verify whether the returned resources can be shared.

### Request

- **NextToken** – UTF-8 string.

A continuation token, if this is not the first call to retrieve this list.

- **MaxResults** – Number (integer), not less than 1 or more than 1000.

The maximum number of results to return.

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 192\)](#).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- **Expression** – *Required:* An array of [LFTag \(p. 183\)](#) objects, not less than 1 or more than 5 structures.

A list of conditions (`LFTag` structures) to search for in table resources.

## Response

- `NextToken` – UTF-8 string.

A continuation token, present if the current list segment is not the last.

- `TableList` – An array of [TaggedTable](#) (p. 182) objects.

A list of tables that meet the tag conditions.

## Errors

- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`
- `GlueEncryptionException`

# SearchDatabasesByLFTags Action (Python: search\_databases\_by\_lf\_tags)

This operation allows a search on `DATABASE` resources by `TagCondition`. This operation is used by admins who want to grant user permissions on certain `TagConditions`. Before making a grant, the admin can use `SearchDatabasesByTags` to find all resources where the given `TagConditions` are valid to verify whether the returned resources can be shared.

## Request

- `NextToken` – UTF-8 string.

A continuation token, if this is not the first call to retrieve this list.

- `MaxResults` – Number (integer), not less than 1 or more than 1000.

The maximum number of results to return.

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern](#) (p. 192).

The identifier for the Data Catalog. By default, the account ID. The Data Catalog is the persistent metadata store. It contains database definitions, table definitions, and other control information to manage your AWS Lake Formation environment.

- `Expression` – *Required:* An array of [LFTag](#) (p. 183) objects, not less than 1 or more than 5 structures.

A list of conditions (`LFTag` structures) to search for in database resources.

## Response

- `NextToken` – UTF-8 string.

A continuation token, present if the current list segment is not the last.

- `DatabaseList` – An array of [TaggedDatabase](#) (p. 183) objects.

A list of databases that meet the tag conditions.

## Errors

- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`
- `GlueEncryptionException`

# 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](#) (p. 192).

The code associated with this error.

- **ErrorMessage** – Description string, not more than 2048 bytes long, matching the [URI address multi-line string pattern](#) (p. 192).

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-\uFFFF\uD800\uDC00-\uDBFF\uDFFF\t]*"`
- URI address multi-line string pattern – `"[\u0020-\uD7FF\uE000-\uFFFF\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/.*"`

# Lake Formation Personas and IAM Permissions Reference

This chapter lists some suggested AWS 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" \(p. 133\)](#).

## 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 <code>AdministratorAccess</code> AWS managed policy. Has all 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.
Data engineer	(Optional) User who can create databases, create and run crawlers and workflows, and grant Lake Formation permissions 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 <a href="#">Creating a Database (p. 43)</a> .
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.

## 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 \(p. 194\)](#)
- [Data Engineer Permissions \(p. 195\)](#)
- [Data Analyst Permissions \(p. 197\)](#)

- [Workflow Role Permissions \(p. 197\)](#)

## 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 \(p. 197\)](#).

Policy Type	Policy
AWS managed policies	<ul style="list-style-type: none"> <li>• AWSLakeFormationDataAdmin</li> <li>• AWSGlueConsoleFullAccess (Optional)</li> <li>• CloudWatchLogsReadOnlyAccess (Optional)</li> <li>• AWSLakeFormationCrossAccountManager (Optional)</li> <li>• AmazonAthenaFullAccess (Optional)</li> </ul> <p>For information about the optional AWS managed policies, see <a href="#">the section called "Create a Data Lake Administrator" (p. 8)</a>.</p>
Inline policy (for creating the Lake Formation service-linked role)	<pre>{   "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::&lt;account-id&gt;:role/aws-service-role/lakeformation.amazonaws.com/AWSServiceRoleForLakeFormationDataAccess"     }   ] }</pre>
Inline policy (for metadata access control using the tag-based access control (TBAC) method)	<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": [         "lakeformation:AddLFTagsToResource",         "lakeformation:RemoveLFTagsFromResource",         "lakeformation:GetResourceLFTags",         "lakeformation:ListLFTags",         "lakeformation:CreateLFTag",         "lakeformation:GetLFTag", </pre>



Policy Type	Policy
	<pre>         "lakeformation:UpdateLFTag",         "lakeformation:DeleteLFTag",         "lakeformation:SearchTablesByLFTags",         "lakeformation:SearchDatabasesByLFTags"       ],       "Resource": "*"     }   ] }</pre>
(Optional) Inline policy (passrole policy for the workflow role). This is required only if the data lake administrator creates and runs workflows.	<pre> {   "Version": "2012-10-17",   "Statement": [     {       "Sid": "PassRolePermissions",       "Effect": "Allow",       "Action": [         "iam:PassRole"       ],       "Resource": [         "arn:aws:iam::&lt;account-id&gt;:role/&lt;workflow_role&gt;"       ]     }   ] }</pre>
(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. <code>ram:EnableSharingWithAwsOrganization</code> is required only for data lake administrators in the AWS Organizations management account.	<pre> {   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": [         "ram:AcceptResourceShareInvitation",         "ram:RejectResourceShareInvitation",         "ec2:DescribeAvailabilityZones",         "ram:EnableSharingWithAwsOrganization"       ],       "Resource": "*"     }   ] }</pre>

## 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	<code>AWSGlueConsoleFullAccess</code>
Inline policy (basic)	<pre> {   "Version": "2012-10-17",   "Statement": [     {</pre>

Policy Type	Policy
	<pre> "Effect": "Allow", "Action": [     "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",     "iam:CreateRole",     "iam:CreatePolicy",     "iam:AttachRolePolicy" ], "Resource": "*"     ] } </pre>
Inline policy (for metadata access control using the tag-based access control (TBAC) method)	<pre> {   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": [         "lakeformation:AddLFTagsToResource",         "lakeformation:RemoveLFTagsFromResource",         "lakeformation:GetResourceLFTags",         "lakeformation:ListLFTags",         "lakeformation:GetLFTag",         "lakeformation:SearchTablesByLFTags",         "lakeformation:SearchDatabasesByLFTags"       ],       "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::&lt;account-id&gt;:role/&lt;workflow_role&gt;"       ]     }   ] } </pre>

## Data Analyst Permissions

Policy Type	Policy
AWS managed policy	AmazonAthenaFullAccess
Inline policy (basic)	<pre>{   "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": "*"     }   ] }</pre>

## 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)	<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Sid": "Lakeformation",       "Effect": "Allow",       "Action": [         "lakeformation:GetDataAccess",         "lakeformation:GrantPermissions"       ],       "Resource": "*"     }   ] }</pre>

Policy Type	Policy
	<pre>     ]   } </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::&lt;account-id&gt;:role/&lt;workflow_role&gt;"       ]     }   ] } </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:::&lt;your-s3-cloudtrail-bucket&gt;/*"]     }   ] } </pre>

# Troubleshooting Lake Formation

If you encounter issues when working with AWS Lake Formation, consult the topics in this section.

## Topics

- [General Troubleshooting](#) (p. 199)
- [Troubleshooting Cross-Account Access](#) (p. 199)
- [Troubleshooting Blueprints and Workflows](#) (p. 201)

## 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](#) (p. 69) and [Granting Data Location Permissions](#) (p. 128).

### 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.

## 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 \(p. 200\)](#)
- [Principals in the recipient account can see the Data Catalog resource but can't access the underlying data \(p. 200\)](#)
- [Error: "Not authorized to grant permissions for the resource" \(p. 200\)](#)
- [Error: "Access denied to retrieve AWS Organization information" \(p. 201\)](#)
- [Error: "Organization <organization-ID> not found" \(p. 201\)](#)
- [Error: "Insufficient Lake Formation permissions: Illegal combination" \(p. 201\)](#)

## 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" \(p. 46\)](#).

- 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" \(p. 89\)](#).

- Do you have the AWS Identity and Access Management (IAM) permissions required to grant cross-account access?

For more information, see [the section called "Cross-Account Access Prerequisites" \(p. 82\)](#).

- 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 \(p. 85\)](#).

## 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](#) (p. 83).

## 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.

# 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>"](#) (p. 201)
- [My workflow failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>"](#) (p. 202)
- [A crawler in my workflow failed with "Resource does not exist or requester is not authorized to access requested permissions"](#) (p. 202)
- [A crawler in my workflow failed with "An error occurred \(AccessDeniedException\) when calling the CreateTable operation..."](#) (p. 202)

## 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 [Lake Formation Personas and IAM Permissions Reference](#) (p. 193).

## 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 "Create an IAM Role for Workflows" \(p. 7\)](#).

## 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" \(p. 139\)](#).



# Known Issues for AWS Lake Formation

Review these known issues for AWS Lake Formation.

## Topics

- [Limitation on Filtering of Table Metadata \(p. 203\)](#)
- [Issue with Renaming an Excluded Column \(p. 204\)](#)
- [Issue with Deleting Columns in CSV Tables \(p. 204\)](#)
- [Table Partitions Must Be Added Under a Common Path \(p. 204\)](#)
- [Issue with Creating a Database during Workflow Creation \(p. 204\)](#)
- [Issue with Deleting and Then Re-Creating a User \(p. 204\)](#)

## 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. For more information, see [the section called “\(Optional\) Allow Data Filtering on Amazon EMR Clusters” \(p. 12\)](#).

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 IAM user that 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.

# Document History for AWS Lake Formation

The following table describes important changes to the documentation for AWS Lake Formation.

update-history-change	update-history-description	update-history-date
<a href="#">Support for tag-based access control (p. 205)</a>	Added information about a new more scalable way to manage access to Data Catalog resources and underlying data by using policy tags. For more information, see <a href="#">Tag-Based Access Control in Lake Formation</a> .	May 7, 2021
<a href="#">Support for VPC interface endpoints (p. 205)</a>	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 <a href="#">Using Lake Formation with VPC Endpoints</a> .	March 31, 2021
<a href="#">New opt-in requirement for data filtering on Amazon EMR. (p. 205)</a>	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 <a href="#">Allow Data Filtering on Amazon EMR</a> .	October 9, 2020
<a href="#">Support for granting full cross-account permissions on Data Catalog databases (p. 205)</a>	Added information about granting full Lake Formation permissions on Data Catalog databases across AWS accounts, including <code>CREATE_TABLE</code> . For more information, see <a href="#">Sharing Data Catalog Databases</a> .	October 1, 2020
<a href="#">Support for Amazon Athena users authenticating through SAML. (p. 205)</a>	Added information about support for Athena users who connect through the JDBC or ODBC driver and authenticate through SAML identity providers such as Okta and Microsoft Active Directory Federation Service (AD FS). For more information, see <a href="#">AWS</a>	September 30, 2020

	<a href="#">Service Integrations with Lake Formation.</a>	
<a href="#">Support for cross-account access with an encrypted Data Catalog (p. 205)</a>	Added information about granting cross-account permissions when the Data Catalog is encrypted. For more information, see <a href="#">Cross-Account Access Prerequisites</a> .	July 30, 2020
<a href="#">Support for cross-account access to the data lake (p. 205)</a>	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 information, see <a href="#">Cross-Account Access</a> .	July 7, 2020
<a href="#">Integration with Amazon QuickSight (p. 205)</a>	Added information about how to grant Lake Formation permissions to Amazon QuickSight Enterprise Edition users so that they may access datasets residing in registered Amazon S3 locations. For more information, see <a href="#">Granting Data Catalog Permissions</a> .	June 29, 2020
<a href="#">Updates to Setting Up and Getting Started chapters (p. 205)</a>	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
<a href="#">Support for (p. 205)</a>	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 information about how to register Amazon S3 locations that are encrypted with AWS KMS customer master keys. For more information, see <a href="#">Adding an Amazon S3 Location to Your Data Lake (p. 32)</a> .	February 27, 2020

<a href="#">Updates to blueprints and data lake administrator IAM policies (p. 205)</a>	Clarified input parameters for incremental database blueprints. Updated the IAM policies required for a data lake administrator.	December 20, 2019
<a href="#">Security chapter rewrite and upgrade chapter revisions (p. 205)</a>	Improved the security and upgrading chapters.	October 29, 2019
<a href="#">Super permission replaces All permission (p. 205)</a>	Updated the Security and Upgrading chapters to reflect the replacement of the permission All with Super.	October 10, 2019
<a href="#">Additions, corrections, and clarifications (p. 205)</a>	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
<a href="#">New guide (p. 205)</a>	This is the initial release of the <i>AWS Lake Formation Developer Guide</i> .	August 8, 2019

# AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.