
AWS Lake Formation

Developer Guide



AWS Lake Formation: Developer Guide

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

| | |
|--|----|
| What is AWS Lake Formation? | 1 |
| Lake Formation features | 1 |
| Setup and data management | 1 |
| Security management | 2 |
| AWS service integrations with Lake Formation | 3 |
| Supported Regions | 4 |
| Getting started with Lake Formation | 5 |
| How It Works | 6 |
| Lake Formation Terminology | 6 |
| Data Lake | 6 |
| Data Access | 7 |
| Blueprint | 7 |
| Workflow | 7 |
| Data Catalog | 7 |
| Underlying Data | 7 |
| Principal | 7 |
| Data Lake Administrator | 8 |
| Lake Formation Components | 8 |
| Lake Formation Console | 8 |
| Lake Formation API and Command Line Interface | 8 |
| Other AWS Services | 8 |
| Setting up AWS Lake Formation | 9 |
| Complete initial AWS configuration tasks | 10 |
| Sign up for AWS | 10 |
| Create an Administrator IAM User | 10 |
| Sign in as an IAM user | 11 |
| Create an IAM role for workflows | 11 |
| Create a data lake administrator | 12 |
| Change the default permission model | 15 |
| Create additional Lake Formation users | 16 |
| Configure an Amazon S3 location for your data lake | 17 |
| Prepare for using governed tables and row-level security | 17 |
| Prepare for using governed tables | 18 |
| Prepare for using automatic data compaction with governed tables | 20 |
| Prepare for using row-level security | 21 |
| (Optional) External data filtering settings | 22 |
| (Optional) Grant Access to the Data Catalog Encryption Key | 22 |
| Upgrading AWS Glue data permissions to the Lake Formation model | 23 |
| About upgrading to the Lake Formation permissions model | 23 |
| Step 1: List existing permissions | 24 |
| Using the API operation | 24 |
| Using the AWS Management Console | 25 |
| Using AWS CloudTrail | 25 |
| Step 2: Set up Lake Formation permissions | 25 |
| Step 3: Give users IAM permissions | 26 |
| Step 4: Switch to the Lake Formation permissions model | 26 |
| Verify Lake Formation permissions | 26 |
| Secure existing Data Catalog resources | 27 |
| Turn on Lake Formation permissions for your Amazon S3 location | 28 |
| Step 5: Secure new Data Catalog resources | 28 |
| Step 6: Give users a new IAM policy | 29 |
| Step 7: Clean up existing IAM policies | 29 |
| Getting started tutorials | 31 |
| Creating a data lake from an AWS CloudTrail source | 32 |

| | |
|--|----|
| Intended audience | 32 |
| Prerequisites | 33 |
| Step 1: Create the IAM user to be the data analyst | 33 |
| Step 2: Add permissions to read AWS CloudTrail logs to the workflow role | 34 |
| Step 3: Create an Amazon S3 bucket for the data lake | 34 |
| Step 4: Register an Amazon S3 path | 34 |
| Step 5: Grant data location permissions | 35 |
| Step 6: Create a database in the Data Catalog | 35 |
| Step 7: Grant data permissions | 35 |
| Step 8: Use a blueprint to create a workflow | 37 |
| Step 9: Run the workflow | 37 |
| Step 10: Grant SELECT on the tables | 38 |
| Step 11: Query the data lake Using Amazon Athena | 38 |
| Creating a data lake from a JDBC source | 39 |
| Intended audience | 39 |
| Prerequisites | 40 |
| Step 1: Create the IAM user to be the data analyst | 40 |
| Step 2: Create a connection in AWS Glue | 41 |
| Step 3: Create an Amazon S3 bucket for the data lake | 41 |
| Step 4: Register an Amazon S3 path | 42 |
| Step 5: Grant data location permissions | 42 |
| Step 6: Create a database in the Data Catalog | 42 |
| Step 7: Grant data permissions | 42 |
| Step 8: Use a blueprint to create a workflow | 43 |
| Step 9: Run the workflow | 44 |
| Step 10: Grant SELECT on the tables | 44 |
| Step 11: Query the data lake using Amazon Athena | 45 |
| Step 12: Query the data in the data lake using Amazon Redshift Spectrum | 45 |
| Step 13: Grant or revoke Lake Formation permissions using Amazon Redshift Spectrum | 48 |
| Creating a governed table in Lake Formation | 48 |
| Intended audience | 48 |
| Prerequisites | 49 |
| Step 1: Provision your resources | 49 |
| Step 2: Set up a governed table | 52 |
| Step 3: Configure Lake Formation permissions | 58 |
| Step 4: Add table objects into the governed table | 58 |
| Step 5: Querying the governed table using Amazon Athena | 61 |
| Step 6: Clean up AWS resources | 64 |
| Managing a data lake using tag-based access control | 65 |
| Intended audience | 66 |
| Prerequisites | 66 |
| Step 1: Provision your resources | 66 |
| Step 2: Register your data location, create an LF-tag ontology, and grant permissions | 67 |
| Step 3: Create Lake Formation databases | 69 |
| Step 4: Grant table permissions | 77 |
| Step 5: Run a query in Amazon Athena to verify the permissions | 78 |
| Step 6: Clean up AWS resources | 79 |
| Securing data lakes with row-level access control | 79 |
| Intended audience | 80 |
| Prerequisites | 80 |
| Step 1: Provision your resources | 80 |
| Step 2: Query without data filters | 81 |
| Step 3: Set up data filters and grant permissions | 85 |
| Step 4: Query with data filters | 86 |
| Step 5: Clean up AWS resources | 88 |
| Sharing Data Catalog resources with external AWS accounts using tag-based access control | 88 |
| Intended audience | 89 |

| | |
|---|-----|
| Configure Lake Formation settings | 90 |
| Step 1: Provision your resources using AWS CloudFormation templates | 92 |
| Step 2: Lake Formation cross-account sharing prerequisites | 93 |
| Step 3: Implement cross-account sharing using the tag-based access control method | 95 |
| Step 4: Implement the named resource method | 99 |
| Step 5: Clean up AWS resources | 101 |
| Sharing Data Catalog resources with external AWS accounts using fine-grained access control | 102 |
| Intended audience | 104 |
| Prerequisites | 104 |
| Step 1: Provide fine-grained access to another account | 105 |
| Step 2: Provide fine-grained access to a user in the same account | 106 |
| Adding an Amazon S3 location to your data lake | 107 |
| Requirements for roles used to register locations | 107 |
| Registering an Amazon S3 location | 109 |
| Registering an encrypted Amazon S3 location | 110 |
| Registering an Amazon S3 location in another AWS account | 113 |
| Registering an encrypted Amazon S3 location across AWS accounts | 115 |
| Deregistering an Amazon S3 location | 117 |
| Managing Data Catalog Tables and Databases | 119 |
| Creating a Database | 119 |
| Creating Tables | 120 |
| Managing governed tables | 120 |
| Governed tables in Lake Formation | 120 |
| Performance optimization of governed tables and row filters | 122 |
| Prerequisites for governed tables | 122 |
| Creating governed tables | 123 |
| Reading and writing governed Tables | 124 |
| Storage optimizations for governed tables | 125 |
| Notes and restrictions for governed tables | 128 |
| Searching for Tables | 129 |
| Sharing Data Catalog tables and databases across accounts | 130 |
| Accessing and viewing shared Data Catalog tables and databases | 130 |
| Accepting an AWS RAM resource share invitation | 154 |
| Viewing shared Data Catalog tables and databases | 156 |
| Creating resource links | 157 |
| How Resource Links Work | 157 |
| Creating a Resource Link to a Shared Table | 159 |
| Creating a Resource Link to a Shared Database | 161 |
| Resource link handling in AWS Glue APIs | 162 |
| Importing data using workflows | 166 |
| Blueprints and workflows | 166 |
| Creating a workflow | 167 |
| Running a workflow | 169 |
| Managing Permissions | 171 |
| Overview of Lake Formation permissions | 171 |
| IAM Permissions Required to Grant Lake Formation Permissions | 172 |
| Implicit Lake Formation permissions | 173 |
| Granting data location permissions | 174 |
| Granting data location permissions (same account) | 174 |
| Granting data location permissions (external account) | 177 |
| Granting permissions on a data location shared with your account | 179 |
| Granting and revoking Data Catalog permissions | 180 |
| Granting Data Catalog permissions using the named resource method | 181 |
| Granting Data Catalog permissions using the LF-TBAC method | 192 |
| Viewing Database and Table Permissions | 196 |
| Cross-account data sharing | 199 |
| Cross-account data sharing prerequisites | 200 |

| | |
|--|-----|
| Updating cross-account version settings | 203 |
| Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts | 206 |
| Granting permissions on a database or table shared with your account | 207 |
| Granting resource link permissions | 209 |
| Cross-account best practices and limitations | 210 |
| Accessing the underlying data of a shared table | 212 |
| Cross-account CloudTrail logging | 213 |
| Managing cross-account permissions using both AWS Glue and Lake Formation | 216 |
| Viewing all cross-account grants using the GetResourceShares API operation | 218 |
| Revoking Permissions Using the Console | 219 |
| Lake Formation Permissions Reference | 219 |
| Lake Formation Grant and Revoke AWS CLI Commands | 220 |
| ALTER | 222 |
| CREATE_DATABASE | 223 |
| CREATE_TABLE | 224 |
| DATA_LOCATION_ACCESS | 225 |
| DELETE | 225 |
| DESCRIBE | 226 |
| DROP | 227 |
| INSERT | 227 |
| SELECT | 228 |
| Super | 229 |
| Tag-based access control | 231 |
| Overview of Lake Formation tag-based access control | 231 |
| What is Lake Formation tag-based access control? | 231 |
| Comparison of Lake Formation tag-based access control to IAM attribute-based access control .. | 231 |
| How Lake Formation tag-based access control works | 232 |
| Lake Formation tag-based access control permissions model | 237 |
| Lake Formation Tag-based access control notes and restrictions | 239 |
| Managing LF-Tags for metadata access control | 240 |
| Creating LF-Tags | 241 |
| Updating LF-Tags | 242 |
| Deleting LF-Tags | 242 |
| Listing LF-Tags | 243 |
| Assigning LF-Tags to Data Catalog resources | 246 |
| Viewing LF-Tags assigned to a resource | 250 |
| Viewing the resources that a LF-Tag is assigned to | 252 |
| Granting, revoking, and listing LF-Tag permissions | 254 |
| Listing LF-Tag permissions using the console | 254 |
| Granting LF-Tag permissions using the console | 255 |
| Granting, Revoking, and Listing LF-Tag Permissions Using the AWS CLI | 257 |
| Data filtering and cell-level security | 260 |
| Overview of data filtering | 260 |
| Data filters | 261 |
| PartiQL support in row filter expressions | 263 |
| Supported data types | 264 |
| Row filter expressions | 264 |
| Reserved keywords | 264 |
| PartiQL reference | 265 |
| Notes and restrictions for column-level filtering | 265 |
| Notes and restrictions for row-Level and cell-level filtering | 265 |
| Permissions required for querying tables with cell-level filtering | 266 |
| Managing data filters | 267 |
| Creating a data filter | 267 |
| Granting data filter permissions | 269 |
| Granting data permissions provided by data filters | 271 |

| | |
|---|-----|
| Viewing data filters | 273 |
| Listing data filter permissions | 274 |
| Accessing the data lake within transactions | 276 |
| Transactional data operations | 277 |
| Commit process in governed table | 277 |
| Rolling back Amazon S3 writes | 277 |
| Metadata transactions | 278 |
| Limitations | 279 |
| API operations that support transactions | 280 |
| Coding best practices for transactions | 280 |
| Data lake transactions code samples | 281 |
| Security | 285 |
| Data Protection | 285 |
| Encryption at Rest | 286 |
| Infrastructure Security | 286 |
| VPC endpoints (AWS PrivateLink) | 287 |
| Cross-service confused deputy prevention | 288 |
| Security and access control | 289 |
| Lake Formation access control overview | 290 |
| AWS managed policies for Lake Formation | 298 |
| Changing the default security settings for your data lake | 299 |
| Permissions example scenario | 301 |
| Security Event Logging in AWS Lake Formation | 302 |
| Using Service-Linked Roles | 302 |
| Service-Linked Role Permissions for Lake Formation | 302 |
| Integrating with Lake Formation | 304 |
| Using Lake Formation credential vending | 304 |
| How Lake Formation credential vending works | 304 |
| Roles and responsibilities in Lake Formation credential vending | 306 |
| Lake Formation workflow for credential vending API operations | 306 |
| Registering a third-party query engine | 307 |
| Enabling permissions for a third-party query engine to call credential vending API operations ... | 308 |
| Working with other AWS services | 311 |
| Amazon Athena | 311 |
| Support for transactional table formats | 312 |
| Additional resources | 313 |
| Amazon Redshift Spectrum | 313 |
| Support for transactional table types | 314 |
| Additional resources | 315 |
| AWS Glue | 315 |
| Support for transactional table types | 316 |
| Additional resources | 317 |
| Amazon EMR | 317 |
| Support for transactional table formats | 318 |
| Additional resources | 318 |
| Amazon QuickSight | 318 |
| Additional resources | 319 |
| Logging AWS Lake Formation API Calls Using AWS CloudTrail | 320 |
| Lake Formation Information in CloudTrail | 320 |
| Understanding Lake Formation Events | 321 |
| Lake Formation API | 323 |
| Permissions | 325 |
| — data types — | 325 |
| Resource | 326 |
| DatabaseResource | 326 |
| TableResource | 327 |
| TableWithColumnsResource | 327 |

| | |
|---|-----|
| DataCellsFilterResource | 328 |
| DataLocationResource | 328 |
| DataLakePrincipal | 328 |
| ResourcePermissions | 329 |
| ResourcePermissionsError | 329 |
| PrincipalResourcePermissions | 329 |
| DetailsMap | 330 |
| PrincipalResourcePermissionsError | 330 |
| ColumnWildcard | 330 |
| BatchPermissionsRequestEntry | 330 |
| BatchPermissionsFailureEntry | 331 |
| PrincipalPermissions | 331 |
| — operations — | 331 |
| GrantPermissions (grant_permissions) | 331 |
| RevokePermissions (revoke_permissions) | 332 |
| BatchGrantPermissions (batch_grant_permissions) | 333 |
| BatchRevokePermissions (batch_revoke_permissions) | 334 |
| GetEffectivePermissionsForPath (get_effective_permissions_for_path) | 334 |
| ListPermissions (list_permissions) | 335 |
| Data Lake Settings | 336 |
| — data types — | 336 |
| DataLakeSettings | 336 |
| — operations — | 337 |
| GetDataLakeSettings (get_data_lake_settings) | 337 |
| PutDataLakeSettings (put_data_lake_settings) | 338 |
| Credential Vending | 338 |
| — data types — | 338 |
| FilterCondition | 338 |
| ColumnNames | 339 |
| ResourceInfo | 339 |
| — operations — | 339 |
| RegisterResource (register_resource) | 339 |
| DeregisterResource (deregister_resource) | 340 |
| ListResources (list_resources) | 341 |
| Tagging | 341 |
| — data types — | 341 |
| Tag | 342 |
| LFTagKeyResource | 342 |
| LFTagPolicyResource | 342 |
| TaggedTable | 343 |
| TaggedDatabase | 343 |
| LFTag | 343 |
| LFTagPair | 344 |
| LFTagError | 344 |
| ColumnLFTag | 344 |
| — operations — | 344 |
| AddLFTagsToResource (add_lf_tags_to_resource) | 345 |
| RemoveLFTagsFromResource (remove_lf_tags_from_resource) | 345 |
| GetResourceLFTags (get_resource_lf_tags) | 346 |
| ListLFTags (list_lf_tags) | 347 |
| CreateLFTag (create_lf_tag) | 348 |
| GetLFTag (get_lf_tag) | 348 |
| UpdateLFTag (update_lf_tag) | 349 |
| DeleteLFTag (delete_lf_tag) | 350 |
| SearchTablesByLFTags (search_tables_by_lf_tags) | 351 |
| SearchDatabasesByLFTags (search_databases_by_lf_tags) | 352 |
| Transaction APIs | 352 |

| | |
|--|-----|
| — data types — | 353 |
| TransactionDescription | 353 |
| VirtualObject | 353 |
| — operations — | 353 |
| StartTransaction (start_transaction) | 354 |
| CommitTransaction (commit_transaction) | 354 |
| CancelTransaction (cancel_transaction) | 355 |
| ExtendTransaction (extend_transaction) | 355 |
| DescribeTransaction (describe_transaction) | 356 |
| ListTransactions (list_transactions) | 356 |
| DeleteObjectsOnCancel (delete_objects_on_cancel) | 357 |
| — exceptions — | 358 |
| TransactionCommitInProgressException | 358 |
| TransactionAbortedException | 358 |
| TransactionCommittedException | 359 |
| TransactionCanceledException | 359 |
| TransactionContentionException | 359 |
| ResourceNotReadyException | 359 |
| Object APIs | 359 |
| — data types — | 360 |
| TableObject | 360 |
| PartitionObjects | 360 |
| AddObjectInput | 360 |
| DeleteObjectInput | 361 |
| WriteOperation | 361 |
| — operations — | 361 |
| GetTableObjects (get_table_objects) | 362 |
| UpdateTableObjects (update_table_objects) | 363 |
| Data Filter APIs | 364 |
| — data types — | 364 |
| DataCellsFilter | 364 |
| RowFilter | 364 |
| — operations — | 365 |
| CreateDataCellsFilter (create_data_cells_filter) | 365 |
| DeleteDataCellsFilter (delete_data_cells_filter) | 365 |
| ListDataCellsFilter (list_data_cells_filter) | 366 |
| Storage APIs | 367 |
| — data types — | 367 |
| StorageOptimizer | 367 |
| — operations — | 368 |
| ListTableStorageOptimizers (list_table_storage_optimizers) | 368 |
| UpdateTableStorageOptimizer (update_table_storage_optimizer) | 369 |
| Common Data Types | 370 |
| ErrorDetail | 370 |
| String Patterns | 370 |
| Lake Formation Personas and IAM Permissions Reference | 371 |
| AWS Lake Formation Personas | 371 |
| Personas Suggested Permissions | 371 |
| Data Lake Administrator Permissions | 372 |
| Data Engineer Permissions | 373 |
| Data Analyst Permissions | 375 |
| Workflow Role Permissions | 375 |
| Troubleshooting Lake Formation | 377 |
| General troubleshooting | 377 |
| Error: Insufficient Lake Formation permissions on <Amazon S3 location> | 377 |
| Error: "Insufficient encryption key permissions for Glue API" | 377 |
| My Amazon Athena or Amazon Redshift query that uses manifests is failing | 377 |

| | |
|---|-----|
| Error: "Insufficient Lake Formation permission(s): Required create tag on catalog" | 377 |
| Troubleshooting cross-account access | 378 |
| I granted a cross-account Lake Formation permission but the recipient can't see the resource | 378 |
| Principals in the recipient account can see the Data Catalog resource but can't access the underlying data | 378 |
| Error: "Association failed because the caller was not authorized" when accepting a AWS RAM resource share invitation | 379 |
| Error: "Not authorized to grant permissions for the resource" | 379 |
| Error: "Access denied to retrieve AWS Organization information" | 379 |
| Error: "Organization <organization-ID> not found" | 379 |
| Error: "Insufficient Lake Formation permissions: Illegal combination" | 380 |
| ConcurrentModificationException on grant/revoke requests to external accounts | 380 |
| Error when using Amazon EMR to access data shared via cross-account | 380 |
| Troubleshooting blueprints and workflows | 381 |
| My blueprint failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>" | 381 |
| My workflow failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>" | 381 |
| A crawler in my workflow failed with "Resource does not exist or requester is not authorized to access requested permissions" | 381 |
| A crawler in my workflow failed with "An error occurred (AccessDeniedException) when calling the CreateTable operation..." | 382 |
| Known issues for AWS Lake Formation | 383 |
| Limitation on filtering of table metadata | 383 |
| Issue with renaming an excluded column | 384 |
| Issue with deleting columns in CSV tables | 384 |
| Table partitions must be added under a common path | 384 |
| Issue with creating a database during workflow creation | 384 |
| Issue with deleting and then re-creating a user | 384 |
| GetTables and SearchTables APIs do not update the value for the IsRegisteredWithLakeFormation parameter | 385 |
| Data Catalog API operations do not update the value for the IsRegisteredWithLakeFormation parameter | 385 |
| Lake Formation operations do not support AWS Glue Schema Registry | 385 |
| Document History | 386 |
| AWS glossary | 391 |

What is AWS Lake Formation?

Welcome to the AWS Lake Formation Developer Guide.

AWS Lake Formation is a fully managed service that makes it easy 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.

Lake Formation provides its own permissions model that augments the IAM permissions model. This centrally defined permissions model enables fine-grained access to data stored in data lakes through a simple grant or revoke mechanism, much like a relational database management system (RDBMS). Lake Formation permissions are enforced using granular controls at the column, row, and cell-levels across AWS analytics and machine learning services, including Amazon Athena, Amazon QuickSight, and Amazon Redshift.

Topics

- [Lake Formation features \(p. 1\)](#)
- [AWS service integrations with Lake Formation \(p. 3\)](#)
- [Supported Regions \(p. 4\)](#)
- [Getting started with Lake Formation \(p. 5\)](#)

Lake Formation features

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

Topics

- [Setup and data management \(p. 1\)](#)
- [Security management \(p. 2\)](#)

Setup and data management

Import data from databases already in AWS

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

Import data from other external sources

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

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

Catalog and label your data

Lake Formation crawls and reads your data sources to extract technical metadata and creates a searchable catalog to describe this information for users so they can discover available datasets. You can also add your own custom labels to your data (at the table and column level) to define attributes, such as "sensitive information" and "European sales data." Lake Formation provides a text-based search over this metadata so your users can quickly find the data they need to analyze. For more information about adding tables to the Data Catalog, see [Managing Data Catalog Tables and Databases \(p. 119\)](#).

Transform data

Lake Formation can perform transformations on your data, such as rewriting various date formats for consistency to ensure that the data is stored in an analytics-friendly fashion. Lake Formation creates transformation templates and schedules jobs to prepare your data for analysis. Your data is transformed with AWS Glue and written in columnar formats, such as Parquet and ORC, for better performance.

Clean and deduplicate data

Lake Formation helps clean and prepare your data for analysis by providing a machine learning transform called FindMatches for deduplication and finding matching records. For example, use FindMatches to find duplicate records in your database of restaurants, such as when one record lists "Joe's Pizza" at "121 Main St." and another shows "Joseph's Pizzeria" at "121 Main." FindMatches will simply ask you to label sets of records as either "matching" or "not matching." The system will then learn your criteria for calling a pair of records a match and will build an machine learning transform that you can use to find duplicate records within a database or matching records across two databases. For more information about FindMatches, see [Matching Records with AWS Lake Formation FindMatches](#) in the [AWS Glue Developer Guide](#).

Storage optimizations

Analytics performance can be impacted by inefficient storage of many small files that are automatically created as new data is written to the data lake. Processing these many small files creates additional overhead for analytics services and causes slower query responses. Lake Formation includes a storage optimizer that automatically combines small files into larger files to speed up queries by up to 7x. This process, commonly known as compaction, is performed in the background so that there is no performance impact on your production workloads while this is taking place. For more information about the storage optimization features of Lake Formation, see [Storage optimizations for governed tables \(p. 125\)](#).

Row and cell-level security

Lake Formation provides data filters that allow you to restrict access to a combination of columns and rows. Use row and cell-level security to protect sensitive data like Personal Identifiable Information (PII). For more information about row-level security, see [Overview of data filtering \(p. 260\)](#).

Security management

Define and manage access controls

Lake Formation provides a single place to manage access controls for data in your data lake. You can define security policies that restrict access to data at the database, table, column, row, and cell levels. These policies apply to IAM users and roles, and to users and groups when federating through an external identity provider. You can use fine-grained controls to access data secured by Lake Formation within Amazon Redshift Spectrum, Athena, AWS Glue ETL, and Amazon EMR for Apache Spark.

Implement audit logging

Lake Formation provides comprehensive audit logs with CloudTrail to monitor access and show compliance with centrally defined policies. You can audit data access history across analytics and machine learning services that read the data in your data lake via Lake Formation. This lets you see which users or roles have attempted to access what data, with which services, and when. You can access audit logs in the same way you access any other CloudTrail logs using the CloudTrail APIs and console. For more information about CloudTrail logs see [Logging AWS Lake Formation API Calls Using AWS CloudTrail \(p. 320\)](#).

Tag-based access control

You can classify your data and limit access to sensitive information. You can also add your own custom labels (LF-tags) to the data at the table- and column-level to define attributes, like "sensitive information" or "European sales data." Lake Formation provides a text-based search over this metadata, so your users can quickly find the data they need to analyze. You can grant access to the data based on these LF-tags. For more information about tag-based access control, see [Lake Formation Tag-based access control \(p. 231\)](#).

Cross account access

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

Governed tables

Data lakes need to show users the correct view of data at all times, even while there are simultaneous real-time or frequent updates to the data. Loading streaming data or incorporating changes from multiple source data systems requires processing inserts and deletes across multiple tables in parallel. Today, developers write custom application code or use open source tools to manage these updates. These solutions are complex and difficult to scale because writing application code that maintains consistency when concurrently reading and writing the same data is tedious, brittle, and error prone.

Lake Formation introduces new APIs that support atomic, consistent, isolated, and durable (ACID) transactions using a new data lake table type, called a *governed table*. A governed table allows multiple users to concurrently insert and delete data across tables using manifests, while still allowing other users to simultaneously run analytical queries and ML models on the same data sets that return consistent and up-to-date results.

For more information about using transactions with Lake Formation, see the following topics:

- [Governed tables in Lake Formation \(p. 120\)](#)
- [Reading from and writing to the data lake within transactions \(p. 276\)](#)

AWS service integrations with Lake Formation

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

| AWS Service | How Integrated |
|-------------|--|
| AWS Glue | AWS Glue and Lake Formation share the same Data Catalog. For console operations (such as viewing a list of tables) and all API operations, AWS Glue users can access only the databases and tables on which they have Lake Formation permissions. Note AWS Glue does not support Lake Formation column permissions. |

| AWS Service | How Integrated |
|--------------------------------------|--|
| Amazon Athena | <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 Using Lake Formation and the Athena JDBC and ODBC Drivers for Federated Access to Athena in the <i>Amazon Athena User Guide</i>.</p> <p>Note Currently, authorizing access to SAML identities in Lake Formation is not supported in the following regions:</p> <ul style="list-style-type: none"> • Middle East (Bahrain) - me-south-1 • Asia Pacific (Hong Kong) - ap-east-1 • Africa (Cape Town) - af-south-1 • China (Ningxia) - cn-northwest-1 • Asia Pacific (Osaka) - ap-northeast-3 |
| Amazon Redshift Spectrum | <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> |
| Amazon QuickSight Enterprise Edition | 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 SELECT permission on the data. |
| Amazon EMR | 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.

Supported Regions

For the AWS Regions supported by AWS Lake Formation, see [AWS Lake Formation pricing](#).

For a list of the Lake Formation service endpoints for each Region and the Lake Formation service quotas, see [AWS Lake Formation endpoints and quotas](#).

The governed tables, transaction support, cell-level security, and storage optimizations features for Lake Formation are available in the following AWS Regions.

| Region name | Region parameter |
|-----------------------|------------------|
| US East (N. Virginia) | us-east-1 |

| Region name | Region parameter |
|---------------------------|------------------|
| US East (Ohio) | us-east-2 |
| US West (Oregon) | us-west-2 |
| Asia Pacific (Mumbai) | ap-south-1 |
| Asia Pacific (Seoul) | ap-northeast-2 |
| Asia Pacific (Singapore) | ap-southeast-1 |
| Asia Pacific (Sydney) | ap-southeast-2 |
| Asia Pacific (Tokyo) | ap-northeast-1 |
| Europe (Frankfurt) | eu-central-1 |
| Europe (Ireland) | eu-west-1 |
| Europe (London) | eu-west-2 |
| Europe (Stockholm) | eu-north-1 |
| Canada (Central) | ca-central-1 |
| South America (São Paulo) | sa-east-1 |

Getting started with Lake Formation

We recommend that you start with the following sections:

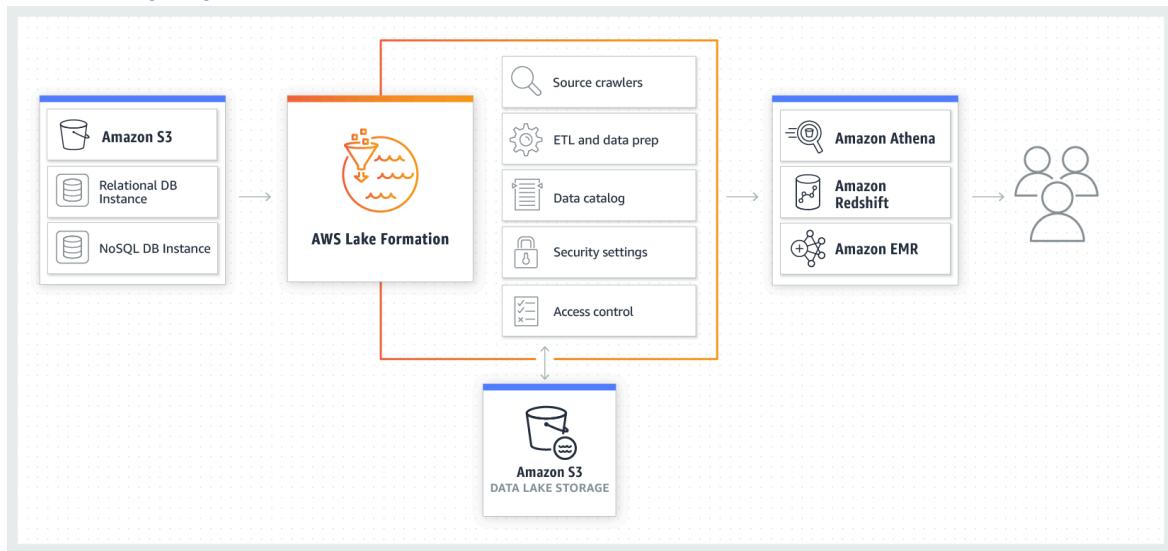
- [AWS Lake Formation: How It Works \(p. 6\)](#) — Learn about essential terminology and how the various components interact.
- [Setting up AWS Lake Formation \(p. 9\)](#) — Get information about prerequisites, and complete important setup tasks.
- [Tutorials \(p. 31\)](#) — Follow step-by-step tutorials to learn how to use Lake Formation.
- [Security in AWS Lake Formation \(p. 285\)](#) — 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 \(p. 6\)](#)
- [Lake Formation Components \(p. 8\)](#)

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 \(p. 173\)](#). For information about designating a user as a data lake administrator, see [Create a data lake administrator \(p. 12\)](#).

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. Complete initial AWS configuration tasks ([p. 10](#))
2. Create an IAM role for workflows ([p. 11](#))
3. Create a data lake administrator ([p. 12](#))
4. Change the default permission model ([p. 15](#))
5. the section called “Create additional Lake Formation users” ([p. 16](#))
6. the section called “Configure an Amazon S3 location for your data lake” ([p. 17](#))
7. the section called “Prepare for using governed tables” ([p. 18](#))
8. the section called “(Optional) External data filtering settings” ([p. 22](#))
9. the section called “(Optional) Grant Access to the Data Catalog Encryption Key” ([p. 22](#))

Creating your resources using AWS CloudFormation template

You may also use the AWS CloudFormation template to perform the initial Lake Formation set up in your account.

Note

The AWS CloudFormation stack performs steps 2 to 7 of the above, except step 4. Perform [Change the default permission model \(p. 15\)](#) manually from the Lake Formation console.

1. Sign in to the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation> as an IAM user in the US East (N. Virginia) Region.
2. Choose **Launch Stack**.
3. Choose **Next** on the **Create stack** screen.
4. Enter a **Stack name**.
5. For **DatalakeAdminName** and **DatalakeAdminPassword**, enter your IAM user name and password for data lake admin user.
6. For **DatalakeUserName** and **DatalakeUser1Password**, enter your IAM user name and password for data lake analyst user.
7. For **DataLakeBucketName**, enter your new bucket name that will be created.
8. Choose **Next**.
9. On the next page, choose **Next**.
10. Review the details on the final page and select **I acknowledge that AWS CloudFormation might create IAM resources**.
11. Choose **Create**.

The stack creation can take up to two minutes.

Clean up resources

If you like to clean up the AWS CloudFormation stack resources:

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

Complete initial AWS configuration tasks

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

Topics

- [Sign up for AWS \(p. 10\)](#)
- [Create an Administrator IAM User \(p. 10\)](#)
- [Sign in as an IAM user \(p. 11\)](#)

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.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, [assign administrative access to an administrative user](#), and use only the root user to perform [tasks that require root user access](#).

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, choose one of the following options.

| Choose one way to manage your administrator | To | By | You can also |
|---|--|--|---|
| In IAM Identity Center (Recommended) | Use short-term credentials to access AWS. This aligns with the security best practices. For information about best practices, see Security best practices in IAM in the <i>IAM User Guide</i> . | Following the instructions in Getting started in the <i>AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide</i> . | Configure programmatic access by Configuring the AWS CLI to use AWS IAM Identity Center (successor to AWS Single Sign-On) in the <i>AWS Command Line Interface User Guide</i> . |
| In IAM (Not recommended) | Use long-term credentials to access AWS. | Following the instructions in Creating your first IAM admin user and user group in the <i>IAM User Guide</i> . | Configure programmatic access by Managing access keys for IAM users in the <i>IAM User Guide</i> . |

Sign in as an IAM user

Sign in to the [IAM console](#) by choosing **IAM user** and entering your AWS account ID or account alias. On the next page, enter your IAM user name and your password.

Note

For your convenience, the AWS sign-in page uses a browser cookie to remember your IAM user name and account information. If you previously signed in as a different user, choose the sign-in link beneath the button to return to the main sign-in page. From there, you can enter your AWS account ID or account alias to be redirected to the IAM user sign-in page for your account.

Create an IAM role for workflows

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

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

The following procedure assumes familiarity with IAM.

To create an IAM role for workflows

1. Open the IAM console at <https://console.aws.amazon.com/iam> and sign in as the IAM administrator user that you created in [Create an Administrator IAM User \(p. 10\)](#) 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**.

4. On the **Add permissions** page, 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 (instances of workflows), and to attach the role to the created crawlers and jobs.
7. Verify that the role `LakeFormationWorkflowRole` has two policies attached.
 8. If you are ingesting data that is outside the data lake location, add an inline policy granting permissions to read the source data.

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 \(p. 173\)](#). By default, Lake Formation allows you to create up to 30 data lake administrators.

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

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

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

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

To create a data lake administrator (console)

1. If the 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 |
|--|------------|---|
| <code>AWSLakeFormationDataAdmin</code> | Mandatory | Basic data lake administrator permissions. |
| <code>AWSGlueConsoleFullAccess</code> , <code>CloudWatchLogsReadOnlyAccess</code> | 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 Importing data using workflows (p. 166) . |
| <code>AWSLakeFormationCrossAccountManager</code> | Optional | Attach this policy to enable the data lake administrator to grant and revoke cross-account permissions on Data Catalog resources. For more information, see Cross-account data sharing in Lake Formation (p. 199) . |
| <code>AmazonAthenaFullAccess</code> | 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” \(p. 302\)](#).

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 [Cross-account data sharing in Lake Formation \(p. 199\)](#).

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 \(p. 10\)](#) or as any IAM administrative user.
6. If a **Welcome to Lake Formation** window appears, choose the IAM user that you created or selected in Step 1, and then choose **Get started**.
7. If you do not see a **Welcome to Lake Formation** window, then perform the following steps to configure a Lake Formation Administrator.
 - a. In the navigation pane, under **Permissions**, choose **Administrative Roles and tasks**. In the **Data lake administrators** section of the console page, choose **Choose administrators**.
 - b. 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 the default permission model

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 and tag-based access control with Lake Formation permissions.

For more information, see [the section called "Changing the default security settings for your data lake" \(p. 299\)](#).

Important

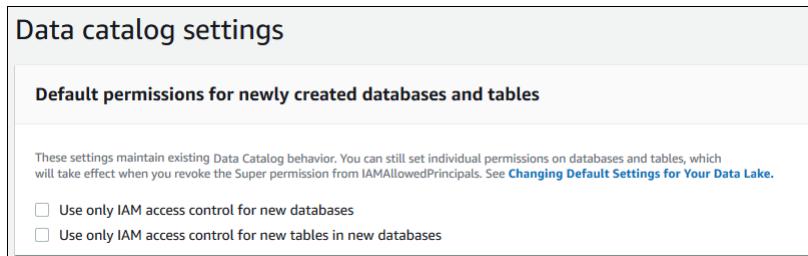
If you have existing AWS Glue Data Catalog databases and tables, do not follow the instructions in this section. Instead, follow the instructions in [Upgrading AWS Glue data permissions to the Lake Formation model \(p. 23\)](#).

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" \(p. 219\)](#).

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 \(p. 10\)](#) or as an IAM user with the **AdministratorAccess** AWS managed policy.
2. Modify the Data Catalog settings:
 - a. In the navigation pane, under **Data catalog**, choose **Settings**.
 - b. Clear both check boxes and choose **Save**.



3. Revoke **IAMAllowedPrincipals** permission for database creators.

- a. In the navigation pane, under **Permissions**, choose **Administrative roles and tasks**.
- b. In the **Administrative roles and tasks** console page, in the **Database creators** section, select the **IAMAllowedPrincipals** group, and choose **Revoke**.

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

- c. Choose **Revoke**.

Create additional Lake Formation users

Create an IAM user to have access to the data lake in AWS Lake Formation. This user has the minimum set of permissions to query the data lake.

To create a non-administrator user with access to Lake Formation data

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 \(p. 10\)](#) or as an IAM user with the **AdministratorAccess** AWS managed policy.
2. Choose **Users**, and then **Add users**.
3. Enter a name for the user, and then choose the **Password - AWS Management Console** access method. Configure the user password requirements. You can optionally choose to enable **Access key - Programmatic access** for this user.

Choose **Next:Permissions**.

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

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

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

7. Back in the IAM **Add user** window, enter `datalake` in the **Filter policy** search field. If the newly created policy doesn't appear in the result list, choose the Refresh, or page reload button. Check the box for the `DatalakeUserBasic` policy.
8. Choose **Next:tags** and **Next:Review**.
9. On the Review page, you should see that both the `DatalakeUserBasic` policy and the `AmazonAthenaFullAccess` policy were chosen for the user. Choose **Create user** to complete the setup.

Configure an Amazon S3 location for your data lake

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

When you register a location, you specify an IAM role that grants read/write permissions on that location. Lake Formation assumes that role when supplying temporary credentials to integrated AWS services that request access to data in the registered Amazon S3 location. You can specify either the Lake Formation service-linked role (SLR) or create your own role.

Use a custom role in the following situations:

- You plan to create governed tables in the registered Amazon S3 location. The user-defined role must include a policy for adding logs in CloudWatch Logs and publishing metrics in addition to the SLR permissions. For an example inline policy that grants the necessary CloudWatch permissions, see [Requirements for roles used to register locations \(p. 107\)](#).
- The Amazon S3 location exists in a different account. For details, see [the section called "Registering an Amazon S3 location in another AWS account" \(p. 113\)](#).
- The Amazon S3 location contains data encrypted with an AWS managed key. For details, see [Registering an encrypted Amazon S3 location \(p. 110\)](#) and [Registering an encrypted Amazon S3 location across AWS accounts \(p. 115\)](#).
- You plan to access the Amazon S3 location using Amazon EMR. For more information about the role requirements, see [IAM roles for Lake Formation](#) in the *Amazon EMR Management Guide*.

The role that you choose must have the necessary permissions, as described in [Requirements for roles used to register locations \(p. 107\)](#). For instructions on how to register an Amazon S3 location, see [Adding an Amazon S3 location to your data lake \(p. 107\)](#).

Prepare for using governed tables and row-level security

The Lake Formation governed tables, row-level filtering, and storage optimization features require additional configuration before using them.

Topics

- [Prepare for using governed tables \(p. 18\)](#)
- [Prepare for using automatic data compaction with governed tables \(p. 20\)](#)
- [Prepare for using row-level security \(p. 21\)](#)

Prepare for using governed tables

To create governed tables in Lake Formation, you must first register an Amazon S3 location in Lake Formation and specify a role that contains all the required permissions, as described previously in [Configure an Amazon S3 location for your data lake \(p. 17\)](#). You then need to grant permissions to the user or role that will be interacting with governed tables. For more information about data access permissions, see [Underlying Data Access Control](#).

To create a governed table the user must be a data lake administrator or a user with the following permissions:

- The Lake Formation CREATE_TABLE permission on the target database
- The AWS Identity and Access Management (IAM) permission glue:CreateTable
- Data location permissions in Lake Formation, as described in [Granting data location permissions \(p. 174\)](#). Data location permissions control the ability to create or alter Data Catalog resources that point to particular Amazon S3 locations.

To access data in a governed table, the principal will need SELECT permissions on the governed table, and IAM permissions to call:

```
lakeformation:StartQueryPlanning
lakeformation:GetQueryState
lakeformation:GetWorkUnits

lakeformation:GetWorkUnitResults

lakeformation:StartTransaction

lakeformation:CommitTransaction

lakeformation:CancelTransaction
lakeformation:ExtendTransaction
```

To create and assign a role to users for creating and using governed tables

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 \(p. 10\)](#) or as an IAM user with the AdministratorAccess AWS managed policy.
2. In the navigation pane, choose **Roles**, then **Create role**.
3. In the **Attach permissions policies** section, choose **Create policy**. In the newly opened browser window, create a new policy to use with your role.
 - a. On the **Create policy** page, choose the **JSON** tab. Copy the following JSON code into the policy editor field.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTable",
                "glue:GetPartitions",
                "glue:UpdateTable"
            ],
            "Resource": "*"
        }
    ]
}
```

```

    "Effect": "Allow",
    "Action": [
        "lakeformation:StartQueryPlanning",
        "lakeformation:GetQueryState",
        "lakeformation:GetWorkUnits",
        "lakeformation:GetWorkUnitResults",
        "lakeformation:GetQueryStatistics",
        "lakeformation:StartTransaction",
        "lakeformation:CommitTransaction",
        "lakeformation:CancelTransaction",
        "lakeformation:ExtendTransaction",
        "lakeformation:DescribeTransaction",
        "lakeformation>ListTransactions",
        "lakeformation:GetTableObjects",
        "lakeformation:UpdateTableObjects",
        "lakeformation>DeleteObjectsOnCancel"
    ],
    "Resource": "*"
}
]
}

```

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

- `lakeformation:StartQueryPlanning` allows principals to submit requests to process a query statement.
 - `lakeformation:GetQueryState` allows principals to view the state of a previously submitted query.
 - `lakeformation:GetWorkUnits` allows principals to retrieve the work units generated by the `StartQueryPlanning` operation.
 - `lakeformation:GetWorkUnitResults` allows principals to view the work units resulting from the query.
 - `lakeformation:GetQueryStatistics` allows principals to retrieve statistics on the planning and execution of a query.
 - `lakeformation:StartTransaction` allows principals and jobs to start a transaction.
 - `lakeformation:CommitTransaction` allows principals and jobs to commit transactions.
 - `lakeformation:CancelTransaction` allows principals and jobs to stop a transaction before the commit.
 - `lakeformation:ExtendTransaction` allows principals and jobs to indicate that the specified transaction is still active and shouldn't be canceled.
 - `lakeformation:DescribeTransaction` allows principals and jobs to list information about a transaction.
 - `lakeformation>ListTransactions` allows principals and jobs to view metadata about transactions and their statuses.
 - `lakeformation:GetTableObjects` allows principals and jobs to list the table objects stored in the data lake.
 - `lakeformation:UpdateTableObjects` allows principals and jobs to update the table objects stored in the data lake.
 - `lakeformation>DeleteObjectsOnCancel` allows principals and jobs to specify a list of Amazon S3 objects that will be written during the current transaction and that can be automatically deleted if the transaction is canceled.
- b. For users that need to manage the data compaction and garbage collection settings for governed tables, add the following permissions to the above policy:

```
"lakeformation:UpdateTableStorageOptimizer",
```

```
"lakeformation>ListTableStorageOptimizers"
```

- c. Choose **Next:Tags**.
 - d. You can optionally add a tag, then choose **Next: Review**.
 - e. On the Review policy page, enter a name for the policy, for example LakeFormationGovernedTables, then choose **Create policy**.
 - f. You can close this window, and return to the **Create role** page.
4. On the **Create role** page, choose the refresh button, then search for the LakeFormationGovernedTables policy you created in the previous step. Select the check box next to the policy name in the list.
 5. Complete the **Create role** wizard by choosing **Next** until you reach the **Review** page. Enter a name for the role, such as LakeFormationTransactionsRole. To finish, choose **Create role**.
 6. Back on the **Roles** page, search for LakeFormationTransactionsRole and choose the role name.
 7. On the role **Summary** page, under the **Permissions** tab, verify that the role has the LakeFormationGovernedTables policy attached.

You can now assign this role to the principals that work with governed tables.

Prepare for using automatic data compaction with governed tables

To configure data compaction for governed tables, the principal must satisfy the following conditions:

- Be the user that created the table or be a data lake administrator user
- Have the glue:UpdateTable, glue:GetTable and Lake Formation ALTER permission on the table

Additionally, the role used when registering the Amazon S3 data lake location with Lake Formation must contain the following permissions to use data compaction:

- s3:PutObject and lakeformation:UpdateTableObjects
- lakeformation:StartTransaction, lakeformation:CommitTransaction, lakeformation:CancelTransaction, lakeformation>DeleteObjectsOnCancel, and logs>CreateLogGroup, logs>CreateLogStream, logs>PutLogEvents, to "arn:aws:logs:/*<**ACCOUNT_ID**>*:log-group:/aws-lakeformation-acceleration/compaction/logs:/*" as in the following example.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObject"  
            ],  
            "Resource": "arn:aws:s3:::<bucket>/<prefix>/*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "lakeformation:StartTransaction",  
                "lakeformation:CommitTransaction",  
                "lakeformation:CancelTransaction",  
                "lakeformation>DeleteObjectsOnCancel"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

```
{
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs>CreateLogGroup",
            "logs>CreateLogStream",
            "logs:PutLogEvents"
        ],
        "Resource": "arn:aws:logs:***ACCOUNT ID***:log-group:/aws-lakeformation-acceleration/compaction/logs:*"
    }
}
```

- If the Data Catalog is encrypted, the AWS KMS key policy must include a trust relationship with `lakeformation.amazonaws.com`, such as the following example.

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "lakeformation.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

Prepare for using row-level security

When you grant Lake Formation permissions on a Data Catalog table, you can include data filtering specifications to restrict access to certain data in query results and within AWS Glue ETL jobs. Lake Formation uses data filtering to achieve column-level security, row-level security, and cell-level security. You can implement column-level, row-level, and cell-level security by creating named *data filters* and specifying a data filter when you grant the SELECT Lake Formation permission on tables. When you create a data filter, you provide a set of columns and a filter expression for rows that need to be included. This allows you to restrict access to certain data in query results and within engines integrated with Lake Formation, such as Athena and AWS Glue ETL jobs.

For more information about data filters, see [Data filtering and cell-level security in Lake Formation \(p. 260\)](#).

To configure row-level security for a table

- Identify the content you want to restrict access to and create data filters. For instructions on how to create data filters, see [the section called “Creating a data filter” \(p. 267\)](#).
- Grant DESCRIBE permission on the data filter to users who will be able to view the data filter.

When you create a data filter, only you can view it. To allow other principals to view and use a data filter, you can grant the DESCRIBE permission on it.

- Specify the data filter when you grant the SELECT Lake Formation permission on tables to principals.

4. Assign IAM permissions to the principals that will query the table using cell-level filters. Principals that query tables with cell-level filtering must have the following IAM permissions:

```
lakeformation:StartQueryPlanning  
lakeformation:GetQueryState  
lakeformation:GetWorkUnits  
lakeformation:GetWorkUnitResults
```

(Optional) External data filtering settings

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

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

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

1. Continue in the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Ensure that you are signed in as a principal that has the IAM permission on the Lake Formation PutDataLakeSettings API operation. The IAM administrator user that you created in [Create an Administrator IAM User \(p. 10\)](#) 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. Check the box **Allow external engines to filter data in Amazon S3 locations registered with Lake Formation**.
 - b. Enter **Session tag values** defined for third-party engines.
 - c. For **AWS account IDs**, enter the account IDs from where third-party engines are allowed to access locations registered with Lake Formation. Press **Enter** after each account ID.
 - d. Choose **Save**.

(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*.

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” \(p. 290\)](#).

Topics

- [About upgrading to the Lake Formation permissions model \(p. 23\)](#)
- [Step 1: List users' and roles' existing permissions \(p. 24\)](#)
- [Step 2: Set up equivalent Lake Formation permissions \(p. 25\)](#)
- [Step 3: Give users IAM permissions to use Lake Formation \(p. 26\)](#)
- [Step 4: Switch your data stores to the Lake Formation permissions model \(p. 26\)](#)
- [Step 5: Secure new Data Catalog resources \(p. 28\)](#)
- [Step 6: Give users a new IAM policy for future data lake access \(p. 29\)](#)
- [Step 7: Clean up existing IAM policies \(p. 29\)](#)

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" \(p. 24\)](#).

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 \(p. 9\)](#).

Topics

- [Using the API operation \(p. 24\)](#)
- [Using the AWS Management Console \(p. 25\)](#)
- [Using AWS CloudTrail \(p. 25\)](#)

Using the API operation

Use the AWS Identity and Access Management (IAM) `ListPoliciesGrantingServiceAccess` API operation to determine the IAM policies attached to each principal (user or role). From the policies returned in the results, you can determine the IAM permissions that are granted to the principal. You must invoke the API for each principal separately.

Example

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

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

The command returns results similar to the following.

```
{  
    "PoliciesGrantingServiceAccess": [  
        {  
            "ServiceNamespace": "glue",  
            "Policies": [  
                {  
                    "PolicyType": "INLINE",  
                    "PolicyName": "GlueUserBasic",  
                    "EntityName": "glue_user1",  
                    "Arn": "arn:aws:iam::111122223333:policy/GlueUserBasic",  
                    "CreateDate": "2018-01-12T12:00:00Z",  
                    "LastUsedDate": "2018-01-12T12:00:00Z",  
                    "Version": "2018-01-12T12:00:00Z"  
                }  
            ]  
        }  
    ]  
}
```

```
        "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 \(p. 25\)](#).

Step 2: Set up equivalent Lake Formation permissions

Using the information collected in [Step 1: List users' and roles' existing permissions \(p. 24\)](#), 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" \(p. 180\)](#).
- Use the GrantPermissions or BatchGrantPermissions API operations.

See [Permissions APIs \(p. 325\)](#).

For more information, see [Overview of Lake Formation permissions \(p. 171\)](#).

After setting up Lake Formation permissions, proceed to [Step 3: Give users IAM permissions to use Lake Formation \(p. 26\)](#).

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 \(p. 26\)](#).

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 \(p. 26\)](#)
- [Secure existing Data Catalog resources \(p. 27\)](#)
- [Turn on Lake Formation permissions for your Amazon S3 location \(p. 28\)](#)

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” \(p. 219\)](#).

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 \(p. 107\)](#). Or, use the `RegisterResource` API operation as described in [Credential Vending API \(p. 338\)](#).

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 \(p. 28\)](#).

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" \(p. 219\)](#).

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 \(p. 29\)](#).

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 \(p. 371\)](#).

Next, proceed to [Step 7: Clean up existing IAM policies \(p. 29\)](#).

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.

Tutorials

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

Note

The order in which you go through the tutorials is not important. The users you create in the first tutorial can be used in the second tutorial. The rest of the tutorials include AWS CloudFormation templates to create required resources.

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

Before you begin, make sure that you have completed the steps in [Setting up AWS Lake Formation \(p. 9\)](#).

- [Creating a data lake from an AWS CloudTrail source \(p. 32\)](#)

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

- [Creating a data lake from a JDBC source in Lake Formation \(p. 39\)](#)

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

- [Creating a governed table in Lake Formation \(p. 48\)](#)

This tutorial demonstrates how to set up governed tables in Lake Formation. For more information about governed tables, see [Governed tables in Lake Formation \(p. 120\)](#).

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

- [Managing a data lake using Lake Formation tag-based access control \(p. 65\)](#)

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

- [Securing data lakes with row-level access control \(p. 79\)](#)

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

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

- [Sharing a data lake using Lake Formation tag-based access control and named resources \(p. 88\)](#)

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

- [Sharing a data lake using Lake Formation fine-grained access control \(p. 102\)](#)

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

Topics

- [Creating a data lake from an AWS CloudTrail source \(p. 32\)](#)

- [Creating a data lake from a JDBC source in Lake Formation \(p. 39\)](#)
- [Creating a governed table in Lake Formation \(p. 48\)](#)
- [Managing a data lake using Lake Formation tag-based access control \(p. 65\)](#)
- [Securing data lakes with row-level access control \(p. 79\)](#)
- [Sharing a data lake using Lake Formation tag-based access control and named resources \(p. 88\)](#)
- [Sharing a data lake using Lake Formation fine-grained access control \(p. 102\)](#)

Creating a data lake from an AWS CloudTrail source

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

High-level steps for creating a data lake

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

Topics

- [Intended audience \(p. 32\)](#)
- [Prerequisites \(p. 33\)](#)
- [Step 1: Create the IAM user to be the data analyst \(p. 33\)](#)
- [Step 2: Add permissions to read AWS CloudTrail logs to the workflow role \(p. 34\)](#)
- [Step 3: Create an Amazon S3 bucket for the data lake \(p. 34\)](#)
- [Step 4: Register an Amazon S3 path \(p. 34\)](#)
- [Step 5: Grant data location permissions \(p. 35\)](#)
- [Step 6: Create a database in the Data Catalog \(p. 35\)](#)
- [Step 7: Grant data permissions \(p. 35\)](#)
- [Step 8: Use a blueprint to create a workflow \(p. 37\)](#)
- [Step 9: Run the workflow \(p. 37\)](#)
- [Step 10: Grant SELECT on the tables \(p. 38\)](#)
- [Step 11: Query the data lake Using Amazon Athena \(p. 38\)](#)

Intended audience

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

Intended audience

| Role | 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. |

Prerequisites

Before you begin:

- Ensure that you have completed the tasks in [Setting up AWS Lake Formation \(p. 9\)](#).
- 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 \(p. 10\)](#) 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:GetCrawler"
      ]
    }
  ]
}
```

```
        "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 \(p. 10\)](#).
2. Choose **Create bucket**, and go through the wizard to create a bucket named `<yourName>-datalake-cloudtrail`, where `<yourName>` is your first initial and last name. For example: `jdoe-datalake-cloudtrail`.

For detailed instructions on creating an Amazon S3 bucket, see [Creating a bucket](#).

Step 4: Register an Amazon S3 path

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

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

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

For more information about registering locations, see [Adding an Amazon S3 location to your data lake \(p. 107\)](#).

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 \(p. 295\)](#).

Step 6: Create a database in the Data Catalog

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

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

Step 7: Grant data permissions

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

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

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

LF-Tags or catalog resources

Resources matched by LF-Tags (recommended)

Manage permissions indirectly for resources or data matched by a specific set of LF-Tags.

Named data catalog resources

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

Databases

Select one or more databases.

Choose databases ▾

Load more

lakeformation-clouptrail X
007436865787

Tables - optional

Select one or more tables.

Choose tables ▾

Load more

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.

4. Choose Grant.

For more information about granting Lake Formation permissions, see [Security and access Control to metadata and data in Lake Formation \(p. 289\)](#).

Step 8: Use a blueprint to create a workflow

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

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

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

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

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

| | |
|----------------------|-----------------------------|
| Workflow name | lakeformationcloudtrailtest |
| IAM role | LakeFormationWorkflowRole |
| Table prefix | cloudtrailtest |

Note
Must be lower case.

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

Tip

Did you get the following error message?

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

Step 9: Run the workflow

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

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

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

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

When the workflow completes:

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

If the workflow fails, do the following:

- a. Select the workflow, and on the **Actions** menu, choose **View graph**.
The workflow opens in the AWS Glue console.
- b. Ensure that the workflow is selected, and choose the **History** tab.
- c. Under **History**, select the most recent run and choose **View run details**.
- d. Select a failed job or crawler in the dynamic (runtime) graph, and review the error message.
Failed nodes are either red or yellow.

Step 10: Grant SELECT on the tables

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

Note

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

1. In the Lake Formation console, in the navigation pane, under **Data catalog**, choose **Databases**.
2. Choose the `lakeformation_cloudtrail` database, then, from the **Actions** drop-down list, choose **Grant** under the heading **Permissions**.
3. In the **Grant data permissions** dialog box, make these selections:
 - a. Under **Principals**, for **IAM user and roles**, choose `datalake_user`.
 - b. Under **LF-Tags or catalog resources**, choose **Named data catalog resources**.
 - c. For **Databases**, the `lakeformation_cloudtrail` database should already be selected.
 - d. For **Tables**, choose `cloudtrailtest-cloudtrail`.
 - e. Under **Table and column permissions**, choose **Select**.
4. Choose **Grant**.

The next step is performed as the data analyst.

Step 11: Query the data lake Using Amazon Athena

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

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

The **Tables** list populates.

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

The query runs and displays 10 rows of data.

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

Note

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

Creating a data lake from a JDBC source in Lake Formation

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

Topics

- [Intended audience \(p. 39\)](#)
- [JDBC tutorial prerequisites \(p. 40\)](#)
- [Step 1: Create the IAM user to be the data analyst \(p. 40\)](#)
- [Step 2: Create a connection in AWS Glue \(p. 41\)](#)
- [Step 3: Create an Amazon S3 bucket for the data lake \(p. 41\)](#)
- [Step 4: Register an Amazon S3 path \(p. 42\)](#)
- [Step 5: Grant data location permissions \(p. 42\)](#)
- [Step 6: Create a database in the Data Catalog \(p. 42\)](#)
- [Step 7: Grant data permissions \(p. 42\)](#)
- [Step 8: Use a blueprint to create a workflow \(p. 43\)](#)
- [Step 9: Run the workflow \(p. 44\)](#)
- [Step 10: Grant SELECT on the tables \(p. 44\)](#)
- [Step 11: Query the data lake using Amazon Athena \(p. 45\)](#)
- [Step 12: Query the data in the data lake using Amazon Redshift Spectrum \(p. 45\)](#)
- [Step 13: Grant or revoke Lake Formation permissions using Amazon Redshift Spectrum \(p. 48\)](#)

Intended audience

The following table lists the roles that are used in this [AWS Lake Formation JDBC tutorial \(p. 39\)](#).

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

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

For information about prerequisites for completing the tutorial, see [JDBC tutorial prerequisites \(p. 40\)](#).

JDBC tutorial prerequisites

Before you begin the [AWS Lake Formation JDBC tutorial \(p. 39\)](#), ensure that you've done the following:

- Complete the tasks in [Setting up AWS Lake Formation \(p. 9\)](#).
- 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" \(p. 40\)](#).

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 \(p. 10\)](#) 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 the 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 \(p. 10\)](#).
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 \(p. 10\)](#).
2. Choose **Create bucket**, and go through the wizard to create a bucket named `<yourName>-datalake-tutorial`, where `<yourName>` is your first initial and last name. For example: `jdoe-datalake-tutorial`.

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

Step 4: Register an Amazon S3 path

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

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Sign in as the data lake administrator.
2. In the navigation pane, under **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 `AWSLambdaRoleForLakeFormationDataAccess`, and then choose **Register location**.

For more information about registering locations, see [Adding an Amazon S3 location to your data lake \(p. 107\)](#).

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 \(p. 295\)](#).

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 **Data lake 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 **LF-Tags or catalog resources**, choose **Named data catalog resources**.
 - c. For **Databases**, choose the database that you created previously, **lakeformation_tutorial**.
 - d. Under **Database permissions**, select **Create table**, **Alter**, and **Drop**, and clear **Super** if it is selected.
3. Choose **Grant**.

For more information about granting Lake Formation permissions, see [Security and access Control to metadata and data in Lake Formation \(p. 289\)](#).

Step 8: Use a blueprint to create a workflow

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

1. On the Lake Formation console, in the navigation pane, choose **Blueprints**, and then choose **Use blueprint**.
2. On the **Use a blueprint** page, under **Blueprint type**, choose **Database snapshot**.
3. Under **Import source**, for **Database connection**, choose the connection that you just created, **datalake-tutorial**, or choose an existing connection for your data source.
4. For **Source data path**, enter the path from which to ingest data, in the form **<database>/<schema>/<table>**.

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

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

Important

This field is case-sensitive.

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

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

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

| | |
|----------------------|---------------------------|
| Workflow name | lakeformationjdbctest |
| IAM role | LakeFormationWorkflowRole |
| Table prefix | jdbctest |

Note
Must be lower case.

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

Tip

Did you get the following error message?

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

Step 9: Run the workflow

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

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

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

When the workflow is complete:

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

If the workflow fails, do the following:

- a. Select the workflow. Choose **Actions**, and then choose **View graph**.
The workflow opens in the AWS Glue console.
- b. Select the workflow and choose the **History** tab.
- c. Select the most recent run and choose **View run details**.
- d. Select a failed job or crawler in the dynamic (runtime) graph, and review the error message.
Failed nodes are either red or yellow.

Step 10: Grant SELECT on the tables

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

Note

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

1. On the Lake Formation console, in the navigation pane, under **Permissions**, choose **Data permissions**.
2. Choose **Grant**, and in the **Grant data permissions** dialog box, do the following:
 - a. Under **Principals**, for **IAM user and roles**, choose `datalake_user`.
 - b. Under **LF-Tags or catalog resources**, choose **Named data catalog resources**.

- c. For **Databases**, choose `lakeformation_tutorial`.
The **Tables** list populates.
 - d. For **Tables**, choose one or more tables from your data source.
 - e. Under **Table and column permissions**, choose **Select**.
3. Choose **Grant**.

The next step is performed as the data analyst.

Step 11: Query the data lake using Amazon Athena

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

1. Open the Athena console at <https://console.aws.amazon.com/athena/>, and sign in as the data analyst, user `datalake_user`.
2. If necessary, choose **Get Started** to continue to the Athena query editor.
3. For **Data source**, choose **AwsDataCatalog**.
4. For **Database**, choose `lakeformation_tutorial`.
The **Tables** list populates.
5. In the pop-up menu beside one of the tables, choose **Preview table**.

The query runs and displays 10 rows of data.

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

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

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

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

To set up permissions to run Amazon Redshift queries

1. Open the IAM console at <https://console.aws.amazon.com/iam/>. Sign in as the IAM administrator user that you created in [Create an Administrator IAM User \(p. 10\)](#) (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": "*"
    }
]
```

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

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

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

The tables list populates.

 - For **Table**, choose a table within the data source to query.
 - Choose the **Select** table permission.

4. Choose **Grant**.

To set up Amazon Redshift Spectrum and run queries

1. Open the Amazon Redshift console at <https://console.aws.amazon.com/redshift>. Sign in as the user **Administrator**.
2. Choose **Create cluster**.
3. On the **Create cluster** page, enter `redshift-lakeformation-demo` for the **Cluster identifier**.
4. For the **Node type**, select `dc2.large`.
5. Scroll down, and under **Database configurations**, enter or accept these parameters:
 - **Admin user name:** `awsuser`
 - **Admin user password:** (*Choose a password*)
6. Expand **Cluster permissions**, and for **Available IAM roles**, choose `RedshiftLakeFormationRole`. Then choose **Add IAM role**.
7. If you must use a different port than the default value of 5439, next to **Additional configurations**, turn off the **Use defaults** option. Expand the section for **Database configurations**, and enter a new **Database port** number.
8. Choose **Create cluster**.

The **Clusters** page loads.

9. Wait until the cluster status becomes **Available**. Choose the refresh icon periodically.
10. Grant the data analyst permission to run queries against the cluster. To do so, complete the following steps.
 - a. Open the IAM console at <https://console.aws.amazon.com/iam/>, and sign in as the **Administrator** user.
 - b. In the navigation pane, choose **Users**, and attach the following managed policies to the user `datalake_user`.
 - `AmazonRedshiftQueryEditor`
 - `AmazonRedshiftReadOnlyAccess`
11. Sign out of the Amazon Redshift console and sign back in as user `datalake_user`.
12. In the left vertical toolbar, choose the **EDITOR** icon to open the query editor and connect to the cluster. If the **Connect to database** dialog box appears, choose the cluster name `redshift-lakeformation-demo`, and enter the database name `dev`, the user name `awsuser`, and the password that you created. Then choose **Connect to database**.

Note

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

13. In the **New Query 1** text box, enter and run the following statement to map the database `lakeformationTutorial` 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 'lakeformationTutorial' iam_role 'arn:aws:iam::<account-id>:role/
  RedshiftLakeFormationRole' region '<region>';
```

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

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

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

Amazon Redshift returns the first 10 rows.

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

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

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

Creating a governed table in Lake Formation

AWS Lake Formation supports atomic, consistent, isolated, and durable (ACID) transactions using a new data lake table type called *Governed Tables*. Lake Formation transactions simplify ETL script and workflow development, and allow multiple users to concurrently and reliably insert, delete, and modify governed tables. Lake Formation automatically compacts and optimizes storage of governed tables in the background to improve query performance.

In this tutorial, you learn how to create a new governed table using existing data on Amazon S3. It also explains how to query governed tables and how to run time travel queries using Amazon Athena.

Note

Since this tutorial uses a public Amazon S3 bucket, it only includes read-only use cases. In real-world situations, you may want to do more by putting objects to your Amazon S3 buckets, adding them to the governed table, and enabling compaction.

This tutorial includes an AWS CloudFormation template for quick set up. You can review and customize it to suit your needs. If you prefer setting up resources on [AWS console](#), see the instructions in [Step 1: Provision your resources \(p. 49\)](#).

Topics

- [Intended audience \(p. 48\)](#)
- [Prerequisites \(p. 49\)](#)
- [Step 1: Provision your resources \(p. 49\)](#)
- [Step 2: Set up a governed table \(p. 52\)](#)
- [Step 3: Configure Lake Formation permissions \(p. 58\)](#)
- [Step 4: Add table objects into the governed table \(p. 58\)](#)
- [Step 5: Querying the governed table using Amazon Athena \(p. 61\)](#)
- [Step 6: Clean up AWS resources \(p. 64\)](#)

Intended audience

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

| Role | Description |
|-------------------------|--|
| IAM Administrator | A user who can create IAM users and roles and Amazon S3 buckets. Has the <code>AdministratorAccess</code> 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. |

Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in to as an IAM user with correct permissions. For more information, see [Sign up for AWS \(p. 10\)](#) and [Create an Administrator IAM User \(p. 10\)](#).

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

Step 1: Provision your resources

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

- IAM users, roles, and policies
- Lake Formation data lake settings and permissions

This section shows you how to set up the AWS resources in two different ways:

1. Using an AWS CloudFormation template
2. Using the AWS console

Creating your resources using AWS CloudFormation template

Complete the following steps to create your resources using the AWS CloudFormation template:

1. Sign in to the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation> as an IAM user in the US East (N. Virginia) Region.
2. Choose [Launch Stack](#).
3. Choose [Next](#) on the [Create stack](#) screen.
4. Enter a [Stack name](#).
5. For **DatalakeAdminUserName** and **DatalakeAdminUserPassword**, enter your IAM user name and password for data lake admin user.
6. For **DatalakeAnalystUserName** and **DatalakeAnalystUserPassword**, enter your IAM user name and password for data lake analyst user.
7. For **DataLakeBucketName**, enter your new bucket name that will be created.
8. For **DatabaseName**, leave as the default.

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

The stack creation can take up to two minutes.

Create your resources using AWS console

Complete the following steps to create the resources using AWS console:

1. First, you need to set up two IAM roles; one for AWS Glue ETL jobs, and the other for the Lake Formation data lake location. To create IAM policies, complete the following steps:
 - a. On the IAM console (<https://console.aws.amazon.com/iam/>), create a new policy for Amazon S3. Save the following policy as S3DataLakePolicy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObject",  
                "s3:GetObject",  
                "s3:DeleteObject"  
            ],  
            "Resource": [  
                "arn:aws:s3:::your-datalake-bucket-name/*"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3>ListBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::your-datalake-bucket-name"  
            ]  
        }  
    ]  
}
```

- b. Create a new IAM policy named LFLocationPolicy with the following statements:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "LFtransactions",  
            "Effect": "Allow",  
            "Action": [  
                "lakeformation:StartTransaction",  
                "lakeformation:CommitTransaction",  
                "lakeformation:CancelTransaction",  
                "lakeformation:GetTableObjects",  
                "lakeformation:UpdateTableObjects"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

```
{
    "Effect": "Allow",
    "Action": [
        "glue:GetDatabase",
        "glue:GetDatabases",
        "glue:GetTableVersions",
        "glue:GetPartitions",
        "glue:GetTable",
        "glue:GetTables",
        "glue:UpdateTable"
    ],
    "Resource": "*"
}
]
```

- c. Create a new IAM policy named LFQuery with the following statements:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LFtransactions",
            "Effect": "Allow",
            "Action": [
                "lakeformation:StartTransaction",
                "lakeformation:CommitTransaction",
                "lakeformation:CancelTransaction",
                "lakeformation:ExtendTransaction",
                "lakeformation:StartQueryPlanning",
                "lakeformation:GetTableObjects",
                "lakeformation:GetQueryState",
                "lakeformation:GetWorkUnits",
                "lakeformation:GetWorkUnitResults"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Next, complete the following steps to create your IAM roles for Lake Formation data location:

- Create a new Lake Formation role called LFRegisterLocationServiceRole with a Lake Formation trust relationship:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "lakeformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Attach the customer managed policies (S3DataLakePolicy and LFLocationPolicy) you created in the previous step. This role is used to register locations with Lake Formation, which in-turn performs credential vending for Athena at query time.

3. Next, follow these steps to create your IAM users:
 - a. [Create an IAM user](#) named DatalakeAdmin and attach the following AWS managed policies:
 1. AWSLakeFormationDataAdmin
 2. AmazonAthenaFullAccess
 3. IAMReadOnlyAccess
 - b. For **DataLakeBucketName**, enter your new bucket name that will be created.
 - c. Attach the customer managed policy LFQueryPolicy.
 - d. Create an IAM user named DataAnalyst that can use Athena to query data.
 - e. Attach the AWS managed policy AmazonAthenaFullAccess.
 - f. Attach the customer managed policy LFQueryPolicy.
4. Follow below steps to configure Lake Formation:
 - a. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Under **Permissions**, choose **Admins** and database creators.
 - b. In the **Data lake administrators** section, choose **Grant**.
 - c. For **IAM users and roles**, choose your IAM user **DatalakeAdmin**.
 - d. Choose **Save**.
 - e. In the **Database creators** section, choose **Grant**.
 - f. For **IAM users and roles**, choose the **LFRegisterLocationServiceRole**.
 - g. Select **Create Database**.
 - h. Choose **Grant**.
 - i. Under **Register and ingest**, choose **Data lake locations**.
 - j. Choose **Register location**.
 - k. Select **Database**.
 - l. For **Amazon S3 path**, enter your Amazon S3 location of the bucket where your data is stored. This must be the same bucket you listed in **LFLocationPolicy**. Lake Formation uses this role to vend temporary Amazon S3 credentials to query services that need read/write access to the bucket and all prefixes under it.
 - m. For **IAM role**, choose the **LFRegisterLocationServiceRole**.
 - n. Choose **Register location**.
 - o. Under **Data Catalog**, choose **Settings**.
 - p. Make sure that both check boxes for **Use only IAM access control for new databases** and **Use only IAM access control for new tables in new databases** are deselected.
 - q. Under **Data catalog**, choose **Databases**.
 - r. Choose **Create database**.
 - s. For **Name**, enter **lakeformation_tutorial_amazon_reviews**.
 - t. Choose **Create database**.

Step 2: Set up a governed table

Now you can create and configure your first governed table in Lake Formation.

Creating a governed table

1. Sign into Lake Formation console at <https://console.aws.amazon.com/lakeformation/> as the **DatalakeAdmin1** user.

2. Choose **Tables**.
3. Choose **Create table**.
4. For name, enter `amazon_reviews_governed`.
5. For database, enter `lakeformation_tutorial_amazon_reviews`.
6. Select **Enable governed data access and management**.

Create table

Table details

Create a table in the AWS Glue Data Catalog



Table

Create a table in my account

Name

amazon_reviews_governed

Name may contain letters (A-Z), numbers (0-9), underscores (_), and hyphens (-).

Database⁵⁴

Table is contained within this database.

7. For **Data located in**, choose **Specified path in my account**.
8. Enter the path `s3://your-datalake-bucket-name/parquet/` where `your-datalake-bucket-name` is the bucket name you entered in the AWS CloudFormation template.
9. For **Classification**, choose **PARQUET**.
10. Choose **Upload Schema**.
11. Enter the following Json array into the text box.

```
[  
  {  
    "Name": "marketplace",  
    "Type": "string"  
  },  
  {  
    "Name": "customer_id",  
    "Type": "string"  
  },  
  {  
    "Name": "review_id",  
    "Type": "string"  
  },  
  {  
    "Name": "product_id",  
    "Type": "string"  
  },  
  {  
    "Name": "product_parent",  
    "Type": "string"  
  },  
  {  
    "Name": "product_title",  
    "Type": "string"  
  },  
  {  
    "Name": "star_rating",  
    "Type": "int"  
  },  
  {  
    "Name": "helpful_votes",  
    "Type": "int"  
  },  
  {  
    "Name": "total_votes",  
    "Type": "int"  
  },  
  {  
    "Name": "vine",  
    "Type": "string"  
  },  
  {  
    "Name": "verified_purchase",  
    "Type": "string"  
  },  
  {  
    "Name": "review_headline",  
    "Type": "string"  
  },  
  {  
    "Name": "review_body",  
    "Type": "string"  
  },  
  {  
    "Name": "review_date",  
    "Type": "bigint"  
}
```

```
        },
        {
            "Name": "year",
            "Type": "int"
        }
    ]
```

12. Choose **Upload**.
13. Choose **Add column**.
14. For column name, enter `product_category`.
15. For data type, choose **String**.
16. Select **Partition key**.
17. Choose **Add**.
18. Choose **Submit**.

Now you can see that the new governed table has been created.

When you choose the table name, you can see the details of the governed table, and you can also see **Governance: Enabled** in this view. This means that this table is a Lake Formation governed table. Tables that are not governed should show as **Governance: Disabled**.

AWS Lake Formation > Tables > amazon_reviews_governed

amazon_reviews_governed

Choose

Actions ▾

Compare versions

Drop table

Table details

Database

[lakeformationTutorial_amazon_reviews](#)

Location

<s3://amazon-reviews-pds/parquet/> ↗

Connection

-

▼ Advanced table properties

Input format

`org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputForm`

Serialization lib

`org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe`

By default, automatic compaction is enabled for governed tables. To disable automatic compaction using the AWS AWS CLI and SDK, run the following command:

```
$ aws lakeformation update-table-storage-optimizer --database-name lakeformation_tutorial_amazon_reviews --table-name amazon_reviews_governed --storage-optimizer-config '{"compaction": {"is_enabled": "false"}}'
```

Step 3: Configure Lake Formation permissions

Currently, the governed table you created in [the section called “Step 2: Set up a governed table” \(p. 52\)](#) does not contain any data or partitions. In the next step, you add existing Amazon S3 objects to this governed table using manifest APIs. Even if your data is located in the table location of the governed table, the data is not recognized until you add it to the governed table. Before adding objects to the governed table, configure Lake Formation permissions to grant required permissions to users.

Configure Lake Formation permissions

To grant users required Lake Formation permissions to your governed table, complete the following steps:

1. Sign in to the Lake Formation console at <https://console.aws.amazon.com/lakeformation/> in US East (N. Virginia) Region as the DatalakeAdmin1 user.
2. Under **Permissions**, choose **Data lake permissions**.
3. Under **Data permission**, choose **Grant**.
4. For **Principals**, choose **IAM users and roles**, and select the role LFRegisterLocationServiceRole - *CloudFormation stack name* and the user DatalakeAdmin1.
5. For **Policy tags or catalog resources**, choose **Named data catalog resources**.
6. For **Database**, choose lakeformation_tutorial_amazon_reviews.
7. For **Table**, choose amazon_reviews_governed.
8. For **Table permissions**, select **Select, Describe, Insert, Alter, Delete, and Drop**.
9. For **Data permissions**, select **All data access**.
10. Choose **Grant**.
11. Under **Permissions**, choose **Data lake permissions**.
12. Under **Data permission**, choose **Grant**.
13. For **Principals**, choose **IAM users and roles**, and select the user DataAnalyst1.
14. For **Policy tags or catalog resources**, choose **Named data catalog resources**.
15. For **Database**, choose lakeformation_tutorial_amazon_reviews.
16. For **Table**, choose amazon_reviews_governed.
17. For **Table permissions**, select **Select and Describe**.
18. For **Data permissions**, select **All data access**.
19. Choose **Grant**.

Step 4: Add table objects into the governed table

To add Amazon S3 objects to a governed table, you need to call the `UpdateTableObjects` API. You can call it using the AWS Command Line Interface (AWS CLI) and SDK, and also the AWS Glue ETL library (the API is called implicitly in the library). For this tutorial, we use the AWS CLI to explain the behavior at

the API level. To install AWS CLI, see [Installing or updating the latest version of the AWS Command Line Interface](#).

To simplify the example, we only add one partition with one file. In real-world usage, you may want to add all the files within all the partitions that you need.

1. Run the following commands using DatalakeAdmin1 user's credential.
2. First, start a new transaction with the StartTransaction API.

```
$ aws lakeformation start-transaction
{
    "TransactionId": "396880372a0045dc9c8faff2d19dfea"
}
```

3. Now you can add files to this table within the transaction. To add files, choose a sample partition called `product_category=Camera` from the `amazon-reviews-pds` table, and choose one file under this partition. You need to know the Uri, ETag, and Size of the files you add. To find this information, enter the following AWS CLI command (replace `your-datalake-bucket-name` with a valid bucket name):

```
$ aws s3 ls s3://your-datalake-bucket-name/parquet/product_category=Camera/
2021-11-23 18:19:19    65227429 part-00004-495c48e6-96d6-4650-
aa65-3c36a3516ddd.c000.snappy.parquet

$ aws s3api head-object --bucket your-datalake-bucket-name --key
    parquet/product_category=Camera/part-00004-495c48e6-96d6-4650-
aa65-3c36a3516ddd.c000.snappy.parquet
{
    "AcceptRanges": "bytes",
    "LastModified": "Mon, 09 Apr 2018 06:37:07 GMT",
    "ContentLength": 65227429,
    "ETag": "\"980669fcf6ccf31d2d686b9ccccdd45e3-8\"",
    "ContentType": "binary/octet-stream",
    "Metadata": {}
}
```

4. Create a new file named `write-operations1.json` and enter the following JSON: (replace Uri, ETag, and Size with the values you copied.)

```
[
  {
    "AddObject": {
      "Uri": "s3://your-datalake-bucket-name/parquet/product_category=Camera/
part-00000-495c48e6-96d6-4650-aa65-3c36a3516ddd.c000.snappy.parquet",
      "ETag": "d4c25c40f33071620fb31cf0346ed2ec-8",
      "Size": 65386769,
      "PartitionValues": [
        "Camera"
      ]
    }
  }
]
```

5. To add a file to the governed table, use the `UpdateTableObjects` API call using the `write-operations1.json` file you just created. Replace `transaction-id` with the transaction id you got in `start-transaction` command.

```
$ aws lakeformation update-table-objects --database-name
lakeformation_tutorial_amazon_reviews --table-name amazon_reviews_governed --
transaction-id transaction-id --write-operations file://./write-operations1.json
```

6. Next, inspect the changes before a transaction is committed by calling the GetTableObjects API with the same transaction ID: (replace *transaction-id* with the id you got in start-transaction command). You will need to run this command twice because the first time the changes above have not been applied to the transaction. The second time, the UpdateTableObjects operation will have succeeded.

```
$ aws lakeformation get-table-objects --database-name
    lakeformation_tutorial_amazon_reviews --table-name amazon_reviews_governed --
    transaction-id transaction-id
{
    "Objects": [
        {
            "PartitionValues": [
                "Camera"
            ],
            "Objects": [
                {
                    "Uri": "s3://your-datalake-bucket-name/parquet/product_category=Camera/part-00000-495c48e6-96d6-4650-aa65-3c36a3516ddd.c000.snappy.parquet",
                    "ETag": "d4c25c40f33071620fb31cf0346ed2ec-8",
                    "Size": 65386769
                }
            ]
        }
    ]
}
```

7. Commit the transaction so this data is available outside this transaction to other users. To do this, call the CommitTransaction API: (replace *transaction-id* with the transaction id you got in start-transaction command).

```
$ aws lakeformation commit-transaction --transaction-id transaction-id
```

Note the current date-time right after committing the change of the UpdateTableObjects API call here. We use this time stamp for time travel queries later in this example.

```
$ date -u
Wed May 26 08:12:00 UTC 2021
```

After you commit the transaction, you can see the partition on the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.

Use the following commands to add all files within all partitions:

1. Call the StartTransaction API to start another Lake Formation transaction.

```
$ aws lakeformation start-transaction
{
    "TransactionId": "c96cdbeab7e34b35a383faa75b372234"
}
```

2. List Amazon S3 objects located on amazon-reviews-pds bucket to choose another sample file.

```
s3://your-datalake-bucket-name/parquet/product_category=Books/
```

```
2018-04-09 15:35:58 1094842361 part-00000-495c48e6-96d6-4650-
aa65-3c36a3516ddd.c000.snappy.parquet
```

3. Call the HeadObject API against one sample file in order to copy ETag and Size.

```
$ aws s3api head-object --bucket your-datalake-bucket-name --key
parquet/product_category=Books/part-00000-495c48e6-96d6-4650-
aa65-3c36a3516ddd.c000.snappy.parquet
{
    "AcceptRanges": "bytes",
    "LastModified": "Mon, 09 Apr 2018 06:35:58 GMT",
    "ContentLength": 1094842361,
    "ETag": "\"9805c2c9a0459ccf337e01dc727f8efc-131\"",
    "ContentType": "binary/octet-stream",
    "Metadata": {}
}
```

4. Create a new file named write-operations2.json and enter the following JSON: (replace *Uri*, *ETag*, and *Size* with the values you copied.)

```
[
    {
        "AddObject": {
            "Uri": "s3://your-datalake-bucket-name/parquet/product_category=Books/
part-00000-495c48e6-96d6-4650-aa65-3c36a3516ddd.c000.snappy.parquet",
            "ETag": "9805c2c9a0459ccf337e01dc727f8efc-131",
            "Size": 1094842361,
            "PartitionValues": [
                "Books"
            ]
        }
    }
]
```

5. Call the UpdateTableObjects API using write-operations2.json: (replace *transaction-id* with the transaction id you got in start-transaction command).

```
$ aws lakeformation update-table-objects --database-name
lakeformation_tutorial_amazon_reviews --table-name amazon_reviews_governed --
transaction-id transaction-id --write-operations file://./write-operations2.json
```

6. Call the CommitTransaction API: (replace *transaction-id* with the transaction id you got in the start-transaction command).

```
$ aws lakeformation commit-transaction --transaction-id transaction-id
```

Now the two partitions are visible on the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.

Step 5: Querying the governed table using Amazon Athena

Now start querying the governed table you created using Amazon Athena.

If it is the first time you are running queries in Athena, you need to configure a query result location. For more information, see [Specifying a query result location](#).

Running a simple query

- To run a simple query, sign in to the Athena console at <https://console.aws.amazon.com/athena/> in US East (N. Virginia) region as the DataAnalyst1 user. Run the following query to preview 10 records stored in a governed table:

```
SELECT *  
FROM lakeformationTutorial_amazon_reviews.amazon_reviews_governed  
LIMIT 10
```

The results appears as below:

The screenshot shows the AWS Athena Query Editor interface. At the top, there are three tabs: "New query 1" (selected), "New query 2", and "New query 3". Below the tabs is a code editor containing the following SQL query:

```
1 SELECT *
2 FROM lakeformationTutorial_amazon_reviews.amazon_reviews
3 LIMIT 10
```

Below the code editor are three buttons: "Run query" (highlighted in blue), "Save as", and "Create". To the right of these buttons is the text "(Run time: 13.38 seconds, Da)". A note at the bottom says "Use Ctrl + Enter to run query, Ctrl + Space to autocomplete".

The screenshot shows the "Results" page of the AWS Athena interface. The table has the following columns: marketplace, customer_id, review_id, and product_id. The data is as follows:

| | marketplace | customer_id | review_id | product_id |
|----|-------------|-------------|----------------|------------|
| 1 | US | 12115179 | R249G3JHC13PYV | 0765356159 |
| 2 | US | 30255545 | R17L9M1OUKCWY0 | 0764209833 |
| 3 | US | 10609904 | R3FAAKHI4W47AK | 1634154791 |
| 4 | US | 32941543 | R2VWIDVTOLEYCP | 1616385944 |
| 5 | US | 13909031 | R2S5U9H76YLWKT | 0736062789 |
| 6 | US | 9911218 | RM0CW1UGZTCZK | 0312538367 |
| 7 | US | 17033727 | RXEZMN6LBLSBU | 1581060629 |
| 8 | US | 45799330 | RN3F4XOK39ZS0 | 0240812581 |
| 9 | US | 51455130 | R26RXQ0QNNI9CN | 1563921197 |
| 10 | US | 47323097 | R1ZTLC3RWK0PP | 0135026961 |

Running an analytic query

- Run the following script to run an analytic query with aggregation to simulate real-world use cases:

```
SELECT product_category, count(*) as TotalReviews, avg(star_rating) as AverageRating
FROM lakeformation_tutorial_amazon_reviews.amazon_reviews_governed
GROUP BY product_category
```

This query returns the total number of reviews and average rating per product category.

Running an analytic query with time travel

Governed tables enable time travel – you can query a table as of a previous time.

Note

In order to run time travel queries in Athena, you need to use Athena engine version 2. If your workgroup still uses Athena engine version 1, update your workgroup to use Athena engine version 2.

- To submit a time travel query, use FOR SYSTEM_TIME AS OF timestamp after the table name in the SELECT statement, as in the following example syntax:

```
SELECT *
FROM database.table
FOR SYSTEM_TIME AS OF timestamp
```

The timestamp parameter is either a timestamp or timestamp with a time zone. If not specified, Athena considers the value to be a timestamp in UTC time. Run the time travel query to retrieve data as of 2021-05-26 08:15:00 UTC:

```
SELECT product_category, count(*) as TotalReviews, avg(star_rating) as AverageRating
FROM lakeformation_tutorial_amazon_reviews.amazon_reviews_governed
FOR SYSTEM_TIME AS OF TIMESTAMP '2021-05-26 08:15:00 UTC'
GROUP BY product_category
```

The **Results** screen includes the record of product_category=Camera. This is because the file under product_category=Books was added after the timestamp (**2021-05-26 08:15:00 UTC**), which has been specified in FOR SYSTEM_TIME AS OF.

Step 6: Clean up AWS resources

Clean up resources

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

- [Delete the cloud formation stack](#). The governed table you created is automatically deleted with the stack.

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

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

LF-TBAC is an authorization strategy that defines permissions based on attributes. In Lake Formation, these attributes are called LF-tags. You can attach LF-tags to Data Catalog resources and Lake Formation principals. Data lake administrators can assign and revoke permissions on Lake Formation resources using LF-tags. For more information about see, [Lake Formation Tag-based access control \(p. 231\)](#).

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

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

- You have a large number of tables and principals that the data lake administrator has to grant access
- You want to classify your data based on an ontology and grant permissions based on classification
- The data lake administrator wants to assign permissions dynamically, in a loosely coupled way

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

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

Topics

- [Intended audience \(p. 66\)](#)
- [Prerequisites \(p. 66\)](#)
- [Step 1: Provision your resources \(p. 66\)](#)
- [Step 2: Register your data location, create an LF-tag ontology, and grant permissions \(p. 67\)](#)
- [Step 3: Create Lake Formation databases \(p. 69\)](#)
- [Step 4: Grant table permissions \(p. 77\)](#)
- [Step 5: Run a query in Amazon Athena to verify the permissions \(p. 78\)](#)
- [Step 6: Clean up AWS resources \(p. 79\)](#)

Intended audience

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

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

| Role | Description |
|------------------------------|--|
| Data steward (administrator) | The <code>lf-data-steward</code> user has the following access: <ul style="list-style-type: none">• Read access to all resources in the Data Catalog• Can create LF-tags and associate to the data engineer role for grantable permission to other principals |
| Data engineer | <code>lf-data-engineer</code> user has the following access: <ul style="list-style-type: none">• Full read, write, and update access to all resources in the Data Catalog• Data location permissions in the data lake• Can associate LF-tags and associate to the Data Catalog• Can attach LF-tags to resources, which provides access to principals based on any policies created by data stewards |
| Data analyst | The <code>lf-data-analyst</code> user has the following access: <ul style="list-style-type: none">• Fine-grained access to resources shared by Lake Formation tag-based access policies |

Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in to as an IAM user with correct permissions. For more information, see [Sign up for AWS \(p. 10\)](#) and [Create an Administrator IAM User \(p. 10\)](#).

The tutorial assumes that you are familiar with IAM. For information about IAM, see the [IAM User Guide](#).

Step 1: Provision your resources

This tutorial includes an AWS CloudFormation template for a quick setup. You can review and customize it to suit your needs. The template creates three different roles (listed in [Intended audience \(p. 66\)](#)) to perform this exercise and copies the nyc-taxi-data dataset to your local Amazon S3 bucket.

- An Amazon S3 bucket
- The appropriate Lake Formation settings

- The appropriate Amazon EC2 resources
- Three IAM user roles with user ID credentials

Create your resources

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

The stack creation can take up to five minutes.

Note

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

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

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

Register a data location and define LF-tag ontology

1. Perform the first step as the data steward user (`lf-data-steward`) to verify the data in Amazon S3 and the Data Catalog in Lake Formation.
 - a. Sign in to the Lake Formation console at <https://console.aws.amazon.com/lakeformation/> as `lf-data-steward` with the password used while deploying the AWS CloudFormation stack.
 - b. In the navigation pane, under **Permissions**, choose **Administrative roles and tasks**.
 - c. For **IAM users and roles**, choose the user `lf-data-steward`.
 - d. Choose **Save** to add `lf-data-steward` as a Lake Formation administrator.
2. Next, update the Data Catalog settings to use Lake Formation permission to control catalog resources instead of IAM based access control.
 - a. In the navigation pane, under **Data Catalog**, choose **Settings**.
 - b. Uncheck **Use only IAM access control for new databases**.
 - c. Uncheck **Use only IAM access control for new tables in new databases**.
 - d. Click **Save**.
3. Next, register the data location for the data lake.
 - a. In the navigation pane, under **Register and ingest**, choose **Data lake locations**.
 - b. For **Amazon S3 path**, enter `s3://lf-tagbased-demo-Account-ID`.
 - c. For **IAM role**, leave the default value `AWSServiceRoleForLakeFormationDataAccess` as it is.

- d. Choose **Register location**.
4. Next, create the ontology by defining an LF-tag.
 - a. Under **Permissions** in the navigation pane, under **Administrative roles**, choose **LF-tags..**
 - b. Choose **Add LF-tags**.
 - c. For **Key**, enter Confidential.
 - d. For **Values**, add True and False.
 - e. Choose **Add LF-tag**.
 - f. Repeat the steps to create the **LF-tag Sensitive** with the value True.

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

Grant permissions to IAM users

1. Next, give specific IAM principals the ability to attach newly created LF-tags to resources.
 - a. Under **Permissions** in the navigation pane, under **Administrative roles**, choose **LF-tag permissions**.
 - b. Choose **Grant**.
 - c. Select **IAM users and roles**.
 - d. For **IAM users and roles**, search for and choose the lf-data-engineer role.
 - e. In the **LF-tag permission scope** section, add the key Confidential with values True and False, and the key Sensitive with value True.
 - f. Under **Permissions**, select **Describe** and **Associate** for **LF-tag permissions** and **Grantable permissions**.
 - g. Choose **Grant**.
2. Next, grant permissions to lf-data-engineer to create databases in our Data Catalog and on the underlying Amazon S3 bucket created by AWS CloudFormation.
 - a. Under **Permissions** in the navigation pane, choose **Administrative roles**.
 - b. In the **Database creators** section, choose **Grant**.
 - c. For **IAM users and roles**, choose the lf-data-engineer role.
 - d. For **Catalog permissions**, select **Create database**.
 - e. Choose **Grant**.
3. Next, grant permissions on the Amazon S3 bucket (`s3://lf-tagbased-demo-Account-ID`) to the lf-data-engineer user.
 - a. In the navigation pane, choose **Data locations**.
 - b. Choose **Grant**.
 - c. Select **My account**.
 - d. For **IAM users and roles**, choose the lf-data-engineer role.
 - e. For **Storage locations**, enter the Amazon S3 bucket created by the AWS CloudFormation template (`s3://lf-tagbased-demo-Account-ID`).
 - f. Choose **Grant**.
4. Next, grant lf-data-engineer grantable permissions on resources associated with the **LF-tag** expression Confidential=True.
 - a. In the navigation pane, choose **Data permissions**.
 - b. Choose **Grant**.
 - c. Select **IAM users and roles**.

- d. Choose the role lf-data-engineer.
 - e. In the **LF-tag or catalog resources** section, select **Resources matched by LF-tags**.
 - f. Choose **Add LF-tag**.
 - g. Add the key Confidential with the values True.
 - h. In the **Database permissions** section, select **Describe** for **Database permissions** and **Grantable permissions**.
 - i. In the **Table and column permissions** section, select **Describe**, **Select**, and **Alter** for both **Table permissions** and **Grantable permissions**.
 - j. Choose **Grant**.
5. Next, grant lf-data-engineer grantable permissions on resources associated with the LF-tag expression Confidential=False.
- a. In the navigation pane, choose **Data permissions**.
 - b. Choose **Grant**.
 - c. Select **IAM users and roles**.
 - d. Choose the role lf-data-engineer.
 - e. Select **Resources matched by LF-tags**.
 - f. Choose **Add LF-tag**.
 - g. Add the key Confidential with the value False.
 - h. In the **Database permissions** section, select **Describe** for **Database permissions** and **Grantable permissions**.
 - i. In the **Table and column permissions** section, do not select anything.
 - j. Choose **Grant**.
6. Next, we grant lf-data-engineer grantable permissions on resources associated with the **LF-tag** expression Confidential=False and Sensitive=True.
- a. In the navigation pane, choose **Data permissions**.
 - b. Choose **Grant**.
 - c. Select **IAM users and roles**.
 - d. Choose the role lf-data-engineer.
 - e. Select **Resources matched by LF-tags**.
 - f. Choose **Add LF-tag**.
 - g. Add the key Confidential with the value False.
 - h. Choose **Add LF-tag**.
 - i. Add the key Sensitive with the value True.
 - j. In the **Database permissions** section, select **Describe** for **Database permissions** and **Grantable permissions**.
 - k. In the **Table and column permissions** section, select **Describe**, **Select**, and **Alter** for both **Table permissions** and **Grantable permissions**.
 - l. Choose **Grant**.

Step 3: Create Lake Formation databases

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

Create your databases and table for database-level access

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

- a. On the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>), choose **Databases**.
 - b. Choose **Create database**.
 - c. For **Name**, enter tag_database.
 - d. For **Location**, enter the Amazon S3 location created by the AWS CloudFormation template (s3://lf-tagbased-demo-*Account-ID*/tag_database/).
 - e. Deselect **Use only IAM access control for new tables in this database**.
 - f. Choose **Create database**.
2. Next, create a new table within tag_database.
 - a. On the **Databases** page, select the database tag_database.
 - b. Choose **View Tables** and click **Create table**.
 - c. For **Name**, enter source_data.
 - d. For **Database**, choose the database tag_database.
 - e. For **Data is located in**, select **Specified path in my account**.
 - f. For **Include path**, enter the path to tag_database created by the AWS CloudFormation template (s3://lf-tagbased-demo*Account-ID*/tag_database/).
 - g. For **Data format**, select **CSV**.
 - h. Under **Upload schema**, enter the following JSON array of column structure to create a schema:

```
[  
  {  
    "Name": "vendorid",  
    "Type": "string"  
  },  
  {  
    "Name": "lpep_pickup_datetime",  
    "Type": "string"  
  },  
  {  
    "Name": "lpep_dropoff_datetime",  
    "Type": "string"  
  },  
  {  
    "Name": "store_and_fwd_flag",  
    "Type": "string"  
  },  
  {  
    "Name": "ratecodeid",  
    "Type": "string"  
  },  
  {  
    "Name": "pulocationid",  
    "Type": "string"  
  },  
  {  
    "Name": "dolocationid",  
    "Type": "string"  
  },  
  {  
    "Name": "passenger_count",  
    "Type": "string"  
  }]
```

```
        },
        {
            "Name": "trip_distance",
            "Type": "string"
        },
        {
            "Name": "fare_amount",
            "Type": "string"
        },
        {
            "Name": "extra",
            "Type": "string"
        },
        {
            "Name": "mta_tax",
            "Type": "string"
        },
        {
            "Name": "tip_amount",
            "Type": "string"
        },
        {
            "Name": "tolls_amount",
            "Type": "string"
        },
        {
            "Name": "ehail_fee",
            "Type": "string"
        },
        {
            "Name": "improvement_surcharge",
            "Type": "string"
        },
        {
            "Name": "total_amount",
            "Type": "string"
        },
        {
            "Name": "payment_type",
            "Type": "string"
        }
    ]
}
```

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

| # | Column Name | Data type |
|----|----------------------------|-----------|
| 1 | vendorid | string |
| 2 | lpep_pickup_datetime | string |
| 3 | lpep_dropoff_datetime | string |
| 4 | store_and_fwd_flag | string |
| 5 | ratecodeid | string |
| 6 | pulocationid | string |
| 7 | dolocationid | string |
| 8 | passenger_count | string |
| 9 | trip_distance | string |
| 10 | fare_amount | string |
| 11 | extra | string |
| 12 | mta_tax | string |
| 13 | tip_amount | string |
| 14 | tolls_amount | string |
| 15 | ehail_fee | string |
| 16 | improvement_surcharge | string |
| 17 | total_amount | string |
| 18 | payment_type ⁷² | string |

- j. Choose **Submit**.
3. Next, attach LF-tags at the database level.
 - a. On the **Databases** page, find and select tag_database.
 - b. On the **Actions** menu, choose **Edit LF-tags**.
 - c. Choose **Assign new LF-tag**.
 - d. For **Assigned keys**, choose the Confidential LF-tag you created earlier.
 - e. For **Values**, choose True.
 - f. Choose **Save**.

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

Create your database and table for column-level access

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

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

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

```
{  
    "Name": "store_and_fwd_flag",  
    "Type": "string"  
  
},  
{  
    "Name": "ratecodeid",  
    "Type": "string"  
  
},  
{  
    "Name": "pulocationid",  
    "Type": "string"  
  
},  
{  
    "Name": "dolocationid",  
    "Type": "string"  
  
},  
{  
    "Name": "passenger_count",  
    "Type": "string"  
  
},  
{  
    "Name": "trip_distance",  
    "Type": "string"  
  
},  
{  
    "Name": "fare_amount",  
    "Type": "string"  
  
},  
{  
    "Name": "extra",  
    "Type": "string"  
  
},  
{  
    "Name": "mta_tax",  
    "Type": "string"  
  
},  
{  
    "Name": "tip_amount",  
    "Type": "string"  
  
},  
{  
    "Name": "tolls_amount",  
    "Type": "string"  
  
},  
{
```

```
{  
    "Name": "ehail_fee",  
    "Type": "string"  
  
},  
{  
    "Name": "improvement_surcharge",  
    "Type": "string"  
  
},  
{  
    "Name": "total_amount",  
    "Type": "string"  
  
},  
{  
    "Name": "payment_type",  
    "Type": "string"  
  
}  
]
```

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

| # | Column Name | Data type |
|----|----------------------------|-----------|
| 1 | vendorid | string |
| 2 | lpep_pickup_datetime | string |
| 3 | lpep_dropoff_datetime | string |
| 4 | store_and_fwd_flag | string |
| 5 | ratecodeid | string |
| 6 | pulocationid | string |
| 7 | dolocationid | string |
| 8 | passenger_count | string |
| 9 | trip_distance | string |
| 10 | fare_amount | string |
| 11 | extra | string |
| 12 | mta_tax | string |
| 13 | tip_amount | string |
| 14 | tolls_amount | string |
| 15 | ehail_fee | string |
| 16 | improvement_surcharge | string |
| 17 | total_amount | string |
| 18 | payment_type ⁷⁶ | string |

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

Step 4: Grant table permissions

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

1. Follow these steps to grant permissions to the lf-data-analyst user on the objects associated with the LF-tag Confidential=True (Database : tag_database) to have Describe the database and Select permission on tables.
 - a. Sign in to the Lake Formation console at <https://console.aws.amazon.com/lakeformation/> as lf-data-engineer.
 - b. On the **Permissions** page, select **Data Permissions**.
 - c. Choose **Grant**.
 - d. Under **Principals**, select **IAM users and roles**.
 - e. For **IAM users and roles**, choose lf-data-analyst.
 - f. Select **Resources matched by LF-tags**.
 - g. Choose **Add LF-tag**.
 - h. For **Key**, choose Confidential.
 - i. For **Values**, choose True.
 - j. For **Database permissions**, select **Describe**.
 - k. For **Table permissions**, choose **Select** and **Describe**.
 - l. Choose **Grant**.
2. Next, repeat the steps to grant permissions to data analysts for LF-tag expression for Confidential=False. This LF-tag is used for describing the col_tag_database and the table source_data_col_lvl when logged in as lf-data-analyst from Amazon Athena.
 - a. Sign in to the Lake Formation console at <https://console.aws.amazon.com/lakeformation/> as lf-data-engineer.
 - b. On the **Databases** page, select the database col_tag_database.
 - c. Choose **Action** and **Grant**.

- d. Under **Principals**, select **IAM users and roles**.
 - e. For **IAM users and roles**, choose **lf-data-analyst**.
 - f. Select **Resources matched by LF-tags**.
 - g. Choose **Add LF-tag**.
 - h. For **Key**, choose **Confidential**.
 - i. For **Values**, choose **False**.
 - j. For **Database permissions**, select **Describe**.
 - k. For **Table permissions**, do not select anything.
 - l. Choose **Grant**.
3. Next, repeat the steps to grant permissions to data analysts for LF-tag expression for **Confidential=False** and **Sensitive=True**. This LF-tag is used for describing the **col_tag_database** and the table **source_data_col_lvl** (column-level) when logged in as **lf-data-analyst** from Amazon Athena.
 - a. Sign in to the Lake Formation console at <https://console.aws.amazon.com/lakeformation/> as **lf-data-engineer**.
 - b. On the Databases page, select the database **col_tag_database**.
 - c. Choose **Action and Grant**.
 - d. Under **Principals**, select **IAM users and roles**.
 - e. For **IAM users and roles**, choose **lf-data-analyst**.
 - f. Select **Resources matched by LF-tags**.
 - g. Choose **Add LF-tag**.
 - h. For **Key**, choose **Confidential**.
 - i. For **Values**, choose **False**.
 - j. Choose **Add LF-tag**.
 - k. For **Key**, choose **Sensitive**.
 - l. For **Values**, choose **True**.
 - m. For **Database permissions**, select **Describe**.
 - n. For **Table permissions**, select **Select** and **Describe**.
 - o. Choose **Grant**.

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

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

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

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

5. Run another query using **col_tag_database** and **source_data_col_lvl**.

- The second query returns the two columns that were tagged as Non-Confidential and Sensitive.
6. You can also check to see the Lake Formation tag-based access policy behavior on columns to which you do not have policy grants. When an untagged column is selected from the table source_data_col_lvl, Athena returns an error. For example, you can run the following query to choose untagged columns geolocationid:

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

Step 6: Clean up AWS resources

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

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

Securing data lakes with row-level access control

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

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

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

Topics

- [Intended audience \(p. 80\)](#)
- [Prerequisites \(p. 80\)](#)
- [Step 1: Provision your resources \(p. 80\)](#)
- [Step 2: Query without data filters \(p. 81\)](#)
- [Step 3: Set up data filters and grant permissions \(p. 85\)](#)
- [Step 4: Query with data filters \(p. 86\)](#)
- [Step 5: Clean up AWS resources \(p. 88\)](#)

Intended audience

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

| Role | Description |
|-------------------------|--|
| IAM Administrator | A user who can create AWS Identity and Access Management (IAM) users and roles and Amazon Simple Storage Service (Amazon S3) buckets. Has the <code>AdministratorAccess</code> AWS managed policy. |
| Data lake administrator | A user responsible for setting up the data lake, creating data filters, and granting permissions to data analysts. |
| Data analyst | A user who can run queries against the data lake. Data analysts residing in different countries (for our use case, the US and Japan) can only analyze product reviews for customers located in their own country and for compliance reasons, should not be able to see customer data located in other countries. |

Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management (IAM) user with correct permissions. For more information, see [Sign up for AWS \(p. 10\)](#) and [Create an Administrator IAM User \(p. 10\)](#).

The tutorial assumes that you are familiar with IAM. For information about IAM, see the [IAM User Guide](#).

Change Lake Formation settings

Important

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

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

Step 1: Provision your resources

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

- IAM users and policies for:
 - DataLakeAdmin
 - DataAnalystUS

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

Create your resources

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

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

The stack creation can take one minute to complete.

Step 2: Query without data filters

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

Query the table without row-level access control

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

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

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

New query 1 +

```
1 SELECT *
2 FROM lakeformation_tutorial_row_security.amazon_re
3 LIMIT 10
```

Run query Save as Create (Run time: 12.62 seconds, D)
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Results

| | marketplace | customer_id | review_id | product_id |
|----|-------------|-------------|----------------|------------|
| 1 | US | 22066705 | R3HZYXMJ5HEXIG | 630487862 |
| 2 | US | 20838467 | RJC8PH4K3DVQB | 630335663 |
| 3 | US | 15338666 | R1OH4581ARVWNX | 630026943 |
| 4 | US | 7080939 | R3TWQ5OT8KW0E8 | B000EKCC |
| 5 | US | 30548191 | R3BK9ULGX82VG0 | 078311317 |
| 6 | US | 16052189 | R1LV7NN89A38YT | 630286283 |
| 7 | US | 43430756 | R2IJAEL03PXEYM | B00027VB |
| 8 | US | 43539164 | R3TN0J9JANR9Q5 | 630320554 |
| 9 | US | 21187650 | R2AVXCQOLI53IC | 63026067 |
| 10 | US | 76222222 | R2Z1NIPBLURSKA | 63000771 |

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

```
SELECT marketplace, count(*) as total_count
FROM lakeformationTutorial_row_security.amazon_reviews
GROUP BY marketplace
```

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

The screenshot shows the AWS Lake Formation query editor interface. At the top, there is a header bar with a back arrow, a search bar containing 'New query 1', and a plus sign icon. Below the header, the query text is displayed:

```
1 SELECT marketplace, count(*) as total_count
2 FROM lakeformation_tutorial_row_security.amazon_review
3 GROUP BY marketplace
```

Below the query text are three buttons: 'Run query' (blue), 'Save as' (gray), and 'Create' (gray). To the right of the 'Run query' button, the text '(Run time: 12.4 seconds, Dat...' is visible. A note at the bottom says 'Use Ctrl + Enter to run query, Ctrl + Space to autocomplete'.

The bottom section is titled 'Results' and displays a table with the following data:

Results

| | marketplace |
|---|-------------|
| 1 | FR |
| 2 | UK |
| 3 | JP |
| 4 | DE |
| 5 | US |

Step 3: Set up data filters and grant permissions

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

Create data filters and grant permissions

1. Create a filter to restrict access to the US marketplace data.
 - a. Sign in to the Lake Formation console at <https://console.aws.amazon.com/lakeformation/> in US East (N. Virginia) region as the DatalakeAdmin user.
 - b. Choose **Data filters**.
 - c. Choose **Create new filter**.
 - d. For **Data filter name**, enter amazon_reviews_US.
 - e. For **Target database**, choose the database lakeformationTutorial_row_security.
 - f. For **Target table**, choose the table amazon_reviews.
 - g. For **Column-level access**, leave as the default.
 - h. For **Row filter expression**, enter marketplace='US'.
 - i. Choose **Create filter**.
2. Create a filter to restrict access to the Japanese marketplace data.
 - a. On the **Data filters** page, choose **Create new filter**.
 - b. For **Data filter name**, enter amazon_reviews_JP.
 - c. For **Target database**, choose the database lakeformationTutorial_row_security.
 - d. For **Target table**, choose the table amazon_reviews.
 - e. For **Column-level access**, leave as the default.
 - f. For **Row filter expression**, enter marketplace='JP'.
 - g. Choose **Create filter**.
3. Next, grant permissions to the data analysts using these data filters. Follow these steps to grant permissions to the US data analyst (DataAnalystUS):
 - a. Under **Permissions**, choose **Data lake permissions**.
 - b. Under **Data permission**, choose **Grant**.
 - c. For **Principals**, choose **IAM users and roles**, and select the role DataAnalystUS.
 - d. For **LF tags or catalog resources**, choose **Named data catalog resources**.
 - e. For **Database**, choose lakeformationTutorial_row_security.
 - f. For **Tables-optional**, choose amazon_reviews.
 - g. For **Data filters – optional**, select amazon_reviews_US.
 - h. For **Data filter permissions**, select **Select**.
 - i. Choose **Grant**.
4. Follow these steps to grant permissions to the Japanese data analyst (DataAnalystJP):
 - a. Under **Permissions**, choose **Data lake permissions**.
 - b. Under **Data permission**, choose **Grant**.
 - c. For **Principals**, choose **IAM users and roles**, and select the role DataAnalystJP.
 - d. For **LF tags or catalog resources**, choose **Named data catalog resources**.
 - e. For **Database**, choose lakeformationTutorial_row_security.

- f. For **Tables-optional**, choose amazon_reviews.
- g. For **Data filters – optional**, select amazon_reviews_JP.
- h. For **Data filter permissions**, select **Select**.
- i. Choose **Grant**.

Step 4: Query with data filters

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

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

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

The following screenshot shows the query result.

New query 1 New query 2 +

```
1 SELECT *
2 FROM lakeformation_tutorial_row_security.amazon_re
3 LIMIT 10
```

Run query Save as Create (Run time: 11.9 seconds, Da)
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Results

| | marketplace | customer_id | review_id | product_id |
|----|-------------|-------------|----------------|------------|
| 1 | US | 43836277 | R2NUBTTUO60VYU | B00068S4 |
| 2 | US | 20261976 | R2QTOLZUQERU5B | 63030600 |
| 3 | US | 15947067 | R1PHKR75RKZNSU | 63039273 |
| 4 | US | 19288153 | R1BL2WVE5X34UN | 63040321 |
| 5 | US | 19712967 | R2DKOCIBS5FSP7 | 07840177 |
| 6 | US | 51047097 | R2XF5HQATT4IVR | 07939601 |
| 7 | US | 43836277 | R2NUBTTUO60VYU | B00068S4 |
| 8 | US | 51047097 | R1C0H0G6NATZXO | 63048725 |
| 9 | US | 42808630 | R2HXW7UD4IGZLN | 63030600 |
| 10 | US | 11682952 | R18IURLUPYI4DP | 63029937 |

3. Similarly, run a query to count the total number of records per marketplace.

```
SELECT marketplace , count ( * ) as total_count
FROM lakeformationTutorialRowSecurity .amazon_reviews
GROUP BY marketplace
```

The query result only shows the marketplace US in the results. This is because the user is only allowed to see rows where the marketplace column value is equal to US.

4. Switch to the DataAnalystJP user and run the same query.

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

The query result shows only the records belong to the JP marketplace.

5. Run the query to count the total number of records per marketplace.

```
SELECT marketplace, count(*) as total_count
FROM lakeformationTutorialRowSecurity .amazon_reviews
GROUP BY marketplace
```

The query result shows only the row belonging to the JP marketplace.

Step 5: Clean up AWS resources

Clean up resources

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

- [Delete the cloud formation stack](#).

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

This tutorial demonstrates how you can configure AWS Lake Formation to securely share data stored within a data lake with multiple companies, organizations, or business units, without having to copy the entire database. There are two options to share your databases and tables with another AWS account by using Lake Formation cross-account access control:

- **Lake Formation tag-based access control (recommended)**

Lake Formation tag-based access control is an authorization strategy that defines permissions based on attributes. In Lake Formation, these attributes are called *LF-tags*. For more details, refer to [Managing a data lake using Lake Formation tag-based access control \(p. 65\)](#).

- **Lake Formation named resources**

The Lake Formation named resource method is an authorization strategy that defines permissions for resources. Resources include databases, tables, and columns. Data lake administrators can assign and revoke permissions on Lake Formation resources. For more details, refer to [Cross-account data sharing in Lake Formation \(p. 199\)](#).

We recommend using named resources if the data lake administrator prefers granting permissions explicitly to individual resources. When you use the named resource method to grant Lake Formation permissions on a Data Catalog resource to an external account, Lake Formation uses AWS Resource Access Manager (AWS RAM) to share the resource.

Topics

- [Intended audience \(p. 89\)](#)
- [Configure Lake Formation Data Catalog settings in the producer account \(p. 90\)](#)
- [Step 1: Provision your resources using AWS CloudFormation templates \(p. 92\)](#)
- [Step 2: Lake Formation cross-account sharing prerequisites \(p. 93\)](#)
- [Step 3: Implement cross-account sharing using the tag-based access control method \(p. 95\)](#)
- [Step 4: Implement the named resource method \(p. 99\)](#)
- [Step 5: Clean up AWS resources \(p. 101\)](#)

Intended audience

This tutorial is intended for data stewards, data engineers, and data analysts. When it comes to sharing Data Catalog tables from AWS Glue and administering permission in Lake Formation, data stewards within the producing accounts have functional ownership based on the functions they support, and can grant access to various consumers, external organizations, and accounts. The following table lists the roles that are used in this tutorial:

| Role | Description |
|-----------------------|---|
| DataLakeAdminProducer | <p>The data lake admin IAM user has the following access:</p> <ul style="list-style-type: none">• Full read, write, and update access to all resources in the Data Catalog• Ability to grant permissions to resources• Can create resource links for the shared table• Can attach LF-tags to resources, which provides access to principals based on any policies created by data stewards |
| DataLakeAdminConsumer | <p>The data lake admin IAM user has the following access:</p> <ul style="list-style-type: none">• Full read, write, and update access to all resources in the Data Catalog• Ability to grant permissions to resources• Can create resource links for the shared table• Can attach LF-tags to resources, which provides access to principals based on any policies created by data stewards |
| DataAnalyst | <p>The DataAnalyst user has the following access:</p> <ul style="list-style-type: none">• Fine-grained access to resources shared by Lake Formation tag-based access policies or using named resources method |

Configure Lake Formation Data Catalog settings in the producer account

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management (IAM) user with correct permissions. For more information, see [Sign up for AWS \(p. 10\)](#) and [Create an Administrator IAM User \(p. 10\)](#).

The tutorial assumes that you are familiar with IAM. For information about IAM, see the [IAM User Guide](#).

Configure Lake Formation Data Catalog settings in the producer account

Note

In this tutorial, the account that has the source table is called the producer account, and the account that needs access to the source table is called a consumer account.

Lake Formation provides its own permission management model. To maintain backward compatibility with the IAM permission model, the Super permission is granted to the group IAMAllowedPrincipals on all existing AWS Glue Data Catalog resources by default. Also, **Use only IAM access control settings** are enabled for new Data Catalog resources. This tutorial uses fine grained access control using Lake Formation permissions and use IAM policies for coarse grained access control. See [Methods for fine-grained access control \(p. 291\)](#) for details. Therefore, before you use an AWS CloudFormation template for a quick setup, you need to change Lake Formation Data Catalog settings in the producer account.

Important

This setting affects all newly created databases and tables, so we strongly recommend completing this tutorial in a non-production account or in a new account. Also, if you are using a shared account (such as your company's development account), make sure it does not affect others resources. If you prefer to keep the default security settings, you must complete an extra step when sharing resources to other accounts, in which you revoke the default **Super** permission from IAMAllowedPrincipals on the database or table. We discuss the details later in this tutorial.

To configure Lake Formation Data Catalog settings in the producer account, complete the following steps:

1. Sign in to the AWS Management Console using the producer account as an admin user, or as a user with Lake Formation PutDataLakeSettings API permission.
2. On the Lake Formation console, in the navigation pane, under **Data Catalog**, choose **Settings**.
3. Deselect **Use only IAM access control for new databases** and **Use only IAM access control for new tables in new databases**

Choose **Save**.

AWS Lake Formation > Data catalog settings

Data catalog settings

Default permissions for newly created tables

These settings maintain existing AWS Glue Data Catalog permissions. These changes will take effect when you revoke the Super permission.



- Use only IAM access control for new data catalog tables
- Use only IAM access control for new tables

Default permissions for AWS CloudWatch Metrics

These settings specify the information being shown in the Metrics tab.

Resource owners

Enter resource owners you wish to share your CloudWatch Metrics with.

Additionally, you can remove CREATE_DATABASE permissions for IAMAllowedPrincipals under **Administrative roles and tasks, Database creators**. Only then, you can govern who can create a new database through Lake Formation permissions.

Step 1: Provision your resources using AWS CloudFormation templates

The CloudFormation template for the producer account generates the following resources:

- An Amazon S3 bucket to serve as the data lake.
- A Lambda function (for Lambda-backed AWS CloudFormation custom resources). We use the function to copy sample data files from the public Amazon S3 bucket to your Amazon S3 bucket.
- IAM users and policies: DataLakeAdminProducer.
- The appropriate Lake Formation settings and permissions including:
 - Defining the Lake Formation data lake administrator in the producer account
 - Registering an Amazon S3 bucket as the Lake Formation data lake location (producer account)
- An AWS Glue Data Catalog database, table, and partition. Since there are two options for sharing resources across AWS accounts, this template creates two separate sets of database and table.

The AWS CloudFormation template for the consumer account generates the following resources:

- IAM users and policies:
 - DataLakeAdminConsumer
 - DataAnalyst
- An AWS Glue Data Catalog database. This database is for creating resource links to shared resources.

Create your resources in the producer account

1. Sign in to the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation> in the US East (N. Virginia) region.
2. Choose **Launch Stack**.
3. Choose **Next**.
4. For **Stack name**, enter a stack name, such as `stack-producer`.
5. In the **User Configuration** section, enter user name and password for `ProducerDatalakeAdminUserName` and `ProducerDatalakeAdminUserPassword`.
6. For **DataLakeBucketName**, enter the name of your data lake bucket. This name needs to be globally unique.
7. For **DatabaseName** and **TableName**, leave the default values.
8. Choose **Next**.
9. On the next page, choose **Next**.
10. Review the details on the final page and select **I acknowledge that AWS CloudFormation might create IAM resources**.
11. Choose **Create**.

The stack creation can take up to one minute.

Create your resources in the consumer account

1. Sign in to the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation> in the US East (N. Virginia) region.
2. Choose [Launch Stack](#).
3. Choose [Next](#).
4. For **Stack name**, enter a stack name, such as `stack-consumer`.
5. In the **User Configuration** section, enter user name and password for `ConsumerDatalakeAdminUserName` and `ConsumerDatalakeAdminUserPassword`.
6. For `DataAnalystUserName` and `DataAnalystUserPassword`, enter the user name and password you want for the data analyst IAM user.
7. For `DataLakeBucketName`, enter the name of your data lake bucket. This name needs to be globally unique.
8. For **DatabaseName**, leave the default values.
9. For `AthenaQueryResultS3BucketName`, enter the name of the Amazon S3 bucket that stores Amazon Athena query results. If you don't have one, [create an Amazon S3 bucket](#).
10. Choose [Next](#).
11. On the next page, choose [Next](#).
12. Review the details on the final page and select **I acknowledge that AWS CloudFormation might create IAM resources**.
13. Choose [Create](#).

The stack creation can take up to one minutes.

Note

After completing the tutorial, delete the stack in AWS CloudFormation to avoid incurring charges. Verify that the resources are successfully deleted in the event status for the stack.

Step 2: Lake Formation cross-account sharing prerequisites

Before sharing resources with Lake Formation, there are prerequisites for both the tag-based access control method and named resource method.

Complete tag-based access control cross-account sharing prerequisites

- Before you can use the tag-based access control method to grant cross-account access to resources, you must add the following JSON permissions object to the Data Catalog resource policy in the producer account. This gives the consumer account permission to access the Data Catalog when `glue:EvaluatedByLakeFormationTags` is true. Also, this condition becomes true for resources on which you granted permission using Lake Formation permission tags to the consumer's account. This policy is required for every AWS account to which you are granting permissions.

The following policy must be within a Statement element. We discuss the full IAM policy in the next section.

```
{  
    "Effect": "Allow",  
    "Action": [  
        "glue:*"  
    ],  
    "Principal": {  
        "AWS": [  
            "arn:aws:iam::123456789012:root"  
        ]  
    }  
}
```

```

        "consumer-account-id>"  
    ]  
},  
"Resource": [  
    "arn:aws:glue:region:account-id:table/*",  
    "arn:aws:glue:region:account-id:database/*",  
    "arn:aws:glue:region:account-id:catalog"  
],  
"Condition": {  
    "Bool": {  
        "glue:EvaluatedByLakeFormationTags": true  
    }  
}  
}
}

```

Complete named resource method cross-account sharing prerequisites

1. If there is no Data Catalog resource policy in your account, the Lake Formation cross-account grants that you make proceed as usual. However, if a Data Catalog resource policy exists, you must add the following statement to it to permit your cross-account grants to succeed if they're made with the named resource method. If you plan to use only the named resource method, or only the tag-based access control method, you can skip this step. In this tutorial, we evaluate both methods, and we need to add the following policy.

The following policy must be within a Statement element. We discuss the full IAM policy in the next section.

```
{
    "Effect": "Allow",
    "Action": [
        "glue:ShareResource"
    ],
    "Principal": {
        "Service": "ram.amazonaws.com"
    },
    "Resource": [
        "arn:aws:glue:region:account-id:table/*/*",
        "arn:aws:glue:region:account-id:database/*",
        "arn:aws:glue:region:account-id:catalog"
    ]
}
```

2. Next, add the AWS Glue Data Catalog; resource policy using the AWS Command Line Interface (AWS CLI).

If you grant cross-account permissions by using both the tag-based access control method and named resource method, you must set the EnableHybrid argument to 'true' when adding the preceding policies. Because this option is not currently supported on the console, and you must use the glue:PutResourcePolicy API and AWS CLI.

First, create a policy document (such as policy.json) and add the preceding two policies. Replace **consumer-account-id** with the **account ID** of the AWS account receiving the grant, **region** with the Region of the Data Catalog containing the databases and tables that you are granting permissions on, and **account-id** with the producer AWS account ID.

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

```

```
{  
    "Effect": "Allow",  
    "Principal": {  
        "Service": "ram.amazonaws.com"  
    },  
    "Action": "glue:ShareResource",  
    "Resource": [  
        "arn:aws:glue:region:account-id:table/*/*",  
        "arn:aws:glue:region:account-id:database/*/*",  
        "arn:aws:glue:region:account-id:catalog"  
    ]  
},  
{  
    "Effect": "Allow",  
    "Principal": {  
        "AWS": "region:account-id"  
    },  
    "Action": "glue:*",  
    "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"  
        }  
    }  
}  
]  
}
```

Enter the following AWS CLI command. Replace *glue-resource-policy* with the correct values (such as file://policy.json).

```
aws glue put-resource-policy --policy-in-json glue-resource-policy --enable-hybrid TRUE
```

For more information, see [put-resource-policy](#).

Step 3: Implement cross-account sharing using the tag-based access control method

In this section, we walk you through the following high-level steps:

1. Define an LF-tag.
2. Assign the LF-tag to the target resource.
3. Grant LF-tag permissions to the consumer account.
4. Grant data permissions to the consumer account.
5. Optionally, revoke permissions for IAMAllowedPrincipals on the database, tables, and columns.
6. Create a resource link to the shared table.
7. Create an LF-tag and assign it to the target database.
8. Grant LF-tag data permissions to the consumer account.

Define an LF-tag

Note

If you are signed in to your producer account, sign out before completing the following steps.

1. Sign into the producer account as the data lake administrator at <https://console.aws.amazon.com/lakeformation/>. Use the producer account number, IAM user name (the default is DatalakeAdminProducer), and password that you specified during AWS CloudFormation stack creation.
2. On the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>), in the navigation pane, under **Permissions**, and under **Administrative roles and tasks**, choose **LF-tags**.
3. Choose **Add LF-tag**.

Assign the LF-tag to the target resource

Assign the LF-tag to the target resource and grant data permissions to another account

As a data lake administrator, you can attach tags to resources. If you plan to use a separate role, you may have to grant describe and attach permissions to the separate role.

1. In the navigation pane, under **Data Catalog**, select **Databases**.
2. Select the target database (lakeformation_tutorial_cross_account_database_tbac) and on the **Actions** menu, choose **Edit LF-tags**.

For this tutorial, you assign an LF-tag to a database, but you can also assign LF-tags to tables and columns.

3. Choose **Assign new LF-Tag**.
4. Add the key **Confidentiality** and value **public**.
5. Choose **Save**.

Grant LF-tag permission to the consumer account

Still in the producer account, grant permissions to the consumer account to access the LF-tag.

1. In the navigation pane, under **Permissions**, **Administrative roles and tasks**, **LF-tag permissions**, choose **Grant**.
2. For **Principals**, choose **External accounts**.
3. Enter the target **AWS account ID**.

AWS accounts within the same organization appear automatically. Otherwise, you have to manually enter the AWS account ID. As of this writing, Lake Formation tag-based access control does not support granting permission to organizations or organization units.

4. For LF-tags, choose the **key** and **values** of the LF-tag that is being shared with the consumer account (**key Confidentiality** and **value public**).
5. For **Permissions**, select **Describe for LF-tag permissions**.

LF-tag permissions are permissions given to the consumer account. Grantable permissions are permissions that the consumer account can grant to other principals.

6. Choose **Grant**.

At this point, the consumer data lake administrator should be able to find the policy tag being shared via the consumer account Lake Formation console, under **Permissions**, **Administrative roles and tasks**, **LF-tags**.

Grant data permission to the consumer account

We will now provide data access to the consumer account by specifying an LF-Tag expression and granting the consumer account access to any table or database that matches the expression..

1. In the navigation pane, under **Permissions, Data lake permissions**, choose **Grant**.
2. For **Principals**, choose **External accounts**, and enter the target AWS account ID.
3. For **LF-tags or catalog resources**, choose the **key** and **values** of the **LF-tag** that is being shared with the consumer account (**key** Confidentiality and **value** public).
4. For **Permissions**, under **Resources matched by LF-tags (recommended)** choose **Add LF-tag**.
5. Select the **key** and **value** of the tag that is being shared with the consumer account (key Confidentiality and value public).
6. For **Database permissions**, select **Describe** under **Database permissions** to grant access permissions at the database level.
7. The consumer data lake administrator should be able to find the policy tag being shared via the consumer account on the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>, under **Permissions, Administrative roles and tasks, LF-tags**.
8. Select **Describe** under **Grantable permissions** so the consumer account can grant database-level permissions to its users.
9. For **Table and column permissions**, select **Select** and **Describe** under **Table permissions**.
10. Select **Select** and **Describe** under **Grantable permissions**.
11. Choose **Grant**.

Revoke permission for IAMAllowedPrincipals on the database, tables, and columns (Optional).

At the very beginning of this tutorial, you changed the Lake Formation Data Catalog settings. If you skipped that part, this step is required. If you changed your Lake Formation Data Catalog settings, you can skip this step.

In this step, we need to revoke the default **Super** permission from **IAMAllowedPrincipals** on the database or table. See [Step 4: Switch your data stores to the Lake Formation permissions model \(p. 26\)](#) for details.

Before revoking permission for **IAMAllowedPrincipals**, make sure that you granted existing IAM principals with necessary permission through Lake Formation. This includes three steps:

1. Add IAM permission to the target IAM user or role with the Lake Formation `GetDataAccess` action (with IAM policy).
2. Grant the target IAM user or role with Lake Formation data permissions (alter, select, and so on).
3. Then, revoke permissions for **IAMAllowedPrincipals**. Otherwise, after revoking permissions for **IAMAllowedPrincipals**, existing IAM principals may no longer be able to access the target database or Data Catalog.

Revoking **Super** permission for **IAMAllowedPrincipals** is required when you want to apply the Lake Formation permission model (instead of the IAM policy model) to manage user access within a single account or among multiple accounts using the Lake Formation permission model. You do not have to revoke permission of **IAMAllowedPrincipals** for other tables where you want to keep the traditional IAM policy model.

At this point, the consumer account data lake administrator should be able to find the database and table being shared via the consumer account on the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>, under **Data Catalog, databases**. If not, confirm if the following are properly configured:

1. The correct policy tag and values are assigned to the target databases and tables.
2. The correct tag permission and data permission are assigned to the consumer account.
3. Revoke the default super permission from IAMAllowedPrincipals on the database or table.

Create a resource link to the shared table

When a resource is shared between accounts, and the shared resources are not put in the consumer accounts' Data Catalog. To make them available, and query the underlying data of a shared table using services like Athena, we need to create a resource link to the shared table. A resource link is a Data Catalog object that is a link to a local or shared database or table. For details, see [Creating resource links \(p. 157\)](#). By creating a resource link, you can:

- Assign a different name to a database or table that aligns with your Data Catalog resource naming policies.
- Use services such as Athena and Redshift Spectrum to query shared databases or tables.

To create a resource link, complete the following steps:

1. If you are signed into your consumer account, sign out.
2. Sign in as the consumer account data lake administrator. Use the consumer account ID, IAM user name (default DatalakeAdminConsumer) and password that you specified during AWS CloudFormation stack creation.
3. On the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>), in the navigation pane, under **Data Catalog**, **Databases**, choose the shared database lakeformationTutorial_cross_account_database_tbac.

If you don't see the database, revisit the previous steps to see if everything is properly configured.
4. Choose **View Tables**.
5. Choose the shared table amazon_reviews_table_tbac.
6. On the **Actions** menu, choose **Create resource link**.
7. For **Resource link name**, enter a name (for this tutorial, amazon_reviews_table_tbac_resource_link).
8. Under **Database**, select the database that the resource link is created in (for this post, the AWS CloudFormationn stack created the database lakeformationTutorial_cross_account_database_consumer).
9. Choose **Create**.

The resource link appears under **Data catalog**, **Tables**.

Create an LF-tag and assign it to the target database

Lake Formation tags reside in the same Data Catalog as the resources. This means that tags created in the producer account are not available to use when granting access to the resource links in the consumer account. You need to create a separate set of LF-tags in the consumer account to use LF tag-based access control when sharing the resource links in the consumer account.

1. Define the LF-tag in the consumer account. For this tutorial, we use key Division and values sales, marketing, and analyst.
2. Assign the LF-tag key Division and value analyst to the database lakeformationTutorial_cross_account_database_consumer, where the resource link is created.

Grant LF-tag data permission to the consumer

As a final step, grant LF-tag data permission to the consumer.

1. In the navigation pane, under **Permissions, Data lake permissions**, choose **Grant**.
2. For **Principals**, choose **IAM users and roles**, and choose the user **DataAnalyst**.
3. For **LF-tags or catalog resources**, choose **Resources matched by LF-tags** (recommended).
4. Choose **key** Division and **value** analyst.
5. For **Database permissions**, select **Describe** under **Database permissions**.
6. For **Table and column permissions**, select **Select** and **Describe** under **Table permissions**.
7. Choose **Grant**.
8. Repeat these steps for user **DataAnalyst**, where the LF-tag key is Confidentiality and value is public.

At this point, the data analyst user in the consumer account should be able to find the database and resource link, and query the shared table via the Athena console at <https://console.aws.amazon.com/athena/>. If not, confirm if the following are properly configured:

- The resource link is created for the shared table
- You granted the user access to the LF-tag shared by the producer account
- You granted the user access to the LF-tag associated to the resource link and database that the resource link is created in
- Check if you assigned the correct LF-tag to the resource link, and to the database that the resource link is created in

Step 4: Implement the named resource method

To use the named resource method, we walk you through the following high-level steps:

1. Optionally, revoke permission for **IAMAllowedPrincipals** on the database, tables, and columns.
2. Grant data permission to the consumer account.
3. Accept a resource share from AWS Resource Access Manager.
4. Create a resource link for the shared table.
5. Grant data permission for the shared table to the consumer.
6. Grant data permission for the resource link to the consumer.

Revoke permission for **IAMAllowedPrincipals** on the database, tables, and columns (Optional)

- At the very beginning of this tutorial, we changed Lake Formation Data Catalog settings. If you skipped that part, this step is required. For instructions, see the optional step in the previous section.

Grant data permission to the consumer account

1. **Note**
If you're signed in to producer account as another user, sign out first.

Sign in to the Lake Formation console at <https://console.aws.amazon.com/lakeformation/> using the producer account data lake administrator using the AWS account ID, IAM user name (default is **DatalakeAdminProducer**), and password specified during AWS CloudFormation stack creation.

2. On the **Permissions** page, under **Data lake Permissions** choose **Grant**.

3. Under **Principals**, choose **External accounts**, and enter one or more AWS account IDs or AWS organizations IDs. For more information see: [AWS Organizations](#).

Organizations that the producer account belongs to and AWS accounts within the same organization appear automatically. Otherwise, manually enter the account ID or organization ID.
4. For **LF-tags or catalog resources**, choose **Named data catalog resources**.
5. Under **Databases**, choose the database `lakeformationTutorial_cross_account_database_named_resource`.
6. Choose **Add LF-tag**.
7. Under **Tables**, choose **All tables**.
8. For **Table column permissions**, choose **Select**, and **Describe** under **Table permissions**.
9. Select **Select** and **Describe**, under **Grantable Permissions**.
10. Optionally, for **Data permissions**, choose **Simple column-based access** if column-level permission management is required.
11. Choose **Grant**.

If you have not revoked permission for `IAMAllowedPrincipals`, you get a **Grant permissions failed** error. At this point, you should see the target table being shared via AWS RAM with the consumer account under **Permissions, Data permissions**.

Accept a resource share from AWS RAM

Note

This step is required only for AWS account-based sharing, not for organization-based sharing.

1. Sign in to the AWS console at <https://console.aws.amazon.com/connect/> using the consumer account data lake administrator using the IAM user name (default is `DatalakeAdminConsumer`) and password specified during AWS CloudFormation stack creation.
2. On the AWS RAM console, in the navigation pane, under **Shared with me, Resource shares**, choose the shared Lake Formation resource. The **Status** should be **Pending**.
3. Choose **Action** and **Grant**.
4. Confirm the resource details, and choose **Accept resource share**.

At this point, the consumer account data lake administrator should be able to find the shared resource on the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>) under **Data Catalog, Databases**.

Create a resource link for the shared table

- Follow the instructions in [Step 3: Implement cross-account sharing using the tag-based access control method \(p. 95\)](#) (step 6) to create a resource link for a shared table. Name the resource link `amazon_reviews_table_named_resource_resource_link`. Create the resource link in the database `lakeformationTutorial_cross_account_database_consumer`.

Grant data permission for the shared table to the consumer

To grant data permission for the shared table to the consumer, complete the following steps:

1. On the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>), under **Permissions, Data lake permissions**, choose **Grant**.
2. For **Principals**, choose **IAM users and roles**, and choose the user `DataAnalyst`.
3. For **LF-tags or catalog resources**, choose **Named data catalog resources**.

4. Under **Databases**, choose the database `lakeformation_tutorial_cross_account_database_named_resource`. If you don't see the database on the drop-down list, choose **Load more**.
5. Under **Tables**, choose the table `amazon_reviews_table_named_resource`.
6. For **Table and column permissions**, select **Select** and **Describe** under **Table permissions**.
7. Choose **Grant**.

Grant data permission for the resource link to the consumer

In addition to granting the data lake user permission to access the shared table, you also need to grant the data lake user permission to access the resource link.

1. On the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>), under **Permissions, Data lake permissions**, choose **Grant**.
2. For **Principals**, choose **IAM users and roles**, and choose the user **DataAnalyst**.
3. For **LF-tags or catalog resources**, choose **Named data catalog resources**.
4. Under **Databases**, choose the database `lakeformation_tutorial_cross_account_database_consumer`. If you don't see the database on the drop-down list, choose **Load more**.
5. Under **Tables**, choose the table `amazon_reviews_table_named_resource_resource_link`.
6. For **Resource link permissions**, select **Describe** under **Resource link permissions**.
7. Choose **Grant**.

At this point, the data analyst user in the consumer account should be able to find the database and resource link, and query the shared table via the Athena console.

If not, confirm if the following are properly configured:

- The resource link is created for the shared table
- You granted the user access to the table shared by the producer account
- You granted the user access to the resource link and database for which the resource link is created

Step 5: Clean up AWS resources

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

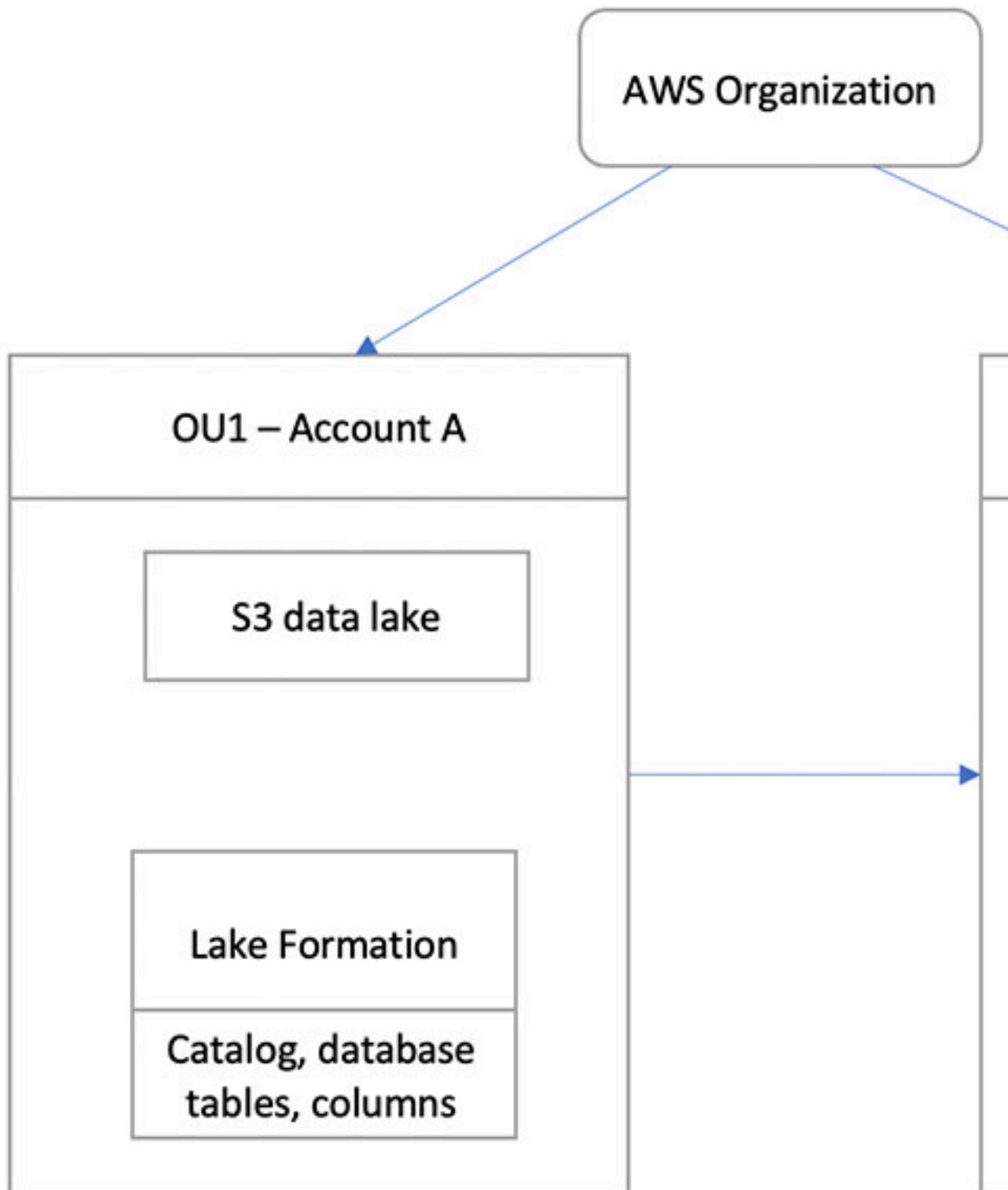
1. Sign in to Lake Formation console at <https://console.aws.amazon.com/lakeformation/> using the producer account and delete or change the following:
 - AWS Resource Access Manager resource share
 - Lake Formation tags
 - AWS CloudFormation stack
 - Lake Formation settings
 - AWS Glue Data Catalog
2. Sign in to Lake Formation console at <https://console.aws.amazon.com/lakeformation/> using the consumer account and delete or change the following:
 - Lake Formation tags
 - AWS CloudFormation stack

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

This tutorial provides step-by-step instructions on how you can quickly and easily share datasets using Lake Formation when managing multiple AWS accounts with AWS Organizations. You define granular permissions to control access to sensitive data.

The following procedures also show how a data lake administrator of Account A can provide fine-grained access for Account B, and how a user in Account B, acting as a data steward, can grant fine-grained access to the shared table for other users in their account. Data stewards within each account can independently delegate access to their own users, giving each team or lines of business (LOB) autonomy.

The use case assumes you are using AWS Organizations to manage your AWS accounts. The user of Account A in one organizational unit (OU1) grants access to users of Account B in OU2. You can use the same approach when not using Organizations, such as when you only have a few accounts. The following diagram illustrates the fine-grained access control of datasets in a data lake. The data lake is available in the Account A. The data lake administrator of Account A provides fine-grained access for Account B. The diagram also shows that a user of Account B provides column-level access of the Account A data lake table to another user in Account B.



- [Intended audience \(p. 104\)](#)
- [Prerequisites \(p. 104\)](#)
- [Step 1: Provide fine-grained access to another account \(p. 105\)](#)
- [Step 2: Provide fine-grained access to a user in the same account \(p. 106\)](#)

Intended audience

This tutorial is intended for data stewards, data engineers, and data analysts. The following table lists the roles that are used in this tutorial:

| Role | Description |
|-------------------------|---|
| IAM administrator | User who can create IAM users and roles and Amazon S3 buckets. Has the <code>AdministratorAccess</code> AWS managed policy. |
| Data lake administrator | User who has the AWS managed policies <code>AWSLakeFormationDataAdmin</code> attached. |
| Data analyst | User who has the AWS managed policy <code>AmazonAthenaFullAccess</code> attached. |

Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management (IAM) user with correct permissions. For more information, see [Sign up for AWS \(p. 10\)](#) and [Create an Administrator IAM User \(p. 10\)](#).

The tutorial assumes that you are familiar with IAM. For information about IAM, see the [IAM User Guide](#).

You need the following resources for this tutorial:

- Two organizational units:
 - OU1 – Contains Account A
 - OU2 – Contains Account B
- An Amazon S3 data lake location (bucket) in Account A.
- A data lake administrator user in Account A. You can create a data lake administrator using the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>) or the `PutDataLakeSettings` operation of the Lake Formation API.
- Lake Formation configured in Account A, and the Amazon S3 data lake location registered with Lake Formation in Account A.
- Two users in Account B with the following IAM managed policies:
 - testuser1 – has the AWS managed policies `AWSLakeFormationDataAdmin` attached.
 - testuser2 – Has the AWS managed policy `AmazonAthenaFullAccess` attached.
- A database `testdb` in the Lake Formation database for Account B.

Step 1: Provide fine-grained access to another account

Learn how a data lake administrator of Account A provides fine-grained access for Account B.

Grant fine-grained access to another account

1. Sign into AWS Management Console at <https://console.aws.amazon.com/connect/> in Account A as a data lake administrator.
2. Open the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>), and choose **Get started**.
3. in the navigation pane, choose **Databases**.
4. Choose **Create database**.
5. In the **Database** details section, select **Database**.
6. For **Name**, enter a name (for this tutorial, we use `sampled01`).
7. Make sure that **Use only IAM access control for new tables in this database** is not selected. Leaving this unselected allows us to control access from Lake Formation.
8. Choose **Create database**.
9. On the **Databases** page, choose your database `sampled01`.
10. On the **Actions** menu, choose **Grant**.
11. In the **Grant permissions** section, select **External account**.
12. For AWS account ID or AWS organization ID, enter the account ID for Account B in OU2.
13. For **Table**, choose the table you want Account B to have access to (for this post, we use table `acc_a_area`). Optionally, you can grant access to columns within the table, which we do in this post.
14. For **Include columns**, choose the columns you want Account B to have access to (for this post, we grant permissions to type, name, and identifiers).
15. For **Columns**, choose **Include columns**.
16. For **Table permissions**, select **Select**.
17. For **Grantable permissions**, select **Select**. Grantable permissions are required so admin users in Account B can grant permissions to other users in Account B.
18. Choose **Grant**.
19. In the navigation pane, choose **Tables**.
20. You could see one active connection in the AWS accounts and AWS organizations with access section.

Create a resource link

Integrated services like Amazon Athena can not directly access databases or tables across accounts. Hence, you need to create a resource link so that Athena can access resource links in your account to databases and tables in other accounts. Create a resource link to the table (`acc_a_area`) so Account B users can query its data with Athena.

1. Sign in to the AWS console at <https://console.aws.amazon.com/connect/> in Account B as `testuser1`.
2. On the Lake Formation console (<https://console.aws.amazon.com/lakeformation/>), in the navigation pane, choose **Tables**. You should see the tables that Account A has provided access.
3. Choose the table `acc_a_area`.
4. On the **Actions** menu, choose **Create resource link**.

5. For **Resource link name**, enter a name (for this tutorial, acc_a_area_rl).
6. For **Database**, choose your database (testdb).
7. Choose **Create**.
8. In the navigation pane, choose **Tables**.
9. Choose the table acc_b_area_rl.
10. On the **Actions** menu, choose **View data**.

You are redirected to the Athena console, where you should see the database and table.

You can now run a query on the table to see the column value for which access was provided to testuser1 from Account B.

Step 2: Provide fine-grained access to a user in the same account

This section shows how a user in Account B (testuser1), acting as a data steward, provides fine-grained access to another user in the same account (testuser2) to the column name in the shared table acc_b_area_rl.

Grant fine-grained access to a user in the same account

1. Sign in to the AWS console at <https://console.aws.amazon.com/connect/> in Account B as testuser1.
2. On the Lake Formation console, in the navigation pane, choose **Tables**.

You can grant permissions on a table through its resource link. To do so, on the **Tables** page, select the resource link acc_b_area_rl, and on the **Actions** menu, choose **Grant on target**.

3. In the **Grant permissions** section, select **My account**.
4. For **IAM users and roles**, choose the user testuser2.
5. For **Column**, choose the column name.
6. For **Table permissions**, select **Select**.
7. Choose **Grant**.

When you create a resource link, only you can view and access it. To permit other users in your account to access the resource link, you need to grant permissions on the resource link itself. You need to grant **DESCRIBE** or **DROP** permissions. On the **Tables** page, select your table again and on the **Actions** menu, choose **Grant**.

8. In the **Grant permissions** section, select **My account**.
9. For **IAM users and roles**, select the user testuser2.
10. For **Resource link permissions**, select **Describe**.
11. Choose **Grant**.
12. Sign in to the AWS console in Account B as testuser2.

On the Athena console (<https://console.aws.amazon.com/athena/>), you should see the database and table acc_b_area_rl. You can now run a query on the table to see the column value that testuser2 has access to.

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 \(p. 294\)](#).

Topics

- Requirements for roles used to register locations ([p. 107](#))
- Registering an Amazon S3 location ([p. 109](#))
- Registering an encrypted Amazon S3 location ([p. 110](#))
- Registering an Amazon S3 location in another AWS account ([p. 113](#))
- Registering an encrypted Amazon S3 location across AWS accounts ([p. 115](#))
- Deregistering an Amazon S3 location ([p. 117](#))

Requirements for roles used to register locations

You must specify an AWS Identity and Access Management (IAM) role when you register an Amazon Simple Storage Service (Amazon S3) location. AWS Lake Formation assumes that role when accessing the data in that location.

You can use one of the following role types to register a location:

- The Lake Formation service-linked role. This role grants the required permissions on the location. Using this role is the simplest way to register the location. For more information, see [Using Service-Linked Roles for Lake Formation \(p. 302\)](#).
- A user-defined role. Use a user-defined role when you need to grant more permissions than the service-linked role provides.

You must use a user-defined role in the following circumstances:

- When registering a location that will be pointed to by a governed table.

For more information, see [Managing governed tables \(p. 120\)](#).

- When registering a location in another account.

For more information, see [the section called “Registering an Amazon S3 location in another AWS account” \(p. 113\)](#) and [the section called “Registering an encrypted Amazon S3 location across AWS accounts” \(p. 115\)](#).

- If you used an AWS managed CMK (aws/s3) to encrypt the Amazon S3 location.

For more information, see [Registering an encrypted Amazon S3 location \(p. 110\)](#).

- If you plan to access the location using Amazon EMR.

If you already registered a location with the service-linked role and want to begin accessing the location with Amazon EMR, you must deregister the location and reregister it with a user-defined role. For more information, see [the section called “Deregistering an Amazon S3 location” \(p. 117\)](#).

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

- When creating the new role, on the [Create role](#) page of the IAM console, choose **AWS service**, and then under **Choose a use case**, choose **Lake Formation**.

If you create the role using a different path, ensure that the role has a trust relationship with `lakeformation.amazonaws.com`. For more information, see [Modifying a Role Trust Policy \(Console\)](#).

- The role must have 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>"
            ]
        }
    ]
}
```

- To permit Lake Formation to add logs in CloudWatch Logs and publish metrics, add the following inline policy.

Note

Writing to CloudWatch Logs incurs a charge.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Sid1",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:<region>:<account-id>:log-group:/aws-lakeformation-
acceleration/*",
                "arn:aws:logs:<region>:<account-id>:log-group:/aws-lakeformation-
acceleration/*:log-stream:/*"
            ]
        }
    ]
}
```

Registering an Amazon S3 location

You must specify an AWS Identity and Access Management (IAM) role when you register an Amazon Simple Storage Service (Amazon S3) location. Lake Formation assumes that role when it grants temporary credentials to integrated AWS services that access the data in that location.

Important

Avoid registering an Amazon S3 bucket that has **Requester pays** enabled. For buckets registered with Lake Formation, the role used to register the bucket is always viewed as the requester. If the bucket is accessed by another AWS account, the bucket owner is charged for data access if the role belongs to the same account as the bucket owner.

You can use the AWS Lake Formation console, Lake Formation API, or AWS Command Line Interface (AWS CLI) to register an Amazon S3 location.

Before You Begin

Review the requirements for the role used to register the location (p. 107).

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 `AWSLambdaRoleForLakeFormationDataAccess` 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" \(p. 107\)](#).
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 \(Python: register_resource\) \(p. 339\)](#).

Note

Once you register an Amazon S3 location, any AWS Glue table pointing to the location (or any of its child locations) will return the value for the `IsRegisteredWithLakeFormation` parameter as `true` in the `GetTable` call. There is a known limitation that Data Catalog API operations such as `GetTables` and `SearchTables` do not update the value for the `IsRegisteredWithLakeFormation` parameter, and return the default, which is `false`. It is recommended to use the `GetTable` API to view the correct value for the `IsRegisteredWithLakeFormation` parameter.

Registering an encrypted Amazon S3 location

Lake Formation integrates with [AWS Key Management Service \(AWS KMS\)](#) to enable you to more easily set up other integrated services to encrypt and decrypt data in Amazon Simple Storage Service (Amazon S3) locations.

Both customer managed AWS KMS keys and AWS managed keys are supported. 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 AWS KMS key, or the KMS key policy must grant permissions on the key to the role.

Important

Avoid registering an Amazon S3 bucket that has **Requester pays** enabled. For buckets registered with Lake Formation, the role used to register the bucket is always viewed as the requester. If the bucket is accessed by another AWS account, the bucket owner is charged for data access if the role belongs to the same account as the bucket owner.

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" \(p. 107\)](#).

Important

If you used an AWS managed key (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 key or an AWS managed key.

- [Registering a location encrypted with a customer managed key \(p. 111\)](#)
- [Registering a location encrypted with an AWS managed key \(p. 112\)](#)

Before You Begin

Review the [requirements for the role used to register the location \(p. 107\)](#).

To register an Amazon S3 location encrypted with a customer managed key

Note

If the KMS key or Amazon S3 location are not in the same AWS account as the Data Catalog, follow the instructions in [the section called "Registering an encrypted Amazon S3 location across AWS accounts" \(p. 115\)](#) instead.

1. Open the AWS KMS console at <https://console.aws.amazon.com/kms> and log in as an AWS Identity and Access Management (IAM) administrative user or as a user who can modify the key policy of the KMS key used to encrypt the location.
2. In the navigation pane, choose **Customer managed keys**, and then choose the name of the desired KMS key.
3. On the KMS key details page, choose the **Key policy** tab, and then do one of the following to add your custom role or the Lake Formation service-linked role as a KMS key user:
 - **If the default view is showing** (with **Key administrators**, **Key deletion**, **Key users**, and **Other AWS accounts** sections) – Under the **Key users** section, add your custom role or the Lake Formation service-linked role `AWSServiceRoleForLakeFormationDataAccess`.
 - **If the key policy (JSON) is showing** – Edit the policy to add your custom role or the Lake Formation service-linked role `AWSServiceRoleForLakeFormationDataAccess` to the object "Allow use of the key," as shown in the following example.

Note

If that object is missing, add it with the permissions shown in the example. The example uses the service-linked role.

...
{

```

        "Sid": "Allow use of the key",
        "Effect": "Allow",
        "Principal": {
            "AWS": [
                "arn:aws:iam::111122223333:role/aws-service-role/
lakeformation.amazonaws.com/AWSServiceRoleForLakeFormationDataAccess",
                "arn:aws:iam::111122223333:user/keyuser"
            ]
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
        ],
        "Resource": "*"
    },
    ...

```

4. Open the AWS Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Sign in as the data lake administrator or as a user with the `lakeformation:RegisterResource` IAM permission.
5. In the navigation pane, under **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” \(p. 107\)](#).
9. Choose **Register location**.

For more information about the service-linked role, see [Service-Linked Role Permissions for Lake Formation \(p. 302\)](#).

To register an Amazon S3 location encrypted with an AWS managed key

Important

If the Amazon S3 location is not in the same AWS account as the Data Catalog, follow the instructions in [the section called “Registering an encrypted Amazon S3 location across AWS accounts” \(p. 115\)](#) 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” \(p. 107\)](#).
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 can't use the Lake Formation service-linked role to register a location in another account. You must use a user-defined role instead. The role must meet the requirements in [the section called "Requirements for roles used to register locations" \(p. 107\)](#). For more information about the service-linked role, see [Service-Linked Role Permissions for Lake Formation \(p. 302\)](#).

Before You Begin

Review the [requirements for the role used to register the location \(p. 107\)](#).

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” \(p. 115\)](#) 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” \(p. 107\)](#). 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 keys and AWS managed keys are supported. Client-side encryption/decryption is not supported.

Important

Avoid registering an Amazon S3 bucket that has **Requester pays** enabled. For buckets registered with Lake Formation, the role used to register the bucket is always viewed as the requester. If the bucket is accessed by another AWS account, the bucket owner is charged for data access if the role belongs to the same account as the bucket owner.

This section explains how to register an Amazon S3 location under the following circumstances:

- The data in the Amazon S3 location is encrypted with a KMS key created in AWS KMS.
- The Amazon S3 location is not in the same AWS account as the AWS Glue Data Catalog.
- The KMS key either is or is not in the same AWS account as the Data Catalog.

Registering an AWS KMS–encrypted Amazon S3 bucket in AWS account B using an AWS Identity and Access Management (IAM) role in AWS account A requires the following permissions:

- The role in account A must grant permissions on the bucket in account B.
- The bucket policy in account B must grant access permissions to the role in Account A.
- If the KMS key is in account B, the key policy must grant access to the role in account A, and the role in account A must grant permissions on the KMS key.

In the following procedure, you create a role in the AWS account that contains the Data Catalog (account A in the previous discussion). Then, you use this role to register the location. Lake Formation assumes this role when accessing underlying data in Amazon S3. The assumed role has the required permissions on the KMS key. As a result, you don't have to grant permissions on the KMS key to principals accessing underlying data with ETL jobs or with integrated services such as Amazon Athena.

Important

You can't use the Lake Formation service-linked role to register a location in another account.

You must use a user-defined role instead. The role must meet the requirements in [the section called "Requirements for roles used to register locations" \(p. 107\)](#). For more information about the service-linked role, see [Service-Linked Role Permissions for Lake Formation \(p. 302\)](#).

Before You Begin

Review the [requirements for the role used to register the location \(p. 107\)](#).

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" \(p. 107\)](#). Ensure that the role includes a policy that grants Amazon S3 permissions on the location.
3. If the KMS key is not in the same account as the Data Catalog, add to the role an inline policy that grants the required permissions on the KMS key. The following is an example policy. Replace `<cmk-region>` and `<cmk-account-id>` with the region and account number of the KMS key. 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 KMS key.

- 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 KMS key used to encrypt the location.
- b. In the navigation pane, choose **Customer managed keys**, and then choose the name of the KMS key.
- c. On the KMS key details page, under the **Key policy** tab, if the JSON view of the key policy is not showing, choose **Switch to policy view**.
- d. In the **Key policy** section, choose **Edit**, and add the Amazon Resource Name (ARN) of the role to the Allow use of the key object, as shown in the following example.

Note

If that object is missing, add it with the permissions shown in the example.

```
...
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::<catalog-account-id>:role/<role-name>"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
...
...
```

For more information, see [Allowing Users in Other Accounts to Use a KMS key](#) in the *AWS Key Management Service Developer Guide*.

6. Open the AWS Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Sign 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 \(p. 119\)](#)
- [Creating Tables \(p. 120\)](#)
- [Managing governed tables \(p. 120\)](#)
- [Searching for Tables \(p. 129\)](#)
- [Sharing Data Catalog tables and databases across AWS Accounts \(p. 130\)](#)
- [Accessing and viewing shared Data Catalog tables and databases \(p. 130\)](#)
- [Creating resource links \(p. 157\)](#)

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 \(p. 295\)](#).

To create a database using the Lake Formation console, you must be signed in as a data lake administrator or *database creator*. A database creator is a principal who has been granted the Lake Formation CREATE_DATABASE permission. You can see a list of database creators on the **Administrative roles and tasks** page of the Lake Formation console. To view this list, you must have the `lakeformation>ListPermissions` IAM permission and be signed in as a data lake administrator or as a database creator with the grant option on the CREATE_DATABASE permission.

To create a database

1. Open the AWS Lake Formation console at <https://console.aws.amazon.com/lakeformation/>, and sign in as a data lake administrator or database creator.
2. In the navigation pane, under **Data catalog**, choose **Databases**.
3. Choose **Create database**.
4. In the **Create database** dialog box, enter a database name, optional location, and optional description.
5. Optionally select **Use only IAM access control for new tables in this database**.

For information about this option, see the section called “[Changing the default security settings for your data lake](#)” (p. 299).

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. 166).
- 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. 157).

Managing governed tables

Governed tables provide several advanced features in AWS Lake Formation, including support for ACID (atomic, consistent, isolated, and durable) transactions, automatic data compaction, and time-travel queries.

Topics

- [Governed tables in Lake Formation](#) (p. 120)
- [Performance optimization of governed tables and row filters](#) (p. 122)
- [Prerequisites for governed tables](#) (p. 122)
- [Creating governed tables](#) (p. 123)
- [Reading and writing governed tables in Lake Formation](#) (p. 124)
- [Storage optimizations for governed tables](#) (p. 125)
- [Notes and restrictions for governed tables](#) (p. 128)

Governed tables in Lake Formation

The metadata tables in the AWS Glue Data Catalog store information about data sources and targets—including schema information, partition information, data location, and more.

The Data Catalog supports two types of metadata tables: governed tables and non-governed tables. Governed tables are unique to AWS Lake Formation. When you create a table, you can specify whether the table is governed.

Governed tables offer the following advanced features:

ACID transactions

ACID (atomic, consistent, isolated, and durable) transactions protect the integrity of Data Catalog operations such as creating or updating a table. They also enable multiple users to concurrently

and reliably add and delete objects in the Amazon S3 data lake, while still allowing other users to simultaneously run analytical queries and machine learning (ML) models on the same datasets that return consistent and up-to-date results. When governed tables are involved in reads from or writes to the data lake on Amazon S3, those operations occur within a transaction.

Transactions protect the integrity of governed table metadata, including the *manifest*—the metadata that defines the Amazon S3 objects in the table's underlying data. Integrated AWS services such as Amazon Athena support governed tables to provide consistent reads in queries. To use transactions in your AWS Glue ETL jobs, you begin a transaction before you perform any reads from or writes to the data lake, and you commit the transaction upon completion.

For more information about transactions, see [Reading from and writing to the data lake within transactions \(p. 276\)](#).

Automatic data compaction

For better performance by ETL jobs and analytics services such as Athena, Lake Formation automatically compacts the small Amazon S3 objects of governed tables into larger objects.

Compaction is enabled for governed tables by default. You can disable compaction for individual governed tables. For more information, see [Storage optimizations for governed tables \(p. 125\)](#).

Time-travel queries

As mentioned previously, each governed table maintains a versioned manifest of the Amazon S3 objects that it comprises. Previous versions of the manifest can be used for time-travel queries. Your queries against governed tables in Athena and in AWS Glue ETL jobs can include a timestamp to indicate that you want to discover the state of the data at a particular date and time.

To submit a time-travel query in Athena, use the syntax `FOR SYSTEM_TIME AS OF timestamp` or `FOR SYSTEM_VERSION AS OF version`.

```
SELECT *
FROM cloudtraildb.cloudtraildata
FOR SYSTEM_TIME AS OF TIMESTAMP '2021-09-30 10:00:00'
```

For more examples of Athena time-traveling queries of governed tables, see [Querying Governed Tables](#) in the *Amazon Athena User Guide*.

In your ETL job script, to read data into a dynamic frame using time travel, include code similar to the following.

Python

```
dynamic_frame = glueContext.create_dynamic_frame_from_catalog(database =
    'cloudtraildb', table_name = 'cloudtraildata',
    additional_options = {"asOfTime": "2021-09-30 10:00:00"})
```

Scala

```
val persons: DynamicFrame = glueContext.getCatalogSource(database = "cloudtraildb",
    tableName = "cloudtraildata",
    additional_options = JsonOptions("""{"asOfTime": "2021-09-30
    10:00:00"}""").getDynamicFrame()
```

Note

Lake Formation permissions are not versioned. Time travel queries always honor current permissions. For example, if permissions at time T1 limited access to table columns and the current permissions (at time T2) grant access to all columns, a time travel query against the data at time T1 returns all columns.

Performance optimization of governed tables and row filters

Governed tables and tables with row and cell-level security, access data using [Lake Formation Storage API](#). The Storage API consistently enforces permissions and returns a consistent view of the data. You should take note of the following considerations.

- The Storage API allows engines to push predicates to Lake Formation to optimize how data is scanned. These predicates can improve the performance of your queries by reducing the amount of data that needs to be read and transferred between Lake Formation and the query engine.
- Unsupported predicates are passed through with no optimization. The results are returned to the query engine, where further optimizations can be performed. The following predicates are not supported:
 - LIKE operator – For example, `SELECT column1 FROM table1 WHERE column2 LIKE '%some_value%'`
 - OR operator on two different columns from the same table – For example, `SELECT column1 FROM table1 WHERE column2 = 'value1' OR column2 = 'value2'`
 - LIMIT – For example, `SELECT column1 FROM table1 LIMIT 100`
- Push down of aggregations is currently not supported.

Prerequisites for governed tables

The following are prerequisites for governed tables:

- Governed tables are supported only for data in Amazon S3. The Amazon S3 location must be registered with Lake Formation. For information about registering a location, see [Adding an Amazon S3 location to your data lake \(p. 107\)](#).
- Governed tables have automatic data compaction enabled by default. For more information about automatic data compaction, see [Automatic data compaction \(p. 121\)](#).
- Principals that call the [transaction API operations \(p. 352\)](#) and [governed table object API operations \(p. 359\)](#) must have the following IAM policy attached.

In the following policy, the first block of permissions creates and manages transactions. The second block allows reading data from governed tables. The third block allows the principal to interact with a governed table's manifest.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "lakeformation:StartTransaction",  
                "lakeformation:CommitTransaction",  
                "lakeformation:CancelTransaction",  
                "lakeformation:ExtendTransaction",  
                "lakeformation:DescribeTransaction",  
                "lakeformation>ListTransactions",  
  
                "lakeformation:StartQueryPlanning",  
                "lakeformation:GetQueryState",  
                "lakeformation:GetWorkUnitResults",  
                "lakeformation:GetWorkUnits",  
                "lakeformation:GetQueryStatistics",  
            ]  
        }  
    ]  
}
```

```
        "lakeformation:GetTableObjects",
        "lakeformation:UpdateTableObjects",
        "lakeformation:DeleteObjectsOnCancel"

    ],
    "Resource": "*"
}
]
```

- If you plan to read or write to governed table with transactions in a job that runs in a VPC, you must configure the Lake Formation VPC endpoint first. For more information about VPC endpoints, see [AWS Lake Formation and interface VPC endpoints \(AWS PrivateLink\) \(p. 287\)](#).

For more information about IAM permissions and policies, see [Lake Formation Permissions Reference \(p. 219\)](#).

Creating governed tables

Governed tables provide several advanced features in AWS Lake Formation, including support for ACID (atomic, consistent, isolated, and durable) transactions, automatic data compaction, and time-travel queries. Governed tables are supported only for data stored in Amazon S3.

You can create governed tables using the AWS Lake Formation console, AWS Glue API, or AWS Command Line Interface (AWS CLI).

For more information about governed tables, see [the section called “Governed tables in Lake Formation” \(p. 120\)](#).

Console

1. Ensure that all prerequisites for creating a governed table are met. For more information, see [the section called “Prerequisites for governed tables” \(p. 122\)](#).
2. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.

Sign in as a data lake administrator or as a user who has the Lake Formation CREATE_TABLE permission on the target database and who has the glue:CreateTable AWS Identity and Access Management (IAM) permission.

3. In the navigation pane, under **Data catalog**, choose **Tables**. Then choose **Create table**.
4. Fill in the **Create table** form, and specify the following:
 - Under **Data management and security**, turn on **Enable governed data access and management**.
 - Under **Data store**, specify an Amazon S3 location that is registered with Lake Formation.

AWS Glue CLI/SDK

1. Ensure that all prerequisites for creating a governed table are met. For more information, see [the section called “Prerequisites for governed tables” \(p. 122\)](#).
2. In the TableInput structure that you supply to the CreateTable operation, include TableType=GOVERNED, and specify an Amazon S3 location that is registered with Lake Formation.

For more information, see [TableInput Structure](#) in the *AWS Glue Developer Guide*.

Following is an example in Python:

```
aws glue create-table --database-name default --table-input '{
    "Name": "product_table",
    "Description": "product table for manual entries.",
    "StorageDescriptor": {
        "Columns": [
            { "Name": "product_id", "Type": "string" },
            { "Name": "product_name", "Type": "string" }
        ],
        "Location": "s3://my_bucket_name/dbs/product_table/",
        "InputFormat": "org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat",
        "OutputFormat": "org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat",
        "Compressed": false,
        "NumberOfBuckets": 0,
        "SerdeInfo": {
            "SerializationLibrary": "org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe",
            "Parameters": {}
        },
        "SortColumns": [],
        "StoredAsSubDirectories": false
    },
    "TableType": "GOVERNED",
    "Parameters": {
        "classification": "parquet",
        "lakeformation.aso.status": "true"
    }
}'
```

Athena

For instructions on creating a governed table using Athena, see [Getting Started](#) in the *Amazon Athena User Guide*.

Following is an example:

```
CREATE TABLE productsdb.product_table(
product_id string,
product_name string)
STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/governed-folder'
TBLPROPERTIES (
'table_type'='LAKEFORMATION_GOVERNED',
'classification'='parquet'
)
```

Reading and writing governed tables in Lake Formation

Governed tables have a *manifest* in Lake Formation that tracks the Amazon S3 objects that make up the table's data. Lake Formation updates the manifest when you add new objects to the data. The manifest is involved in governed table read and write operations in the following ways:

- Writes – When you add table data as new Amazon S3 objects, you call the [UpdateTableObjects](#) API operation to add those objects to the governed table manifest. You typically make one or more of these calls within a transaction, which can be rolled back if there are any failures. To call the `UpdateTableObjects` API operation, the principal needs `INSERT` and `DELETE` Lake Formation permissions on the governed table.
- Reads – You can query the data for a governed table using the following querying API operations:
 - [StartQueryPlanning](#)
 - [GetQueryState](#)
 - [GetQueryStatistics](#)
 - [GetWorkUnits](#)
 - [GetWorkUnitResults](#)

The querying API operations enforce row-level and column-level permissions. To call the API operations, the principal needs `SELECT` Lake Formation permissions on the governed table. You can optionally pass either a transaction ID or a `QueryAsOfTime` timestamp to retrieve data as of a certain time.

After an object in the manifest is created in Amazon S3 and added to a governed table, it should **not** be modified. Although Amazon S3 permits an object to be written many times, Lake Formation assumes that objects in the data lake are write-once.

For more information about transactions against governed tables, including sample code that demonstrates the use of `UpdateTableObjects` and `GetTableObjects`, see [Reading from and writing to the data lake within transactions \(p. 276\)](#).

CloudTrail data events for governed tables

When your application calls `GetTableObjects`, `UpdateTableObjects`, or `DeleteObjectsOnCancel`, the following AWS CloudTrail events are generated.

- A data event corresponding to `GetTableObjects` for the resource link.
- A management event corresponding to `GetTable` for the resource link.
- A management event corresponding to `GetTable` for the governed table.

CloudTrail data events are not enabled by default. For more information about data events in CloudTrail, see [Logging data events for trails](#) in the *AWS CloudTrail User Guide*.

If you have enabled CloudTrail data events and have shared resources using Lake Formation [resource links](#), you might not see the expected events in CloudTrail. For cross-account calls, either when using resource links or by specifying the Catalog ID in the AP call, Lake Formation emits CloudTrail events only at the caller side. The resource owner does not get a copy of the event.

Storage optimizations for governed tables

Governed tables are created with all storage optimizations features enabled by default. These include data compaction and garbage collection.

Data compaction

An important use case for governed tables is for streaming data or other applications in which small chunks of data arrive into the Amazon S3 data lake continuously. An individual table could grow to thousands of Amazon S3 objects. Each governed table maintains a *manifest* that identifies all Amazon S3 objects that the table comprises. This manifest is versioned and updated atomically to ensure that you always see a consistent view of the table.

Note

The data compaction optimizer constantly monitors your table partitions and will kick off when the threshold is exceeded for the number of files and file sizes. Lake Formation performs compaction without interfering with concurrent queries. Compaction is currently supported only for partitioned tables in the Parquet format.

Garbage collection

Another storage optimization feature of governed tables helps decrease storage costs by deleting Amazon S3 objects that are no longer part of the governed table. When a transaction is cancelled while objects are being added to the manifest for a governed table, the objects are not automatically cleaned up. This is to allow for transactions to be retried without needing to regenerate the data. In some cases, it is desirable to remove objects from canceled transactions.

To use this feature you must first call `DeleteObjectsOnCancel` before calling `S3 PutObject`. This tells Lake Formation to asynchronously delete these files to help save costs. Calling `DeleteObjectsOnCancel` provides the authorization to remove the objects from Amazon S3 in case the transaction aborts. This feature cannot be manually disabled. For more information about aborted transactions and removing unneeded objects, see [Rolling back Amazon S3 writes \(p. 277\)](#).

Note

Data compaction only works for Parquet partitioned tables.

Prerequisites for using storage optimization

Before you can use data compaction, you must complete the setup instructions described in [Prepare for using automatic data compaction with governed tables \(p. 20\)](#).

Disabling and re-enabling data compaction for governed tables

For better performance by ETL jobs and analytics services, Lake Formation automatically compacts small Amazon S3 objects of governed tables into larger objects. Data compaction is enabled for governed tables by default. You can disable compaction for individual governed tables, and re-enable compaction at a later time.

Data compaction can be enabled and disabled using either the Lake Formation console or AWS CLI.

Console

To disable or re-enable data compaction for a governed table

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

Sign in as a data lake administrator, the table creator, or a user who has been granted the `glue:UpdateTable` permission and the Lake Formation `ALTER` permission on the table.

2. In the navigation pane, choose **Tables**.
3. Choose the option button next to a table name, and on the **Actions** menu, choose **Edit**.

Note

Ensure that you choose a governed table. Governed tables have **Enabled** in the **Governance** column.

4. On the **Edit table** page, do one of the following:
 - Under **Data management and security**, select or clear the **Automatic compaction** option.
5. Choose **Save**.

AWS CLI

For example to disable compaction, you can use the following AWS CLI command.

```
aws update-table-storage-optimizer --database-name database-name --table-name table-name
--storage-optimizer-config '{"compaction": {"is_enabled": "false"}}'
```

To re-enable data compaction for a table, use similar code, but set the value of `is_enabled` to `true`.

Checking governed table compaction status

For a governed table, you can see the status of the data compaction and delete objects for canceled transactions optimizers by viewing the table in the console, or by running an AWS CLI command.

Console

To check the compaction status for governed table

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Sign in as a data lake administrator, the table creator, or a user who has been granted the `glue:GetTable` permission and any Lake Formation permission on the table.
2. In the navigation pane, choose **Tables**.
3. On the **Tables** page, choose the table name.
4. Under **Table details**, scroll down to the **Governance and acceleration details** section.

AWS CLI

Use a command similar to the following to view the configuration and last run status of all the accelerations associated with the specific table.

```
aws list-table-storage-optimizers --database-name database-name --table-name table-name
```

The following is an example of the response for this command.

```
[  
 {  
   StorageOptimizerType: "compaction",  
   config: [  
     "state": "enabled"
```

```
        },
        errorMessage: "",
        lastRunDetails: "lastRunTime: December 14, 2021. Compacted 1000 objects"
    },
    {
        StorageOptimizerType: "garbage_collection",
        config: {
            "state": "disabled"
        },
        errorMessage: "IAM role is missing DeleteObject permissions",
        lastRunDetails: "lastRunTime: December 14, 2021. Collected 1000 objects"
    }
]
```

Notes and restrictions for governed tables

Keep in mind the following notes and restrictions for governed tables:

- Currently, only Amazon Athena, Amazon Redshift Spectrum and AWS Glue ETL scripts support querying governed tables. Athena queries are limited to read-only.
- For information about querying Lake Formation tables from Amazon Redshift Spectrum, see [Using Redshift Spectrum with AWS Lake Formation](#) in the *Amazon Redshift Developer Guide*.
- Governed tables work as usual for data with encryption at rest where AWS Glue manages the encryption key. The IAM role associated with the Amazon S3 location where the governed tables reside needs to have AWS KMS permissions.
- Governed tables work as usual with Data Catalog metadata encryption enabled. The IAM role associated with the Amazon S3 location where the governed tables reside needs to have AWS KMS permissions. Additionally, you need to grant permissions to encrypt or decrypt the key to the IAM role and Lake Formation service.
- The default Lake Formation SLR role cannot be used for encrypted governed tables. You must use a custom IAM role with Amazon S3, AWS KMS and CloudWatch policies.
- To create a governed table with the AWS Management Console, you must use the Lake Formation console. You can't use the AWS Glue console.
- Only partitioned tables with Parquet-formatted files are supported for data compaction.
- You can't convert an existing non-governed table to a governed table, and you can't convert an existing governed table to a non-governed table.
- AWS Glue crawlers don't support governed tables.
- You can't use Apache Spark DataFrames to read from and write to governed tables.
- Push down predicates aren't supported in AWS Glue ETL.
- Data compaction might take longer than usual if you actively write to more than 250 partitions within a 30-minute period.
- The following features are not supported when governed tables are read using dynamic frames with AWS Glue ETL:
 - [Job bookmarks](#)
 - [Bounded execution](#)
 - [Push down predicates](#)
 - [Server-side catalog partition predicates](#)
 - [enableUpdateCatalog](#)
- The following AWS Glue API operations are not permitted on governed tables:
 - [CreatePartition](#)
 - [BatchCreatePartition](#)
 - [UpdatePartition](#)

- BatchUpdatePartitions
- DeletePartition
- BatchDeletePartition
- GetPartition
- BatchGetPartition

The reason for these restrictions is that partition operations on governed tables must be performed using the API operations that support transactions. For more information, see [Governed Table Object APIs \(p. 359\)](#).

In addition, there are restrictions on the UpdateTable API operation. You can't update the table type, change partition keys, or change the table location.

- After an object in the governed table manifest is created in Amazon S3 and added to a governed table, it should **not** be modified. Although Amazon S3 permits an object to be written many times, Lake Formation assumes that objects in the data lake are write-once.
- An Amazon S3 object can only be added to a single governed table at a time. It is strongly advised **not** to add the same Amazon S3 object within multiple active transactions to multiple governed tables.

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.

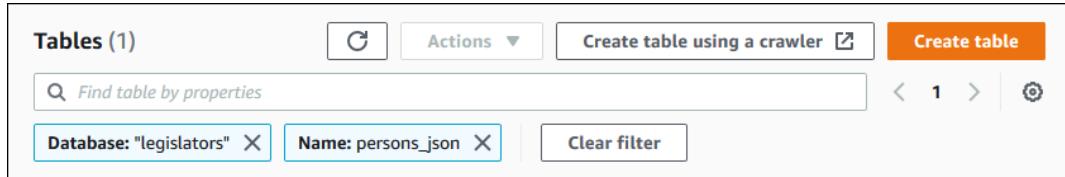
The screenshot shows the AWS Lake Formation console's 'Tables' page. At the top, there is a header with 'Tables (30)' and a 'Create table' button. Below the header is a search bar with the placeholder text 'Find table by properties'. To the right of the search bar are navigation icons for back, forward, and refresh. A sidebar on the left lists properties: 'Properties', 'Name', 'Classification', 'Database', 'Location', and 'Catalog ID'. The main area displays a table with columns for Name, Type, Last modified, and Actions.

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

The tables that you have permissions on in the database are listed.

2. (Optional) To narrow down the list to a single table in the database, position the cursor in the search field again, choose **Name** from the **Properties** menu, and either choose a table name from the **Tables** menu that appears or type a table name and press **Enter**.

The single table is listed, and both the database name and table name appear as tiles under the search field.



To adjust the filter, close either of the tiles or choose **Clear filter**.

- Search by other properties.
 1. Choose a search property from the **Properties** menu.

To search by AWS account ID, choose **Catalog ID** from the **Properties** menu, enter a valid AWS account ID (for example, 111122223333), and press **Enter**.

To search by location, choose **Location** from the **Properties** menu, and select a location from the **Locations** menu that appears. All tables in the root location of the selected location (for example, Amazon S3) are returned.

Sharing Data Catalog tables and databases across AWS Accounts

You can share Data Catalog resources (databases and tables) with external AWS accounts by granting Lake Formation permissions on the resources to the external accounts. Users can then run queries and jobs that join and query tables across multiple accounts. With some restrictions, when you share a Data Catalog resource with another account, principals in that account can operate on that resource as if the resource were in their Data Catalog.

You don't share resources with specific principals in external AWS accounts—you share the resources with an AWS account or organization. When you share a resource with an AWS organization, you're sharing the resource with all accounts at all levels in that organization. The data lake administrator in each external account must then grant permissions on the shared resources to principals in their account.

For more information, see [Cross-account data sharing in Lake Formation \(p. 199\)](#) and [Granting and revoking permissions on Data Catalog resources \(p. 180\)](#).

See Also:

- [Accessing and viewing shared Data Catalog tables and databases \(p. 130\)](#)
- [Cross-account data sharing prerequisites \(p. 200\)](#)

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

See Also:

- [Sharing Data Catalog tables and databases across AWS Accounts \(p. 130\)](#)
- Lake Formation cross-account capabilities allow users to securely share distributed data lakes across multiple AWS accounts, AWS organizations or directly with IAM principals in another account providing fine-grained access to the Data Catalog metadata and underlying data. Large enterprises typically use multiple AWS accounts, and many of those accounts might need access to a data lake managed by a single AWS account. Users and AWS Glue extract, transform, and load (ETL) jobs can query and join tables across multiple accounts and still take advantage of Lake Formation table-level and column-level data protections.

When you grant Lake Formation permissions on a Data Catalog resource to an external account or directly to an IAM principal in another account, Lake Formation uses the AWS Resource Access Manager (AWS RAM) service to share the resource. If the grantee account is in the same organization as the grantor account, the shared resource is available immediately to the grantee. If the grantee account is not in the same organization, AWS RAM sends an invitation to the grantee account to accept or reject the resource grant. Then, to make the shared resource available, the data lake administrator in the grantee account must use the AWS RAM console or AWS CLI to accept the invitation.

Direct cross-account share

Authorized principals can share resources explicitly with an IAM principal in an external account. This feature is useful when an account owner wants to have control over who in the external account can access the resources. The permissions the IAM principal receives will be a union of direct grants and the account level grants that is cascaded down to the principals. The data lake administrator of the recipient account can view the direct cross-account grants, but cannot revoke permissions. The principal who receives the resource share cannot share the resource with other principals.

Methods for sharing Data Catalog resources

With a single Lake Formation grant operation, you can grant cross-account permissions on the following Data Catalog resources.

- A database
 - An individual table (with optional column filtering)
 - A few selected tables
 - All tables in a database (by using the All Tables wildcard)
- There are two options for sharing your databases and tables with another AWS account or IAM principals in another account.
- Lake Formation tag-based access control (LF-TBAC) (recommended)

Lake Formation tag-based access control is an authorization strategy that defines permissions based on attributes. You can use tag-based access control to share Data Catalog resources (databases, tables, and columns) with external IAM principals, AWS accounts, Organizations and organizational units (OUs). In Lake Formation, these attributes are called LF-tags. For more information, see [Managing a data lake using Lake Formation tag-based access control](#).

Note

The LF-TBAC method of granting Data Catalog permissions use AWS Resource Access Manager for cross-account grants. Lake Formation now supports granting cross-account permissions to Organizations and organizational units using LF-TBAC method.

To enable this capability, you need to update the [Cross account version settings to Version 3](#).

For more information, see [Updating cross-account version settings \(p. 203\)](#).

- Lake Formation named resources

The Lake Formation cross-account data sharing using named resource method allows you to grant Lake Formation permissions with a grant option on Data Catalog tables and databases to external AWS accounts, IAM principals, organizations, or organizational units. The grant operation automatically shares those resources.

Note

You can also allow the AWS Glue crawler to access a data store in a different account using Lake Formation credentials. For more information, see [Cross-account crawling in AWS Glue Developer Guide](#).

Integrated services such as Athena and Amazon Redshift Spectrum require resource links to be able to include shared resources in queries. For more information about resource links, see [How resource links work in Lake Formation \(p. 157\)](#).

Topics

- [Cross-account data sharing prerequisites \(p. 200\)](#)
- [Updating cross-account version settings \(p. 203\)](#)

- [Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts \(p. 206\)](#)

- [Granting permissions on a database or table shared with your account \(p. 207\)](#)
- [Granting resource link permissions \(p. 209\)](#)
- [Cross-account best practices and limitations \(p. 210\)](#)
- [Accessing the underlying data of a shared table \(p. 212\)](#)
- [Cross-account CloudTrail logging \(p. 213\)](#)
- [Managing cross-account permissions using both AWS Glue and Lake Formation \(p. 216\)](#)
- [Viewing all cross-account grants using the GetResourceShares API operation \(p. 218\)](#)

Related topics

- [Overview of Lake Formation permissions \(p. 171\)](#)

- [Accessing and viewing shared Data Catalog tables and databases \(p. 130\)](#)
- [Creating resource links \(p. 157\)](#)
- [Troubleshooting cross-account access \(p. 378\)](#)

Cross-account data sharing prerequisites

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

- To share resources directly with an IAM principal in another account, you need to update the **Cross account version settings** to **Version 3**. This setting is available on the **Data catalog settings** page. If you are using **Version 1**, see instructions to update the setting [Updating cross-account version settings \(p. 203\)](#). **Version 3** of the **Cross account version settings** is also required for integrating AWS RAM resource sharing when using TABC method.
- Before granting cross-account permissions on a Data Catalog resource, you must revoke all Lake Formation permissions from the **IAMAllowedPrincipals** group for the resource. If the calling principal has cross account permissions to access a resource and **IAMAllowedPrincipals** permission exist on the resource, then Lake Formation throws **AccessDeniedException**.
- For databases that contain tables that you intend to share, you must prevent new tables from having a default grant of **Super** to **IAMAllowedPrincipals**. On the Lake Formation console, edit the database and turn off **Use only IAM access control for new tables in this database** or enter the following AWS CLI command, replacing **database** with the name of the database.

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

- You cannot share Data Catalog resources encrypted with AWS Glue service managed key with another account. You can share only Data Catalog resources encrypted with customer's encryption key, and the account receiving the resource share must have permissions on the Data Catalog encryption key to decrypt the objects.

Data sharing using LF-TBAC prerequisites

- Add the following JSON permissions object to your AWS Glue Data Catalog resource policy if you are using **Cross account version settings Version 1 or Version 2**. This policy is required for every AWS account to which you are granting permissions.

Add the `glue:PutResourcePolicy` either by using the **Settings** page on the AWS Glue console or the following AWS Command Line Interface (AWS CLI) command:

```
aws glue put-resource-policy --policy-in-json file:///  
policy.json --enable-hybrid TRUE
```

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 \(p. 216\)](#).

Data sharing using AWS Resource Access Manager prerequisites

- 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 Lake Formation tag-based access control (LF-TBAC) method, you must have a Data Catalog resource policy that enables LF-TBAC. For more information, see [Managing cross-account permissions using both AWS Glue and Lake Formation \(p. 216\)](#).
- If you want 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.

- Users who want to use AWS RAM resource sharing to grant cross-account permissions must have the required AWS Identity and Access Management (IAM) permissions on AWS Glue and AWS RAM service. The AWS managed policy `AWSLakeFormationCrossAccountManager` grants the required permissions.

Data lake administrators in accounts that receive resources that are 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": "*"  
        }  
    ]  
}
```

Setup required in each account that accesses the shared resource

- If you are sharing resources with AWS accounts, at least one user in the consumer account must be a data lake administrator to view shared resources. For information on how to create a data lake administrator, see [Create a data lake administrator \(p. 12\)](#). The data lake administrator can grant Lake Formation permissions on the shared resources to other principals in the account. Other principals can't access shared resources until the data lake administrator grants them permissions on the resources.
- The account receiving the cross-account share must have the `glue:PutResourcePolicy` permission to accept the AWS RAM resource share invitation.
- Integrated services such as Athena and Redshift Spectrum require resource links to be able to include shared resources in queries. Principals need to create a resource link in their Data Catalog to a shared resource from another AWS account. For more information about resource links, see [How resource links work in Lake Formation \(p. 157\)](#).
- When a resource is shared directly with an IAM principal, to query the table using Athena, the data lake administrator of the recipient account or an authorized principal with required permissions needs to create a resource link and grant the principal **Describe** permission on the link. The principal who received the direct cross-account share will not be able to create the resource link.
If the producer account shares a different table under the same database with the same or another principal, that principal can immediately query the table.

Important

To create a resource link, you need the Lake Formation `CREATE_TABLE` or `CREATE_DATABASE` permission, and the `glue:CreateTable` or `glue>CreateDatabase` IAM permission.

Note

For the data lake administrator and for principals whom the data lake administrator has granted permissions to, shared resources appear in the Data Catalog as if they are local (owned) resources. Extract, transform, and load (ETL) jobs can access the underlying data of shared resources.

For shared resources, the **Tables** and **Databases** pages on the Lake Formation console display the owner's account ID. When the underlying data of a shared resource is accessed, CloudTrail log events are generated in both the shared resource recipient's account and the resource owner's account. The CloudTrail events can contain the ARN of the principal that accessed the data, but only if the recipient account opts in to include the principal ARN in the logs. For more information, see [Cross-account CloudTrail logging \(p. 213\)](#).

Updating cross-account version settings

From time to time, AWS Lake Formation updates the cross-account settings to distinguish the changes made to the AWS RAM usage. When Lake Formation does this, it creates a new version of the [Cross account version settings](#).

Optimize AWS RAM resource shares

AWS Lake Formation now offers a new version of cross-account grant that optimally utilizes AWS RAM capacity to maximize cross account usage. When you share a resource with an external AWS account or an IAM principal, Lake Formation may create a new resource share or associate the resource with an existing share. By associating with existing shares, Lake Formation reduces the number of resource share invitations a consumer needs to accept.

Enable AWS RAM shares via TBAC or share resources directly to principals

To share resources directly with IAM principals in another account or to enable TBAC cross-account shares to Organizations or organizational units, you need to update the [Cross account version settings](#) to Version 3. For more information about AWS RAM resource limits, see [Cross-account best practices and limitations](#) (p. 210).

Required permissions for updating cross-account version settings

If a cross-account permission grantor has `AWSLakeFormationCrossAccountManager` managed IAM policy permissions, then there is no extra permission setup required for the cross-account permission grantor role or principal. However, if the cross-account grantor is not using the managed policy, then the grantor role or principal should have following IAM permissions granted for the new version of the cross-account grant to be successful.

| |
|----------------------------------|
| |
| { |
| "Version": "2012-10-17", |
| "Statement": [|
| { |
| "Sid": "VisualEditor1", |
| "Effect": "Allow", |
| "Action": [|
| "ram:AssociateResourceShare", |
| "ram:DisassociateResourceShare", |
| "ram:GetResourceShares" 137 |
|], |
| "Resource": "*" |



To enable the new version

Follow these steps to update **Cross account versions settings** through the AWS Lake Formation console or the AWS CLI.

Console

1. Choose **Version 2** or **Version 3** under **Cross account version settings** on the **Data catalog settings** page. If you select

Version 1, Lake Formation will use the default resource sharing mode.

The screenshot shows the 'Data catalog settings' page in the AWS Lake Formation console. In the 'Cross account version settings' section, 'Version 3' is selected from a dropdown menu. The 'Save' button is highlighted in orange at the bottom right of the page.

2. Choose **Save**.

AWS Command Line Interface (AWS CLI)

Use the `put-data-lake-settings` AWS CLI command to set the `CROSS_ACCOUNT_VERSION` parameter. Accepted values are 1, 2, and 3.

```
aws lakeformation put-data-lake-settings --region us-
```

```
east-1 --data-lake-settings file://settings
```

```
{
```

```
"DataLakeAdmins": [
```

```
{
```

```
    "D138LakePrincipalIdentifier":
```

```
        "arn:aws:iam::111122223333:user/test"
```

```
}
```

```
"Parameters": {
```

```
    "CROSS_ACCOUNT_VERSION": "3"
```

```
}
```

```
}
```

Important

Once you choose **Version 2** or **Version 3**, all new grants will go through the new cross-account grant mode. To optimally use AWS RAM capacity for your existing cross-account shares, we recommend you to revoke the grants that were made with the older version, and re-grant in the new mode.

Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts

This section includes instructions on how to enable cross-account permissions on Data Catalog tables and databases to an external AWS account, IAM principal, organization, or organizational unit. The grant operation automatically shares those resources.

Topics

- Data sharing using tag-based access control (p. 206)
- Data sharing using the named resource method (p. 207)

Data sharing using tag-based access control

Set up required on the producer/grantor account

1. Define an LF tag. For instructions to create an LF-tag, see [Creating LF-Tags \(p. 241\)](#).
2. Assign the LF-tag to the target resource. For more information, see [Assigning LF-Tags to Data Catalog resources \(p. 246\)](#).
3. Grant LF-tag permission to the external account. For more information, see [Granting LF-Tag permissions using the console \(p. 255\)](#).
At this point, the consumer data lake administrator should be able to find the policy tag being shared via the grantee account Lake Formation console, under **Permissions, Administrative roles and tasks, LF-tags**.
4. Grant data permission to the external/grantee account.
 - a. In the navigation pane, under **Permissions, Data lake permissions**, choose **Grant**.
 - b. For **Principals**, choose **External accounts**, and enter the target AWS account ID or the IAM role of the principal or the Amazon Resource Name (ARN) for the principal (principal ARN).
 - c. For **LF-tags or catalog resources**, choose the **key** and **values** of the **LF-tag** that is being shared with the consumer account (**key Confidentiality and value public**).

- d. For **Permissions**, under **Resources matched by LF-tags (recommended)** choose **Add LF-tag**.
- e. Select the **key** and **value** of the tag that is being shared with the grantee account (key Confidentiality and value public).
- f. For **Database permissions**, select **Describe** under **Database permissions** to grant access permissions at the database level.
- g. The consumer data lake administrator should be able to find the policy tag being shared via the consumer account on the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>, under **Permissions, Administrative roles and tasks, LF-tags**.
- h. Select **Describe** under **Grantable permissions** so the consumer account can grant database-level permissions to its users. 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.

Note

Principals who receive direct cross-account grants will not have the **Grantable permissions** option.

- i. For **Table and column permissions**, select **Select** and **Describe** under **Table permissions**.
- j. Select **Select** and **Describe** under **Grantable permissions**.
- k. Choose **Grant**.

Set up required on the receiving/grantee account

1. When you share a resource with another account, the resource still belongs to the producer account and is not visible within the Athena console. To make the resource visible in the Athena console, you need to create a resource link pointing to the shared resource. For instructions on creating a resource link, see [Creating a resource link to a shared Data Catalog table \(p. 159\)](#) and [Creating a resource link to a shared Data Catalog database \(p. 161\)](#)
2. You need to create a separate set of LF-tags in the consumer account to use LF tag-based access control when sharing the resource links. Create and assign the required LF-tags to the shared database/tables and the resource links.
3. Grant permissions on these LF-tags to the IAM principals in the grantee account.

Data sharing using the named resource method

You can grant permissions to directly to principals in the another AWS account, or to external AWS accounts or AWS Organizations. Granting Lake Formation permissions to Organizations or organizational units is equivalent to granting the permission to every AWS account in that organization or organizational unit.

When you grant permissions to external accounts or organizations, you must include the **Grantable permissions** option. Only the data lake

administrator in the external account can access the shared resources until the administrator grants permissions on the shared resources to other principals in the external account.

Note

Grantable permissions option is not supported when granting permissions directly to IAM principals from external accounts.

Follow instructions in Granting database permissions using the Lake Formation console and the named resource method (p. 181) to grant cross-account permissions using the named resource method.

Granting permissions on a database or table shared with your account

After a Data Catalog resource belonging to another AWS account is shared with your AWS account, as a data lake administrator, you can grant permissions on the shared resource to other principals in your account. You can't, however, grant permissions on the resource to other AWS accounts or organizations.

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

To grant permissions on a shared database (named resource

method, console)

- Follow the instructions in Granting database permissions using the Lake Formation console and the named resource method (p. 181). In the **Database** list under **LF-Tags or catalog resources**, ensure that you select the database in the external account, not a resource link for the database.
If you don't see the database in the list of databases, ensure that you have accepted the AWS Resource Access Manager (AWS RAM) resource share invitation for the database. For more information, see Accepting a resource share invitation from AWS RAM (p. 154). Also, for the CREATE_TABLE and ALTER permissions, follow the instructions in Granting data location permissions (same account) (p. 174), 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. 185). In the **Database** list under **LF-Tags or catalog resources**, ensure that you select the database in the external account, not a resource link for the database.
If you don't see the table in the list of tables, ensure that you have accepted the AWS RAM resource share invitation for the table. For more information, see Accepting a resource share invitation from AWS RAM (p. 154).

Also, for the ALTER permission, follow the instructions in Granting data location permissions (same account) (p. 174), and be sure to

enter the owning account ID in the **Registered account location** field.

To grant permissions on shared resources (LF-TBAC method,

console)

- Follow the instructions in Granting Data Catalog permissions using the Lake Formation console and the LF-TBAC Method (p. 192). In the **LF-Tags or catalog resources** section, grant the exact LF-tag expression that the external account granted to your account, or a subset of that expression.

For example, if an external account granted the LF-tag expression `module=customers AND environment=production` to your account with the grant option, as a data lake administrator, you can grant that same expression, or `module=customers` or `environment=production` to a principal in your account. You can grant only the same or a subset of the Lake Formation permissions (for example, SELECT, ALTER, and so on) that were granted on resources through the LF-tag expression.

To grant permissions on a shared table (named resource

method, AWS CLI)

- Enter a command similar to the following. In this example:
 - Your AWS account ID is 1111-2222-3333.
 - The account that owns the table and that granted it to your account is 1234-5678-9012.
 - The SELECT permission is being granted on the shared table `pageviews` to user `datalake_user1`. That user is a principal in your account.
 - The `pageviews` table is in the `analytics` database, which is owned by account 1234-5678-9012.

```
aws lakeformation grant-permissions --principal
  DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/
  datalake_user1 --permissions "SELECT" --resource
  '{ "Table": { "CatalogId": "123456789012",
  "DatabaseName": "analytics", "Name": "pageviews"} }'
```

Note that the owning account must be specified in the `CatalogId` property in the `resource` argument.

Granting resource link permissions

Follow these steps to grant AWS Lake Formation permissions on one or more resource links to a principal in your AWS account.

After you create a resource link, only you can view and access it. (This assumes that **Use only IAM access control for new tables in this database** is not enabled for the database.) To permit other principals in your account to access the resource link, grant at least the DESCRIBE permission.

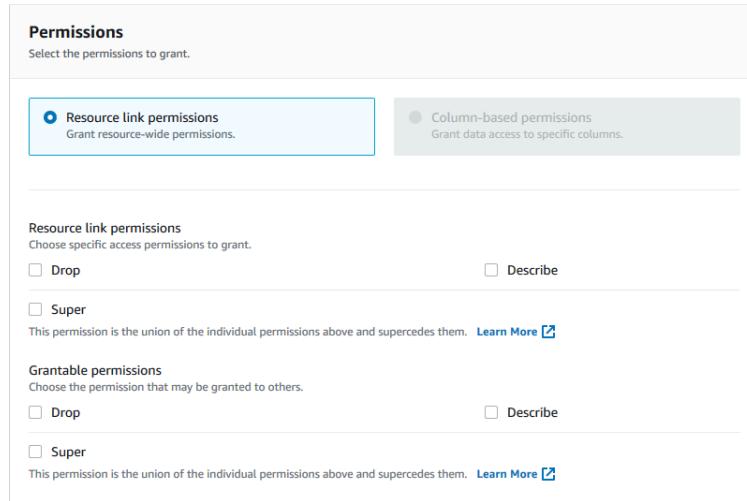
Important

Granting permissions on a resource link doesn't grant permissions on the target (linked) database or table. You must grant permissions on the target separately.

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

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. 181) to do the following:
 1. Open the **Grant Permissions** page (p. 181).
 2. Specify the databases (p. 182). Specify one or more database resource links.
 3. Specify the principals (p. 181).
 - For table resource links, follow the steps in Granting table permissions using the Lake Formation console and the named resource method (p. 185) to do the following:
 1. Open the Grant permissions page (p. 186).
 2. Specify the tables (p. 187). Specify one or more table resource links.
 3. Specify the principals (p. 186).
2. Under **Permissions**, select the permissions to grant. Optionally, select grantable permissions.



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. 157)
- Lake Formation Permissions Reference (p. 219)

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, Lake Formation uses AWS Resource Access Manager (AWS RAM) capacity for cross-account grants that your account can make with the named resource method. To maximize the AWS RAM capacity, follow these best practices for the named resource method:
 - Use the new cross-account grant mode (**Version 3** under **Cross account version settings**) to share a resource with an external AWS account. For more information, see Updating cross-account version settings (p. 203).
 - 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. 130).
 - 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 permissions on Data Catalog resources (p. 180).

Note

For more information about requesting a higher limit for the number of resource shares in AWS RAM, see [AWS service quotas in the AWS General Reference](#).

- You must create a resource link to a shared database for that database to appear in the Amazon Athena and Amazon Redshift Spectrum query editors. Similarly, to be able to query shared tables using Athena and Redshift Spectrum, you must create resource links to the tables. The resource links then appear in the tables list of the query editors.
Instead of creating resource links for many individual tables for querying, you can use the **All tables** wildcard to grant permissions on all tables in a database. Then, when you create a resource link for that database and select that database resource link in the query editor, you'll have access to all tables in that database for your query. For more information, see [Creating resource links \(p. 157\)](#).
- When you share resources directly with principals in another account, the IAM principal in the recipient account may not have permission to create resource links to be able to query the shared tables using Athena and Amazon Redshift Spectrum. Instead of creating a resource link for each table that is shared, the data lake administrator can create a placeholder database and grant `CREATE_TABLE` permission to the `ALLIAMPPrincipal` group. Then, all IAM principals in the recipient account can create resource links in the placeholder database and start querying the shared tables.
See the example CLI command for granting permissions to `ALLIAMPPrincipals` in [Granting database permissions using the AWS CLI and the named resources method \(p. 184\)](#).
- Athena and Redshift Spectrum support column-level access control, but only for inclusion, not exclusion. Column-level access control is not supported in AWS Glue ETL jobs.
- When a resource is shared with your AWS account, you can grant permissions on the resource only to users in your account. You can't grant permissions on the resource to other AWS accounts, to organizations (not even your own organization), or to the `IAMAllowedPrincipals` group.
- You can't grant `DROP` or `Super` on a database to an external account.
- Revoke cross-account permissions before you delete a database or table. Otherwise, you must delete orphaned resource shares in [AWS Resource Access Manager](#).

See Also

- [Lake Formation Tag-based access control notes and restrictions \(p. 239\)](#)
- [CREATE_TABLE \(p. 224\)](#) in the [Lake Formation Permissions Reference \(p. 219\)](#) 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 Lake Formation tag-based access control method.)
- The data lake administrator must re-grant to the principal the Lake Formation SELECT permission that account A granted on the shared table.
- The principal must have the following IAM permissions on the table, the database that contains it, and the account A Data Catalog.

Note

In the following IAM policy:

- Replace `<account-id-A>` with the AWS account ID of account A.
- Replace `<region>` with a valid Region.
- Replace `<database>` with the name of the database in account A that contains the shared table.
- Replace `<table>` with the name of the shared table.

```
{  
    "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:glue:<region>:<account-id-A>:table/  
                <database>/<table>"  
            ]  
        }  
    ]  
}
```

```
"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:

- Accepting a resource share invitation from AWS RAM (p. 154)

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. 213)
- Querying CloudTrail logs for Amazon S3 cross-account access (p. 214)

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":  
      "AROAQGFTBBGOWBV2EMZA: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.

```
"lakeFormationRoleSessionName": "AWSLF-00-  
GL-111122223333-G13T0Rmng2"  
,  
...  
}
```

To opt in to include principal Rans in cross-account CloudTrail

logs

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
Sign in as the Administrator user, or a user with the Administrator Access IAM policy.
2. In the navigation pane, choose **Settings**.
3. On the **Data catalog settings** page, in the **Default permissions for AWS CloudTrail** section, for **Resource owners**, enter one or more AWS resource owner account IDs.
Press **Enter** after each account ID.
4. Choose **Save**.
Now cross-account CloudTrail events stored in the logs for both the shared resource recipient and the resource owner contain the principal ARN.

Querying CloudTrail logs for Amazon S3 cross-account access

As a shared resource owner, you can query S3 CloudTrail logs to determine the accounts that have accessed your Amazon S3 buckets (provided that you enabled object-level logging in Amazon S3). This applies only to S3 locations that you registered with Lake Formation. If shared resource consumers opt in to include principal Rans 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="Put Object"`.

The following is an excerpt from a Lake Formation cross-account CloudTrail event where data in a registered S3 location was accessed.

```
{  
  "eventSource": "lakeformation.amazonaws.com",  
  "eventName": "GetDataAccess",  
  .....  
  .....  
  "additionalEvent149a": {  
    "requesterService": "GLUE_JOB",  
    "lakeFormationPrincipal": "arn:aws:iam::111122223333:role/
```

```
"lakeFormationRoleSessionName": "AWSLF-00-GL-111122223333-  
B8JSAjo5QA"  
}  
}
```

The `lakeFormationRoleSessionName` key value, `AWSLF-00-GL-111122223333-B8JSAjo5QA`, can be joined with the session name in the `principalId` key of the S3 CloudTrail event. The following is an excerpt from the S3 CloudTrail event. It shows the location of the session name.

```
{  
  "eventSource": "s3.amazonaws.com",  
  "eventName": "Get Object",  
  ....  
  ....  
  "principalId": "AROAQSOX5XXUR7D6RMYLR:AWSLF-00-GL-111122223333-B8JSAjo5QA",  
  "arn": "arn:aws:sets::111122223333:assumed-role/  
Deformationally/AWSLF-00-GL-111122223333-B8JSAjo5QA",  
  "session Context": {  
    "session Issuer": {  
      "type": "Role",  
      "principalId": "AROAQSOX5XXUR7D6RMYLR",  
      "arn": "arn:aws:iam::111122223333:role/aws-service-  
role/lakeformation.amazonaws.com/Deformationally",  
      "accountId": "111122223333",  
      "user Name": "Deformationally"  
    },  
    ....  
    ....  
  }  
}
```

150 The session name is formatted as follows:

```
AWSLF-<version-number>-<query-engine-code>-<account-id>-
```

suffix

A randomly generated string.

Managing cross-account permissions using both AWS Glue and Lake Formation

It's possible to grant cross-account access to Data Catalog resources and underlying data by using either AWS Glue or AWS Lake Formation.

In AWS Glue you grant cross-account permissions by creating or updating a Data Catalog resource policy. In Lake Formation, you grant cross-account permissions by using the Lake Formation GRANT/REVOKE permissions model and the `Grant_Permissions` API operation.

Tip

We recommend that rely solely on Lake Formation permissions to secure your data lake.

You can view Lake Formation cross-account grants by using the Lake Formation console or, 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` field 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": "Cataloguers",
            "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 = json.dumps(policy)
glue.put_resource_policy(PolicyInJson=policy,
    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

When using the Lake Formation tag-based access control (LF-TBAC) method to make cross-account grants, if the **Cross account version settings** is **Version 1** or **Version 2** you must have a Data Catalog resource policy with at least the permissions specified in Cross-account data sharing prerequisites (p. 200).

See Also:

- [Metadata access control \(p. 292\)](#) (for a discussion of the named resource method versus the Lake Formation tag-based access control (LF-TBAC) method).
- [Viewing shared Data Catalog tables and databases \(p. 156\)](#)
- [Working with Data Catalog Settings on the AWS Glue Console in the AWS Glue Developer Guide](#)
- [Granting Cross-Account Access in the AWS Glue Developer Guide](#) (for sample Data Catalog resource policies)

Viewing all cross-account grants using the GetResourceShares API operation

If your enterprise grants cross-account permissions using both an AWS Glue Data Catalog resource policy and Lake Formation grants, the only way to view all cross-account grants in one place is to use the `glue:GetResourceShares` API operation.

When you grant Lake Formation permissions across accounts by using the named resource method, AWS Resource Access Manager (AWS RAM) creates an AWS Identity and Access Management (IAM) resource policy and stores it in your AWS account. The policy grants the permissions required to access the resource. AWS RAM creates a separate resource policy for each cross-account grant. You can view all of these policies by using the `glue:GetResourceShares` API operation.

Note

This operation also returns the Data Catalog resource policy. However, if you enabled meta data encryption in Data Catalog settings, and you don't have permission on the AWS KMS key, the operation won't return the Data Catalog resource policy.

To view all cross-account grants

-
- [Enter the following AWS CLI command.](#)

```
aws glue get-resource-policies
```

The following is an example resource policy that AWS RAM creates and stores when you grant permissions on table t in database db1 to AWS account 1111-2222-3333.

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

See Also:

-
- [GetResourceShares Action \(Python: get_resource_policies\) in the AWS Glue Developer Guide](#)

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" \(p. 372\)](#).

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" \(p. 372\)](#).

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",  
            "status": "ACCEPTED"  
        }  
    ]  
}
```

```
        "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) | | | | | |
|------------------|--|-----------|------------------------------|-----------------|-----------------------|
| | | Actions | Create table using a crawler | | Create table |
| Name | | Database | Owner account ... | Shared resource | Shared resource owner |
| adviews | | analytics | 111122223333 | - | - |
| pageviews | | analytics | 111122223333 | - | - |
| blackholes | | hubble | 123456789012 | - | - |
| celestial-events | | hubble | 123456789012 | - | - |
| 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' section of the AWS Lake Formation console. It lists four grants for the 'clickthroughs' table in the 'analytics' database. The grants are:

| Principal | Principal type | Resource type | Resource | Owner account ID | Permissions |
|----------------|----------------|---------------|---------------------------|------------------|------------------------------------|
| datalake_admin | IAM user | Table | clickthroughs | 123456789012 | Super, Alter, Delete, Drop, Insert |
| datalake_admin | IAM user | Column | analytics.clickthroughs.* | 123456789012 | Select |
| 111122223333 | AWS account | Table | clickthroughs | 123456789012 | Insert |
| 111122223333 | 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 \(p. 157\)](#)
- [Creating a resource link to a shared Data Catalog table \(p. 159\)](#)
- [Creating a resource link to a shared Data Catalog database \(p. 161\)](#)
- [Resource link handling in AWS Glue APIs \(p. 162\)](#)

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="button" value="G"/> | Actions | Create table using a crawler | <input type="button" value="Create table"/> |
|---|-----------------------|----------------------------------|-------------------|------------------------------|---|
| <input type="text"/> Find table by properties | | | | | |
| | Name | Database | Owner account ... | Shared resource | Shared resource owner |
| <input type="radio"/> | <i>inventory-link</i> | retail | 123456789012 | inventory | 111122223333 |
| <input type="radio"/> | <i>incidents-link</i> | issues-local | 123456789012 | incidents | 111122223333 |
| <input type="radio"/> | site-logs | logs | 123456789012 | - | - |
| <input type="radio"/> | alexa-logs | 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 [Lake Formation Permissions Reference \(p. 219\)](#).

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 permissions on Data Catalog resources \(p. 180\)](#)

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 [Accepting a resource share invitation from AWS RAM \(p. 154\)](#).

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.

The screenshot shows the 'Table details' section of the AWS Data Catalog creation interface. It has the following fields:

- Table type:** A radio button group with 'Table' (unselected) and 'Resource Link' (selected). Below it is the text 'Create a table in my account'.
- Resource link name:** An input field containing 'clickthroughs-link'. Below it is the note 'Name may contain letters (A-Z), numbers (0-9), hyphens (-), or underscores (_), and must be less than 256 characters long.'
- Database:** A dropdown menu showing 'adtrack'.
- Shared table:** A dropdown menu showing 'clickthroughs'.
- Shared table's database:** An input field containing 'analytics'.
- Shared table owner ID:** An input field containing a redacted account ID.
- Buttons:** 'Cancel' and 'Create' buttons at the bottom right.

- Choose **Create** to create the resource link.

You can then view the resource link name under the **Name** column on the **Tables** page.

- (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)

- 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:

- [How resource links work in Lake Formation \(p. 157\)](#)
- [DESCRIBE \(p. 226\)](#)

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 [Accepting a resource share invitation from AWS RAM \(p. 154\)](#).

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.

The screenshot shows the 'Database details' section of the AWS Glue console. A radio button for 'Resource Link' is selected, indicating the creation of a shared database resource link. The 'Resource link name' field contains 'analytics-db-link'. The 'Shared database' dropdown shows a search result for 'analytics'. The 'Shared database owner ID' field is empty. At the bottom right are 'Cancel' and 'Create' buttons, with 'Create' being highlighted.

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:

- [How resource links work in Lake Formation \(p. 157\)](#)
- [DESCRIBE \(p. 226\)](#)

Resource link handling in AWS Glue APIs

The following tables explain how the AWS Glue Data Catalog APIs handle database and table resource links. For all Get* API operations, only databases and tables that the caller has permissions on get returned. Also, when accessing a target database or table through a resource link, you must have both AWS Identity and Access Management (IAM) and Lake Formation permissions on both the target and the

resource link. The Lake Formation permission that is required on resource links is DESCRIBE. For more information, see [DESCRIBE \(p. 226\)](#).

Database API Operations

| API Operation | Resource Link Handling |
|----------------|---|
| CreateDatabase | If the database is a resource link, creates the resource link to the designated target database. |
| UpdateDatabase | If the designated database is a resource link, follows the link and updates the target database. If the resource link must be modified to link to a different database, you must delete it and create a new one. |
| DeleteDatabase | Deletes the resource link. It doesn't delete the linked (target) database. |
| GetDatabase | If the caller has permissions on the target, follows the link to return the target's properties. Otherwise, it returns the properties of the link. |
| GetDatabases | Returns a list of databases, including resource links. For each resource link in the result set, the operation follows the link to get the properties of the link target. You must specify ResourceShareType = ALL to see the databases shared with your account. |

Table API Operations

| API Operation | Resource Link Handling |
|------------------|---|
| CreateTable | If the database is a resource link, follows the database link and creates a table in the target database. If the table is a resource link, the operation creates the resource link in the 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 |
|-------------------------|--|
| SearchTables | Returns tables and table resource links. It doesn't follow links to return target table properties. You must specify ResourceShareType = ALL to see tables shared with your account. |
| GetTableVersion | Same as GetTable. |
| GetTableVersions | Same as GetTable. |
| DeleteTableVersion | Same as DeleteTable. |
| BatchDeleteTableVersion | Same as DeleteTable. |

Partition API Operations

| API Operation | Resource Link Handling |
|----------------------|---|
| CreatePartition | If the designated database is a resource link, follows the database link and creates a partition in the designated table in the target database. If the table is a resource link, the operation follows the resource link and creates the partition in the target table. Creating a partition through both a table resource link and database resource link is not supported. |
| BatchCreatePartition | Same as CreatePartition. |
| UpdatePartition | If the designated database is a resource link, follows the database link and updates the partition in the designated table in the target database. If the table is a resource link, the operation follows the resource link and updates the partition in the target table. Updating a partition through both a table resource link and database resource link is not supported. |
| DeletePartition | If the designated database is a resource link, follows the database link and deletes the partition in the designated table in the target database. If the table is a resource link, the operation follows the resource link and deletes the partition in the target table. Deleting a partition through both a table resource link and database resource link is not supported. |
| BatchDeletePartition | Same as DeletePartition. |
| GetPartition | If the designated database is a resource link, follows the database link and returns partition information from the designated table. Otherwise, if the table is a resource link, the operation follows the link and returns partition information. If both the table and database are resource links, it returns an empty result set. |
| GetPartitions | If the designated database is a resource link, follows the database link and returns partition information for all partitions in the designated table. Otherwise, if the table is a resource link, the operation follows the link and returns partition information. If both the table and database are resource links, it returns an empty result set. |
| BatchGetPartition | Same as GetPartition. |

User-Defined Functions API Operations

| API Operation | Resource Link Handling |
|----------------------|--|
| (All API operations) | If the database is a resource link, follows the resource link and performs the operation on the target database. |

See Also:

- [How resource links work in Lake Formation \(p. 157\)](#)

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 \(p. 166\)](#)
- [Creating a workflow \(p. 167\)](#)
- [Running a workflow \(p. 169\)](#)

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"> Schema evolution is flexible. (Columns are renamed, previous columns are deleted, and new columns are added in their place.) Complete consistency is needed between the source and the destination. | <ul style="list-style-type: none"> Schema evolution is incremental. (There is only successive addition of columns.) Only new rows are added; previous rows are not updated. |

Note

Users cannot edit blue prints and workflows created by Lake Formation.

Creating a workflow

Before you start, ensure that you have granted the required data permissions and data location permissions to the role `LakeFormationWorkflowRole`. This is so the workflow can create metadata tables in the Data Catalog and write data to target locations in Amazon S3. For more information, see [Create an IAM role for workflows \(p. 11\)](#) and [Overview of Lake Formation permissions \(p. 171\)](#).

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 \(p. 371\)](#).
2. In the navigation pane, choose **Blueprints**, and then choose **Use blueprint**.
3. On the **Use a blueprint** page, choose a tile to select the blueprint type.
4. Under **Import source**, specify the data source.

If you are importing from a JDBC source, specify the following:

- **Database connection**—Choose a connection from the list. Create additional connections using the AWS Glue console. The JDBC user name and password in the connection determine the database objects that the workflow has access to.
- **Source data path**—Enter `<database>/<schema>/<table>` or `<database>/<table>`, depending on the database product. Oracle Database and MySQL don't support schema in the path. You can substitute the percent (%) character for `<schema>` or `<table>`. For example, for an Oracle database with a system identifier (SID) of `orcl`, enter `orcl/%` to import all tables that the user named in the connection has access to.

Important

This field is case sensitive. The workflow will fail if there is a case mismatch for any of the components.

If you specify a MySQL database, AWS Glue ETL uses the MySQL5 JDBC driver by default, so MySQL8 is not natively supported. You can edit the ETL job script to use a `customJdbcDriverS3Path` parameter as described in [JDBC connectionType Values](#) in the [AWS Glue Developer Guide](#) to use a different JDBC driver that supports MySQL8.

If you are importing from a log file, ensure that the role that you specify for the workflow (the "workflow role") has the required IAM permissions to access the data source. For example, to import AWS CloudTrail logs, the user must have the `cloudtrail:DescribeTrails` and `cloudtrail:LookupEvents` permissions to see the list of CloudTrail logs while creating the workflow, and the workflow role must have permissions on the CloudTrail location in Amazon S3.

5. Do one of the following:

- For the **Database snapshot** blueprint type, optionally identify a subset of data to import by specifying one or more exclude patterns. These exclude patterns are Unix-style glob patterns. They are stored as a property of the tables that are created by the workflow.

For details on the available exclude patterns, see [Include and Exclude Patterns](#) in the *AWS Glue Developer Guide*.

- For the **Incremental database** blueprint type, specify the following fields. Add a row for each table to import.

Table name

Table to import. Must be all lower case.

Bookmark keys

Comma-delimited list of column names that define the bookmark keys. If blank, the primary key is used to determine new data. Case for each column must match the case as defined in the data source.

Note

The primary key qualifies as the default bookmark key only if it is sequentially increasing or decreasing (with no gaps). If you want to use the primary key as the bookmark key and it has gaps, you must name the primary key column as a bookmark key.

Bookmark order

When you choose **Ascending**, rows with values greater than bookmarked values are identified as new rows. When you choose **Descending**, rows with values less than bookmarked values are identified as new rows.

Partitioning scheme

(Optional) List of partitioning key columns, delimited by slashes (/). Example: year/month/day.

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 | Remove |
|---|---|--|--|---------------------------------------|
| <input type="text" value="Enter a table name"/> | <input type="text" value="Enter a bookmark"/> | <input type="text" value="Choose a sort. ▾"/> Comma-delimited list of bookmark columns. | <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.

8. Under **Import options**:
 - a. Enter a workflow name.
 - b. For role, choose the role `LakeFormationWorkflowRole`, which you created in [Create an IAM role for workflows \(p. 11\)](#).
 - c. Optionally specify a table prefix. The prefix is prepended to the names of Data Catalog tables that the workflow creates.
9. Choose **Create**, and wait for the console to report that the workflow was successfully created.

Tip

Did you get the following error message?

User: `arn:aws:iam:<account-id>:user/<username>` is not authorized to perform: `iam:PassRole` on resource: `arn:aws:iam::<account-id>:role/<rolename>...`

If so, check that you replaced `<account-id>` with a valid AWS account number in all policies.

See also:

- [Blueprints and workflows in Lake Formation \(p. 166\)](#)

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 \(p. 371\)](#).
2. In the navigation pane, choose **Blueprints**.
3. On the **Blueprints** page, select the workflow. Then on the **Actions** menu, choose **Start**.
4. As the workflow runs, view its progress in the **Last run status** column. Choose the refresh button occasionally.

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

When the workflow is complete:

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

If the workflow fails, do the following:

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

The workflow opens in the AWS Glue console.
- b. Ensure that the workflow is selected, and choose the **History** tab.
- c. Under **History**, select the most recent run and choose **View run details**.
- d. Select a failed job or crawler in the dynamic (runtime) graph, and review the error message. Failed nodes are either red or yellow.

See also:

- [Blueprints and workflows in Lake Formation \(p. 166\)](#)

Managing Lake Formation permissions

Lake Formation provides central access controls for data in your data lake. You can define security policy-based rules for your users and applications by role in Lake Formation, and integration with AWS Identity and Access Management authenticates those users and roles. Once the rules are defined, Lake Formation enforces your access controls at table and column-level granularity for users of Amazon Redshift Spectrum and Amazon Athena.

Topics

- [Overview of Lake Formation permissions \(p. 171\)](#)
- [Granting data location permissions \(p. 174\)](#)
- [Granting and revoking permissions on Data Catalog resources \(p. 180\)](#)
- [Viewing Database and Table Permissions in Lake Formation \(p. 196\)](#)
- [Cross-account data sharing in Lake Formation \(p. 199\)](#)
- [Revoking permission using the Lake Formation console \(p. 219\)](#)
- [Lake Formation Permissions Reference \(p. 219\)](#)

Overview of Lake Formation permissions

There are two main types of permissions in AWS Lake Formation:

- Metadata access – Permissions on Data Catalog resources (*Data Catalog permissions*).
These permissions enable principals to create, read, update, and delete metadata databases and tables in the Data Catalog.
- Underlying data access – Permissions on locations in Amazon Simple Storage Service (Amazon S3) (*data access permissions* and *data location permissions*).
 - Data lake permissions enable principals to read and write data to *underlying* Amazon S3 locations—data pointed to by Data Catalog resources.
 - Data location permissions enable principals to create and alter metadata databases and tables that point to specific Amazon S3 locations.

For both types, Lake Formation uses a combination of Lake Formation permissions and 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.

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.

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 Lake Formation tag-based access control (LF-TBAC), principals need additional IAM permissions. For more information, see [Lake Formation tag-based access control permissions model \(p. 237\)](#) and [Lake Formation Personas and IAM Permissions Reference \(p. 371\)](#).

Cross-account permissions

Users who want to grant cross-account Lake Formation permissions by using the named resource method must also have the permissions in the `AWSLakeFormationCrossAccountManager` AWS managed policy.

Data lake administrators need those same permissions for granting cross-account permissions, plus the AWS Resource Access Manager (AWS RAM) permission to enable granting permissions to organizations. For more information, see [Data Lake Administrator Permissions \(p. 372\)](#).

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.

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 [Create a data lake administrator \(p. 12\)](#).

Database creators

- Have all database permissions on databases that they create, have permissions on tables that they create in the database, and can grant other principals in the same AWS account permission to create tables in the database. A database creator who also has the `AWSLakeFormationCrossAccountManager` AWS managed policy can grant permissions on the database to other AWS accounts or organizations.

Data lake administrators can use the Lake Formation console or API to designate database creators.

Note

Database creators do not implicitly have permissions on tables that others create in the database.

For more information, see [Creating a Database \(p. 119\)](#).

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 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 \(p. 295\)](#)

Topics

- [Granting data location permissions \(same account\) \(p. 174\)](#)
- [Granting data location permissions \(external account\) \(p. 177\)](#)
- [Granting permissions on a data location shared with your account \(p. 179\)](#)

Granting data location permissions (same account)

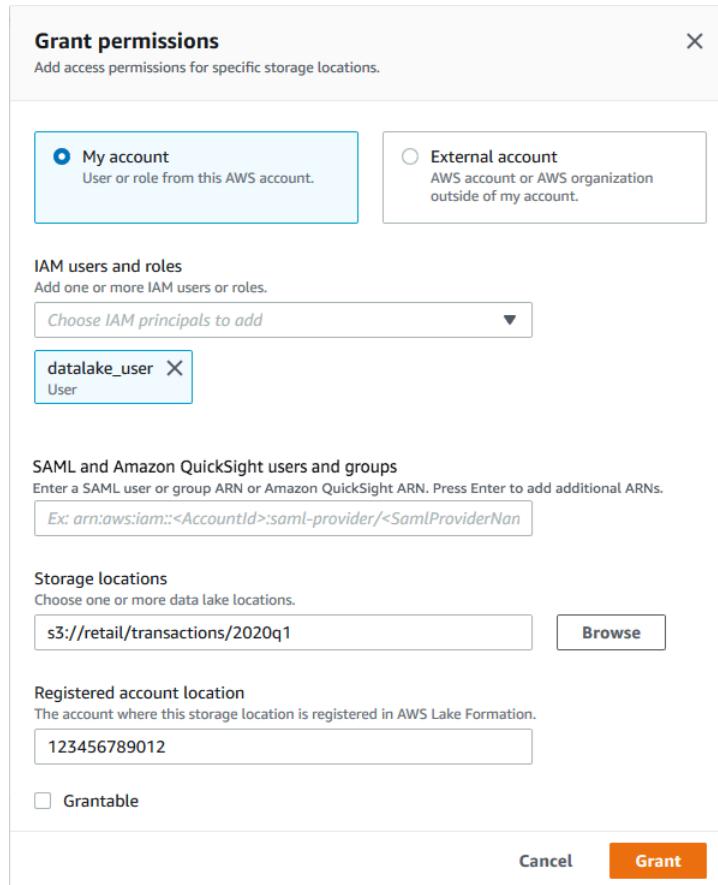
Follow these steps to grant data location permissions to principals in your AWS account. You can grant permissions by using the Lake Formation console, the API, or the AWS Command Line Interface (AWS CLI).

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 [Lake Formation Grant and Revoke AWS CLI Commands \(p. 220\)](#).

- For **Storage locations**, choose **Browse**, and choose an Amazon Simple Storage Service (Amazon S3) storage location. The location must be registered with Lake Formation. Choose **Browse** again to add another location. You can also type the location, but ensure that you precede the location with `s3://`.
- For **Registered account location**, enter the AWS account ID where the location is registered. This defaults to your account ID. In a cross-account scenario, data lake administrators in a recipient account can specify the owner account here when granting the data location permission to other principals in the recipient account.
- (Optional) To enable the selected principals to grant data location permissions on the selected location, select **Grantable**.



5. Choose **Grant**.

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" } }
```

Example

The following example grants data location permissions on `s3://retail` to `AllIAMPrecipals` group.

```
aws lakeformation grant-permissions --principal
  DataLakePrincipalIdentifier=111122223333:IAMPrincipals --
```

```
permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation": {"ResourceArn": "arn:aws:s3:::retail", "CatalogId": "111122223333"} }'
```

See Also:

- [Lake Formation Permissions Reference \(p. 219\)](#)

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 [Cross-account data sharing prerequisites \(p. 200\)](#).

To grant data location permissions (external account, console)

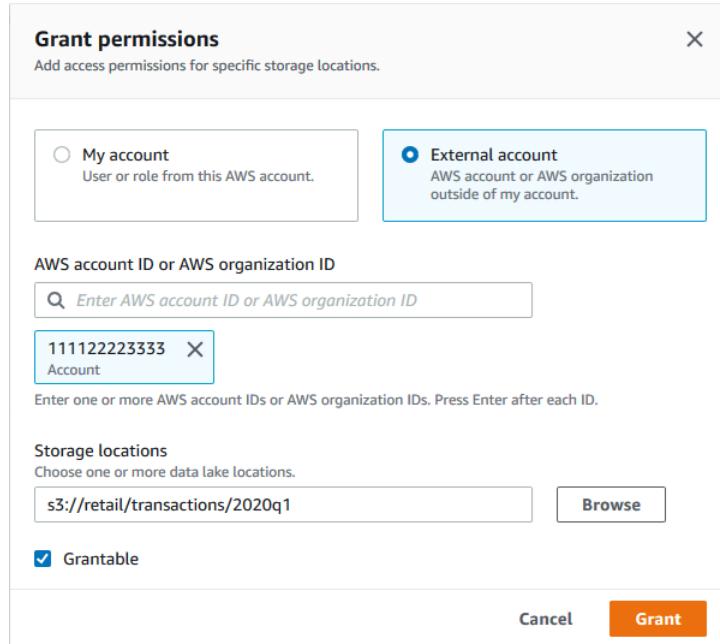
1. Open the AWS Lake Formation console at <https://console.aws.amazon.com/lakeformation/>. Sign in as a data lake administrator.
2. In the navigation pane, choose **Data locations**, and then choose **Grant**.
3. In the **Grant permissions** dialog box, choose the **External account** tile.
4. Provide 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 "-" (hyphen) and 8 to 32 additional lowercase letters or digits.

- Under **Storage locations**, choose **Browse**, and choose an Amazon Simple Storage Service (Amazon S3) storage location. The location must be registered with Lake Formation.



5. Select **Grantable**.
6. Choose **Grant**.

To grant data location permissions (external account, AWS CLI)

- To grant permissions to an external AWS account, enter a command similar to the following.

```
aws lakeformation grant-permissions --principal
  DataLakePrincipalIdentifier=111122223333 --permissions "DATA_LOCATION_ACCESS" --
  permissions-with-grant-option "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
    {"CatalogId":"123456789012", "ResourceArn": "arn:aws:s3:::retail/transactions/2020q1"}'}
```

This command grants DATA_LOCATION_ACCESS with the grant option to account 1111-2222-3333 on the Amazon S3 location s3://retail/transactions/2020q1, which is owned by account 1234-5678-9012.

To grant permissions to an organization, enter a command similar to the following.

```
aws lakeformation grant-permissions --principal
  DataLakePrincipalIdentifier=arn:aws:organizations::111122223333:organization/
  o-abcdefghijkl --permissions "DATA_LOCATION_ACCESS" --permissions-
  with-grant-option "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":
    {"CatalogId":"123456789012", "ResourceArn": "arn:aws:s3:::retail/transactions/2020q1"}'}
```

This command grants DATA_LOCATION_ACCESS with grant option to the organization o-abcdefghijkl on the Amazon S3 location s3://retail/transactions/2020q1, which is owned by account 1234-5678-9012.

To grant permissions to a principal in an external AWS account, enter a command similar to the following.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1  
--permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":  
{"ResourceArn":"arn:aws:s3:::retail/transactions/2020q1", "CatalogId":  
"123456789012"}}'
```

This command grants DATA_LOCATION_ACCESS to a principal in account 1111-2222-3333 on the Amazon S3 location s3://retail/transactions/2020q1, which is owned by account 1234-5678-9012.

Example

The following example grants data location permissions on s3://retail to ALLIAMPrecipals group in an external account.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=111122223333:IAMPrincipals --  
permissions "DATA_LOCATION_ACCESS" --resource '{ "DataLocation":  
{"ResourceArn":"arn:aws:s3:::retail", "CatalogId": "123456789012"}}'
```

See Also:

- [Lake Formation Permissions Reference \(p. 219\)](#)

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 \(same account\) \(p. 174\)](#).

For **Storage locations**, you must type the locations. For **Registered account location**, enter the AWS account ID of the owner account.

To grant permissions on a data location shared with your account (AWS CLI)

- Enter one of the following commands to grant permissions to either a user or a role.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:user/<user-name> --permissions  
"DATA_LOCATION_ACCESS" --resource '{ "DataLocation": { "CatalogId": "<owner-account-  
ID>", "ResourceArn": "arn:aws:s3:::<s3-location>"}}'  
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=arn:aws:iam::<account-id>:role/<role-name> --permissions  
"DATA_LOCATION_ACCESS" --resource '{ "DataLocation": { "CatalogId": "<owner-account-  
ID>", "ResourceArn": "arn:aws:s3:::<s3-location>"}}'
```

Granting and revoking permissions on Data Catalog resources

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 method or the Lake Formation tag-based access control (LF-TBAC) method.

You can grant permissions to principals in the same AWS account or to external accounts or organizations. When you grant to external accounts or organizations, you are sharing 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 LF-TBAC method supports granting cross-account permissions to IAM principals, AWS accounts, AWS Organizations, and organizational units (OUs).

When you grant permissions to external accounts or organizations, you must include the grant option. Only the data lake administrator in the external account can access the shared 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 \(p. 181\)](#)
- [Granting Data Catalog permissions using the LF-TBAC method \(p. 192\)](#)

See also:

- [Sharing Data Catalog tables and databases across AWS Accounts \(p. 130\)](#)
- [Metadata access control \(p. 292\)](#)
- [Lake Formation Permissions Reference \(p. 219\)](#)

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 \(p. 181\)](#)
- [Granting database permissions using the AWS CLI and the named resources method \(p. 184\)](#)
- [Granting table permissions using the Lake Formation console and the named resource method \(p. 185\)](#)
- [Granting table permissions using the AWS CLI and the named resource method \(p. 190\)](#)

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 IAM users, roles, AWS accounts, organizations, or organizational units to grant permissions.
- **LF-Tags or catalog resources** – The databases, tables, or resource links to grant permissions on.
- **Permissions** – The Lake Formation permissions to grant.

Note

To grant permissions on a database resource link, see [Granting resource link permissions \(p. 209\)](#).

Open the Grant Permissions page

1. Open the AWS Lake Formation console at <https://console.aws.amazon.com/lakeformation/>, and sign in as a data lake administrator, the database creator, or an IAM user who has **Grantable permissions** on the database.
2. Do one of the following:
 - In the navigation pane, choose **Data lake 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 \(p. 157\)](#).

Specify the principals

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

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 account, AWS organization or IAM principal outside of this account

AWS account, AWS organization, or IAM principal outside of this account

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

Choose AWS account ID or AWS organization ID

Granting data permissions to organizations is not supported when granting permissions by using LF-Tags.

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 \(p. 220\)](#).

Note

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

External accounts

For **AWS account, AWS organization, or IAM Principal** enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or ARN for the IAM user or role. Press **Enter** after each ID.

An organization ID consists of "o-" followed by 10–32 lower-case letters or digits.

An organizational unit ID starts with "ou-" followed by 4–32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

See Also

- [Accessing and viewing shared Data Catalog tables and databases \(p. 130\)](#)

Specify the databases

In the **LF-Tags or catalog resources** section, choose one or more databases to grant permissions on.

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

LF-Tags or catalog resources

Resources matched by LF-Tags (recommended)
Manage permissions indirectly for resources or data matched by a specific set of LF-Tags.

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

Databases
Select one or more databases.

Choose databases ▾ Load more

retail X

Tables - optional
Select one or more tables.

Choose tables ▾ Load more

2. Choose one or more databases from the **Database** list. You can also choose one or more Tables and/or Data filters.

Specify the permissions

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

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.

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 \(p. 174\)](#).

2. (Optional) Under **Grantable permissions**, select the permissions that the grant recipient can grant to other principals in their AWS account. This option is not supported when you are granting permissions to an IAM principal from an external account.
3. Choose **Grant**.

See Also

- [Lake Formation Permissions Reference \(p. 219\)](#)
- [Granting permissions on a database or table shared with your account \(p. 207\)](#)

Granting database permissions using the AWS CLI and the named 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-abcdefgijkl`.

```
aws lakeformation grant-permissions --principal  
  DataLakePrincipalIdentifier=arn:aws:organizations::111122223333:organization/o-  
  abcdefgijkl --permissions "ALTER" --permissions-with-grant-option "ALTER" --resource  
  '{ "Database": { "Name": "issues" } }'
```

Example - Grant to ALLIAMPrincipals in the same account

The next example grants CREATE_TABLE permission on the database `retail` to all principals in the same account. This option enables every principal in the account to create a table in the database and create a table resource link allowing integrated query engines to access shared databases and tables. This option is especially useful when a principal receives a cross-account grant, and does not have the permission to create resource links. In this scenario, the data lake administrator can create a placeholder database and grant CREATE_TABLE permission to the ALLIAMPrincipal group, enabling every IAM principal in the account to create resource links in the placeholder database.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=111122223333:IAMPrincipals --permissions "CREATE_TABLE"  
--resource '{ "Database": { "Name": "temp", "CatalogId": "111122223333" } }'
```

Example - Grant to ALLIAMPrincipals in an external account

The next example grants CREATE_TABLE on the database `retail` to all principals in an external account. This option enables every principal in the account to create a table in the database.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=111122223333:IAMPrincipals --permissions "CREATE_TABLE"  
--resource '{ "Database": { "Name": "retail", "CatalogId": "123456789012" } }'
```

Note

After granting CREATE_TABLE or ALTER on a database that has a location property that points to a registered location, be sure to also grant data location permissions on the location to the principals. For more information, see [Granting data location permissions \(p. 174\)](#).

See also

- [Lake Formation Permissions Reference \(p. 219\)](#)
- [Granting permissions on a database or table shared with your account \(p. 207\)](#)

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.

When you choose SELECT as the permission to grant, you have the option to apply a column filter, row filter, or cell filter.

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.
- **LF-Tags or catalog resources** – The databases, tables, or resource links to grant permissions on.
- **Permissions** – The Lake Formation permissions to grant.

Note

To grant permissions on a table resource link, see [Granting resource link permissions \(p. 209\)](#).

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 \(p. 157\)](#).

Specify the principals

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

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 account, AWS organization or IAM principal outside of this account

AWS account, AWS organization, or IAM principal outside of this account
Enter one or more AWS account IDs or AWS organization IDs. Press Enter after each ID.
 Choose AWS account ID or AWS organization ID

Granting data permissions to organizations is not supported when granting permissions by using LF-Tags.

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 \(p. 220\)](#).

Note

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

External accounts

For **AWS account**, **AWS organization**, or **IAM Principal** enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or the ARN for the IAM user or role. Press **Enter** after each ID.

An organization ID consists of "o-" followed by 10–32 lower-case letters or digits.

An organizational unit ID starts with "ou-" followed by 4–32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" character and 8 to 32 additional lowercase letters or digits.

See also:

- [Accessing and viewing shared Data Catalog tables and databases \(p. 130\)](#)

Specify the tables

In the **LF-Tags or catalog resources** section, choose a database. Then choose one or more tables, or **All tables**.

LF-Tags or catalog resources

Resources matched by LF-Tags (recommended)
Manage permissions indirectly for resources or data matched by a specific set of LF-Tags.

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

Databases
Select one or more databases.

retail

Tables - optional
Select one or more tables.

inventory
No description available

Specify the permissions

In the **Permissions** section, do one of the following to select permissions and grantable permissions:

- [Specify table permissions \(no data filtering\) \(p. 187\)](#)
- [Specify the Select permission with data filtering \(p. 188\)](#)

Specify table permissions (no data filtering)

1. Select the table permissions to grant, and optionally select grantable permissions.

Table and column permissions

Table permissions
Choose specific access permissions to grant.

Alter Insert Drop
 Delete Select 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.

Alter Insert Drop
 Delete Select Describe

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

If you grant **Select**, the **Data permissions** section appears beneath the **Table and column permissions** section, with the **All data access** option selected by default. Accept the default.

Data permissions

All data access
Grant access to all data without any restrictions.

Simple column-based access
Grant data access to specific columns only.

Advanced cell-level filters
Grant access to specific columns and/or rows with data filters.

2. Choose **Grant**.

Specify the **Select** permission with data filtering

1. Select the **Select** permission. Don't select any other permissions.

The **Data permissions** section appears beneath the **Table and column permissions** section.

2. Do one of the following:

- Apply simple column filtering only.

1. Choose **Simple column-based access**.

Table and column permissions

Table permissions
Choose specific access permissions to grant.

| | | |
|---------------------------------|--|-----------------------------------|
| <input type="checkbox"/> Alter | <input type="checkbox"/> Insert | <input type="checkbox"/> Drop |
| <input type="checkbox"/> Delete | <input checked="" type="checkbox"/> Select | <input type="checkbox"/> 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.

| | | |
|---------------------------------|--|-----------------------------------|
| <input type="checkbox"/> Alter | <input type="checkbox"/> Insert | <input type="checkbox"/> Drop |
| <input type="checkbox"/> Delete | <input checked="" type="checkbox"/> Select | <input type="checkbox"/> Describe |

Super

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

Data permissions

All data access
Grant access to all data without any restrictions.

Simple column-based access
Grant data access to specific columns only.

Advanced cell-level filters
Grant access to specific columns and/or rows with data filters.

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

Choose one or more columns

▾

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

Select

2. Choose whether to include or exclude columns, and then choose the columns to include or exclude.

Only include lists are supported when granting permissions to an external AWS account or organization.

3. (Optional) Under **Grantable permissions**, turn on the grant option for the Select permission.

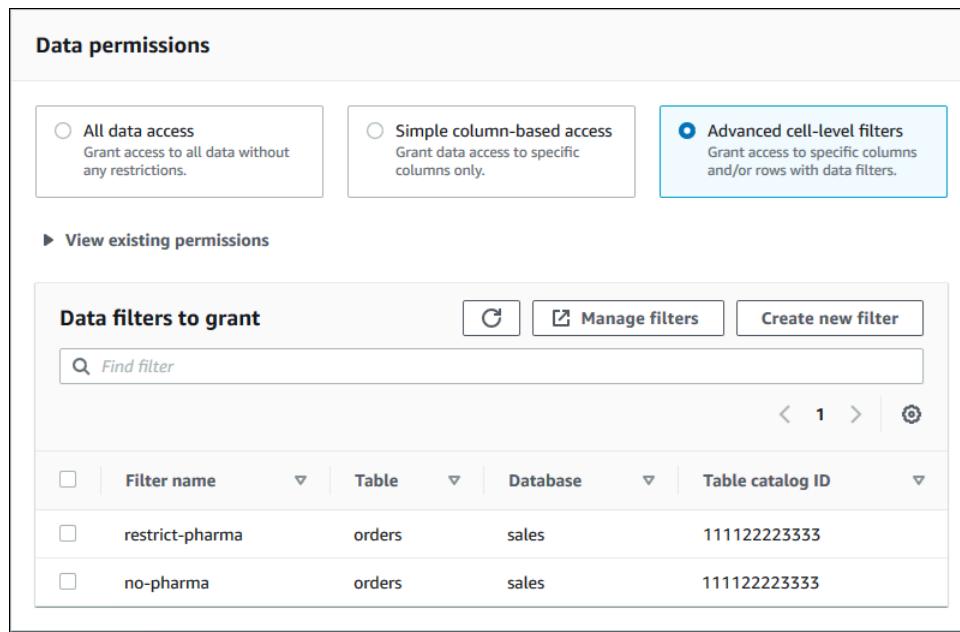
If you include the grant option, the grant recipient can grant permissions only on the columns that you grant to them.

Note

You can also apply column filtering only by creating a data filter that specifies a column filter and specifies all rows as the row filter. However, this requires more steps.

- Apply column, row, or cell filtering.

1. Choose **Advanced cell-level filters**.



2. (Optional) Expand **View existing permissions**.
3. (Optional) Choose **Create new filter**.
4. (Optional) To view details for the listed filters, or to create new or delete existing filters, choose **Manage filters**.

The **Data filters** page opens in a new browser window.

When you are finished on the **Data filters** page, return to the **Grant permissions** page, and if necessary, refresh the page to view any new data filters that you created.

5. Select one or more data filters to apply to the grant.

Note

If there are no data filters in the list, it means that no data filters were created for the selected table.

3. Choose **Grant**.

See also

- [Lake Formation access control overview \(p. 290\)](#)
- [Data filtering and cell-level security in Lake Formation \(p. 260\)](#)
- [Lake Formation Personas and IAM Permissions Reference \(p. 371\)](#)

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 - no filtering

The following example grants SELECT and ALTER to user datalake_user1 in AWS account 1111-2222-3333 on the table inventory in the database retail.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions  
"SELECT" "ALTER" --resource '{ "Table": { "DatabaseName": "retail", "Name": "inventory" } }'
```

Note

If you grant the ALTER permission on a table that has its underlying data in a registered location, be sure to also grant data location permissions on the location to the principals. For more information, see [Granting data location permissions \(p. 174\)](#).

Example – Grant on All Tables with the Grant option - no filtering

The next example grants SELECT with the grant option on all tables in database retail.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions  
"SELECT" --permissions-with-grant-option "SELECT" --resource '{ "Table": { "DatabaseName":  
"retail", "TableWildcard": {} } }'
```

Example – Grant with simple column filtering

This next example grants SELECT on a subset of columns in the table persons. It uses simple column filtering.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions  
"SELECT" --resource '{ "TableWithColumns": { "DatabaseName": "hr", "Name": "persons",  
"ColumnNames": ["family_name", "given_name", "gender"] } }'
```

Example – Grant with a data filter

This example grants SELECT on the orders table and applies the restrict-pharma data filter.

```
aws lakeformation grant-permissions --cli-input-json file://grant-params.json
```

The following are the contents of file grant-params.json.

```
{  
    "Principal": {"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/  
    datalake_user1"},  
    "Resource": {  
        "DataCellsFilter": {  
            "TableCatalogId": "111122223333",  
            "DatabaseName": "sales",  
            "TableName": "orders",  
            "Name": "restrict-pharma"  
        },  
        "Permissions": ["SELECT"],  
        "PermissionsWithGrantOption": ["SELECT"]  
    }  
}
```

}

See also

- [Lake Formation access control overview \(p. 290\)](#)
- [Data filtering and cell-level security in Lake Formation \(p. 260\)](#)
- [Lake Formation Permissions Reference \(p. 219\)](#)

Granting Data Catalog permissions using the LF-TBAC method

You can use the Lake Formation tag-based access control (LF-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 LF-TBAC Method \(p. 192\)](#)
- [Granting Data Catalog permissions using the AWS CLI and the LF-TBAC method \(p. 195\)](#)

See also

- [Granting, revoking, and listing LF-Tag permissions \(p. 254\)](#)
- [Managing LF-Tags for metadata access control \(p. 240\)](#)
- [Lake Formation Tag-based access control \(p. 231\)](#)

Granting Data Catalog permissions using the Lake Formation console and the LF-TBAC Method

The following steps explain how to grant permissions by using the Lake Formation tag-based access control (LF-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.
- **LF-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 LF-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.

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 account, AWS organization or IAM principal outside of this account

AWS account, AWS organization, or IAM principal outside of this account

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

Choose AWS account ID or AWS organization ID

Granting data permissions to organizations is not supported when granting permissions by using LF-Tags.

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 \(p. 220\)](#).

Note

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

External accounts

For **AWS accounts, AWS organization, or IAM principal** enter one or more valid AWS account IDs, organization IDs, organizational unit IDs, or ARN for the IAM user or role. Press **Enter** after each ID.

An organization ID consists of "o-" followed by 10 to 32 lower-case letters or digits.

An organizational unit ID starts with "ou-" followed by 4 to 32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

See also

- [Accessing and viewing shared Data Catalog tables and databases \(p. 130\)](#)

Specify the LF-Tags

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

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

LF-Tags or catalog resources

Resources matched by LF-Tags (recommended)
Manage permissions indirectly for resources or data matched by a specific set of LF-Tags.

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

| Key | Values |
|--|--|
| <input type="text" value="module"/> <input type="button" value="X"/> | <input type="button" value="Choose tag values"/> <input type="button" value="Remove"/> |
| | <input type="checkbox"/> Orders |
| | <input type="checkbox"/> Sales |
| | <input checked="" type="checkbox"/> Customers |

4. (Optional) Choose **Add LF-Tag** again to specify another LF-tag.

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

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 LF-tags that match one of the LF-tag expressions granted to the principal.

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

The screenshot shows the AWS Lake Formation console interface. It is divided into two main sections: 'Database permissions' and 'Table permissions'. Each section contains two columns: 'Grantable permissions' and 'Super'.

Database permissions:

- Grantable 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.

Table permissions:

- Grantable permissions:** Choose the permission that may be granted to others.
 - Alter Insert Drop
 - Delete Select Describe
- Super:** This permission is the union of all the individual permissions to the left, and supersedes them.

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 LF-TBAC method

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

To grant Data Catalog permissions using LF-TBAC (AWS CLI)

- Use the grant-permissions command.

Example

The following example grants the LF-tag expression "module=*" (all values of the LF-tag key module) to user `datalake_user1`. That user will have the `CREATE_TABLE` permission on all matching databases—databases that have been assigned the LF-tag with the key `module`, with any value.

```
aws lakeformation grant-permissions --principal  
  DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/  
  datalake_user1 --permissions "CREATE_TABLE" --resource '{ "LFTagPolicy":
```

```
{"CatalogId": "111122223333", "ResourceType": "DATABASE", "Expression": [{"TagKey": "module", "TagValues": ["*"]}]}}
```

Example

The next example grants the LF-tag expression "(level=director) AND (region=west OR region=south)" to user `datalake_user1`. That user will have the SELECT, ALTER, and DROP permissions with the grant option on matching tables—tables that have been assigned both `level=director` and `(region=west or region=south)`.

```
aws lakeformation grant-permissions --principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions "SELECT" "ALTER" "DROP" --permissions-with-grant-option "SELECT" "ALTER" "DROP" --resource '{ "LFTagPolicy": {"CatalogId": "111122223333", "ResourceType": "TABLE", "Expression": [{"TagKey": "level", "TagValues": ["director"]}, {"TagKey": "region", "TagValues": ["west", "south"]}]}'
```

Example

This next example grants the LF-tag expression "module=orders" to the AWS account 1234-5678-9012. The data lake administrator in that account can then grant the "module=orders" expression to principals in their account. Those principals will then have the CREATE_TABLE permission on matching databases owned by account 1111-2222-3333 and shared with account 1234-5678-9012 by using either the named resource method or the LF-TBAC method.

```
aws lakeformation grant-permissions --principal DataLakePrincipalIdentifier=123456789012 --permissions "CREATE_TABLE" --permissions-with-grant-option "CREATE_TABLE" --resource '{ "LFTagPolicy": {"CatalogId": "111122223333", "ResourceType": "DATABASE", "Expression": [{"TagKey": "module", "TagValues": ["orders"]}]}}'
```

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/>.

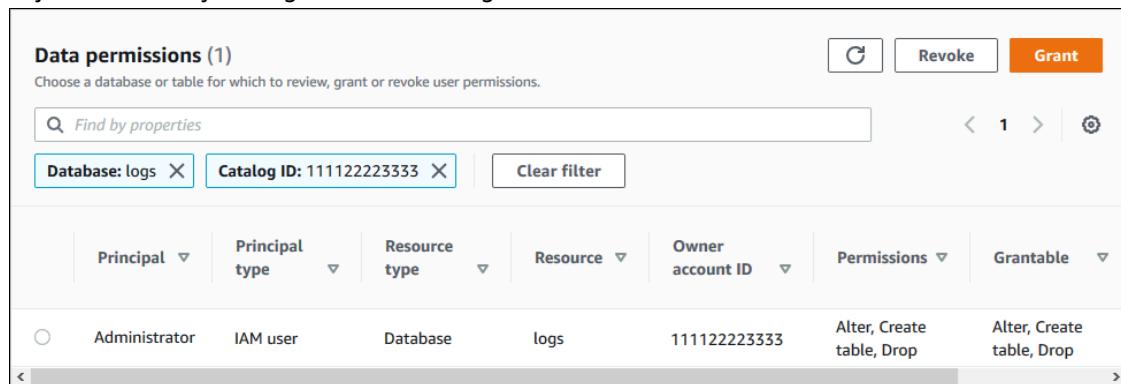
Sign in as a data lake administrator, the database creator, or as a user who has any Lake Formation permission on the database with the grant option.

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 labels under the search box. The tiles indicate that a filter has been applied to list permissions only for that database. You can adjust the filter by closing a tile or choosing **Clear filter**.



The screenshot shows the 'Data permissions' page with a single row of data. The columns are: Principal (radio button selected for Administrator), Principal type (IAM user), Resource type (Database), Resource (logs), Owner account ID (111122223333), Permissions (Alter, Create table, Drop), and Grantable (Alter, Create table, Drop). There are navigation arrows at the bottom left and right, and a 'Clear filter' button at the top right.

| Principal | Principal type | Resource type | Resource | Owner account ID | Permissions | Grantable |
|--|----------------|---------------|----------|------------------|---------------------------|---------------------------|
| <input checked="" type="radio"/> Administrator | IAM user | Database | logs | 111122223333 | Alter, Create table, Drop | Alter, Create table, Drop |

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 labels under the search box. The labels indicate that a filter has been applied to list permissions only for that table. You can adjust the filter by closing a label or choosing **Clear filter**.

The screenshot shows the 'Data permissions' page with a search bar at the top containing 'Choose a database or table for which to review, grant or revoke user permissions.' Below the search bar are three filters: 'Database: logs', 'Table: alexa-logs', and 'Catalog ID: 111122223333'. To the right of these filters are buttons for 'Clear', 'Revoke', and 'Grant'. Below the filters is a table header with columns: Principal, Principal type, Resource type, Resource, Owner account ID, Permissions, and Grantable. A single row of data is shown: Principal is 'Administrator', Principal type is 'IAM user', Resource type is 'Table', Resource is 'alexa-logs', Owner account ID is '111122223333', Permissions is 'Super', and Grantable is 'Super'.

| Data permissions (3) | | | | | | |
|---|-------------------|--------------------------|-------------------------------------|------------------|-------------|-----------|
| Choose a database or table for which to review, grant or revoke user permissions. | | | | | | |
| <input type="text"/> Find by properties | | | | | | |
| Database: logs | Table: alexa-logs | Catalog ID: 111122223333 | <input type="button"/> Clear filter | | | |
| Principal | Principal type | Resource type | Resource | Owner account ID | Permissions | Grantable |
| <input checked="" type="radio"/> 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"} }'
```

Cross-account data sharing in Lake Formation

Lake Formation cross-account capabilities allow users to securely share distributed data lakes across multiple AWS accounts, AWS organizations or directly with IAM principals in another account providing fine-grained access to the Data Catalog metadata and underlying data. Large enterprises typically use multiple AWS accounts, and many of those accounts might need access to a data lake managed by a single AWS account. Users and AWS Glue extract, transform, and load (ETL) jobs can query and join tables across multiple accounts and still take advantage of Lake Formation table-level and column-level data protections.

When you grant Lake Formation permissions on a Data Catalog resource to an external account or directly to an IAM principal in another account, Lake Formation uses the AWS Resource Access Manager (AWS RAM) service to share the resource. If the grantee account is in the same organization as the grantor account, the shared resource is available immediately to the grantee. If the grantee account is not in the same organization, AWS RAM sends an invitation to the grantee account to accept or reject the resource grant. Then, to make the shared resource available, the data lake administrator in the grantee account must use the AWS RAM console or AWS CLI to accept the invitation.

Direct cross-account share

Authorized principals can share resources explicitly with an IAM principal in an external account. This feature is useful when an account owner wants to have control over who in the external account can access the resources. The permissions the IAM principal receives will be a union of direct grants and the account level grants that is cascaded down to the principals. The data lake administrator of the recipient account can view the direct cross-account grants, but cannot revoke permissions. The principal who receives the resource share cannot share the resource with other principals.

Methods for sharing Data Catalog resources

With a single Lake Formation grant operation, you can grant cross-account permissions on the following Data Catalog resources.

- A database
- An individual table (with optional column filtering)
- A few selected tables
- All tables in a database (by using the All Tables wildcard)

There are two options for sharing your databases and tables with another AWS account or IAM principals in another account.

- Lake Formation tag-based access control (LF-TBAC) (recommended)

Lake Formation tag-based access control is an authorization strategy that defines permissions based on attributes. You can use tag-based access control to share Data Catalog resources (databases, tables, and columns) with external IAM principals, AWS accounts, Organizations and organizational units (OUs). In Lake Formation, these attributes are called LF-tags. For more information, see [Managing a data lake using Lake Formation tag-based access control](#).

Note

The LF-TBAC method of granting Data Catalog permissions use AWS Resource Access Manager for cross-account grants.

Lake Formation now supports granting cross-account permissions to Organizations and organizational units using LF-TBAC method.

To enable this capability, you need to update the **Cross account version settings to Version 3**.
For more information, see [Updating cross-account version settings \(p. 203\)](#).

- Lake Formation named resources

The Lake Formation cross-account data sharing using named resource method allows you to grant Lake Formation permissions with a grant option on Data Catalog tables and databases to external AWS accounts, IAM principals, organizations, or organizational units. The grant operation automatically shares those resources.

Note

You can also allow the AWS Glue crawler to access a data store in a different account using Lake Formation credentials. For more information, see [Cross-account crawling](#) in AWS Glue Developer Guide.

Integrated services such as Athena and Amazon Redshift Spectrum require resource links to be able to include shared resources in queries. For more information about resource links, see [How resource links work in Lake Formation \(p. 157\)](#).

Topics

- [Cross-account data sharing prerequisites \(p. 200\)](#)
- [Updating cross-account version settings \(p. 203\)](#)
- [Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts \(p. 206\)](#)
- [Granting permissions on a database or table shared with your account \(p. 207\)](#)
- [Granting resource link permissions \(p. 209\)](#)
- [Cross-account best practices and limitations \(p. 210\)](#)
- [Accessing the underlying data of a shared table \(p. 212\)](#)
- [Cross-account CloudTrail logging \(p. 213\)](#)
- [Managing cross-account permissions using both AWS Glue and Lake Formation \(p. 216\)](#)
- [Viewing all cross-account grants using the GetResourceShares API operation \(p. 218\)](#)

Related topics

- [Overview of Lake Formation permissions \(p. 171\)](#)
- [Accessing and viewing shared Data Catalog tables and databases \(p. 130\)](#)
- [Creating resource links \(p. 157\)](#)
- [Troubleshooting cross-account access \(p. 378\)](#)

Cross-account data sharing prerequisites

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

- To share resources directly with an IAM principal in another account, you need to update the **Cross account version settings to Version 3**. This setting is available on the [Data catalog settings](#) page. If you are using **Version 1**, see instructions to update the setting [Updating cross-account version settings \(p. 203\)](#).

Version 3 of the Cross account version settings is also required for integrating AWS RAM resource sharing when using TABC method.

- Before granting cross-account permissions on a Data Catalog resource, you must revoke all Lake Formation permissions from the IAMAllowedPrincipals group for the resource. If the calling principal has cross account permissions to access a resource and IAMAllowedPrincipals permission exist on the resource, then Lake Formation throws AccessDeniedException.
- For databases that contain tables that you intend to share, you must prevent new tables from having a default grant of Super to IAMAllowedPrincipals. On the Lake Formation console, edit the database and turn off **Use only IAM access control for new tables in this database** or enter the following AWS CLI command, replacing database with the name of the database.

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

- You cannot share Data Catalog resources encrypted with AWS Glue service managed key with another account. You can share only Data Catalog resources encrypted with customer's encryption key, and the account receiving the resource share must have permissions on the Data Catalog encryption key to decrypt the objects.

Data sharing using LF-TBAC prerequisites

- Add the following JSON permissions object to your AWS Glue Data Catalog resource policy if you are using **Cross account version settings Version 1** or **Version 2**. This policy is required for every AWS account to which you are granting permissions.

Add the `glue:PutResourcePolicy` either by using the **Settings** page on the AWS Glue console or the following AWS Command Line Interface (AWS CLI) command:

```
aws glue put-resource-policy --policy-in-json file://policy.json --enable-hybrid TRUE
```

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 \(p. 216\)](#).

Data sharing using AWS Resource Access Manager prerequisites

- 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 Lake Formation tag-based access control (LF-TBAC) method, you must have a Data Catalog resource policy that enables LF-TBAC. For more information, see [Managing cross-account permissions using both AWS Glue and Lake Formation \(p. 216\)](#).
- If you want 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.

- Users who want to use AWS RAM resource sharing to grant cross-account permissions must have the required AWS Identity and Access Management (IAM) permissions on AWS Glue and AWS RAM service. The AWS managed policy `AWSLakeFormationCrossAccountManager` grants the required permissions.

Data lake administrators in accounts that receive resources that are 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": "*"  
        }  
    ]  
}
```

Setup required in each account that accesses the shared resource

- If you are sharing resources with AWS accounts, at least one user in the consumer account must be a data lake administrator to view shared resources. For information on how to create a data lake administrator, see [Create a data lake administrator \(p. 12\)](#).

The data lake administrator can grant Lake Formation permissions on the shared resources to other principals in the account. Other principals can't access shared resources until the data lake administrator grants them permissions on the resources.

- The account receiving the cross-account share must have the `glue:PutResourcePolicy` permission to accept the AWS RAM resource share invitation.
- Integrated services such as Athena and Redshift Spectrum require resource links to be able to include shared resources in queries. Principals need to create a resource link in their Data Catalog to a shared resource from another AWS account. For more information about resource links, see [How resource links work in Lake Formation \(p. 157\)](#).
- When a resource is shared directly with an IAM principal, to query the table using Athena, the data lake administrator of the recipient account or an authorized principal with required permissions needs to create a resource link and grant the principal **Describe** permission on the link. The principal who received the direct cross-account share will not be able to create the resource link.

If the producer account shares a different table under the same database with the same or another principal, that principal can immediately query the table.

Important

To create a resource link, you need the Lake Formation `CREATE_TABLE` or `CREATE_DATABASE` permission, and the `glue:CreateTable` or `glue>CreateDatabase` IAM permission.

Note

For the data lake administrator and for principals whom the data lake administrator has granted permissions to, shared resources appear in the Data Catalog as if they are local (owned) resources. Extract, transform, and load (ETL) jobs can access the underlying data of shared resources.

For shared resources, the **Tables** and **Databases** pages on the Lake Formation console display the owner's account ID.

When the underlying data of a shared resource is accessed, CloudTrail log events are generated in both the shared resource recipient's account and the resource owner's account. The CloudTrail events can contain the ARN of the principal that accessed the data, but only if the recipient account opts in to include the principal ARN in the logs. For more information, see [Cross-account CloudTrail logging \(p. 213\)](#).

Updating cross-account version settings

From time to time, AWS Lake Formation updates the cross-account settings to distinguish the changes made to the AWS RAM usage. When Lake Formation does this, it creates a new version of the **Cross-account version settings**.

Optimize AWS RAM resource shares

AWS Lake Formation now offers a new version of cross-account grant that optimally utilizes AWS RAM capacity to maximize cross account usage. When you share a resource with an external AWS account or an IAM principal, Lake Formation may create a new resource share or associate the resource with an existing share. By associating with existing shares, Lake Formation reduces the number of resource share invitations a consumer needs to accept.

Enable AWS RAM shares via TBAC or share resources directly to principals

To share resources directly with IAM principals in another account or to enable TBAC cross-account shares to Organizations or organizational units, you need to update the [Cross account version settings](#) to Version 3. For more information about AWS RAM resource limits, see [Cross-account best practices and limitations \(p. 210\)](#).

Required permissions for updating cross-account version settings

If a cross-account permission grantor has `AWSLakeFormationCrossAccountManager` managed IAM policy permissions, then there is no extra permission setup required for the cross-account permission grantor role or principal. However, if the cross-account grantor is not using the managed policy, then the grantor role or principal should have following IAM permissions granted for the new version of the cross-account grant to be successful.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor1",  
            "Effect": "Allow",  
            "Action": [  
                "ram:AssociateResourceShare",  
                "ram:DisassociateResourceShare",  
                "ram:GetResourceShares"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringLike": {  
                    "ram:ResourceShareName": "LakeFormation*"  
                }  
            }  
        }  
    ]  
}
```

To enable the new version

Follow these steps to update **Cross account versions settings** through the AWS Lake Formation console or the AWS CLI.

Console

1. Choose **Version 2** or **Version 3** under **Cross account version settings** on the **Data catalog settings** page. If you select **Version 1**, Lake Formation will use the default resource sharing mode.

The screenshot shows the 'Data catalog settings' page in the AWS Lake Formation console. It includes sections for 'Default permissions for newly created databases and tables' (with two checkboxes for IAM access control), 'Default permissions for AWS CloudTrail' (with a search bar for resource owners), and 'Cross account version settings' (with a dropdown menu showing Version 1, Version 2, Version 3, and Version 4). At the bottom right are 'Cancel' and 'Save' buttons.

2. Choose **Save**.

AWS Command Line Interface (AWS CLI)

Use the `put-data-lake-settings` AWS CLI command to set the `CROSS_ACCOUNT_VERSION` parameter. Accepted values are 1, 2, and 3.

```
aws lakeformation put-data-lake-settings --region us-east-1 --data-lake-settings
  file://settings
{
    "DataLakeAdmins": [
        {
            "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/test"
        }
    ],
    "CreateDatabaseDefaultPermissions": [],
    "CreateTableDefaultPermissions": [],
    "Parameters": {
        "CROSS_ACCOUNT_VERSION": "3"
    }
}
```

Important

Once you choose **Version 2** or **Version 3**, all new grants will go through the new cross-account grant mode. To optimally use AWS RAM capacity for your existing cross-account shares, we recommend you to revoke the grants that were made with the older version, and re-grant in the new mode.

Sharing Data Catalog tables and databases across AWS accounts or IAM principals from external accounts

This section includes instructions on how to enable cross-account permissions on Data Catalog tables and databases to an external AWS account, IAM principal, organization, or organizational unit. The grant operation automatically shares those resources.

Topics

- [Data sharing using tag-based access control \(p. 206\)](#)
- [Data sharing using the named resource method \(p. 207\)](#)

Data sharing using tag-based access control

Set up required on the producer/grantor account

1. Define an LF tag. For instructions to create an LF-tag, see [Creating LF-Tags \(p. 241\)](#).
2. Assign the LF-tag to the target resource. For more information, see [Assigning LF-Tags to Data Catalog resources \(p. 246\)](#).
3. Grant LF-tag permission to the external account. For more information, see [Granting LF-Tag permissions using the console \(p. 255\)](#).

At this point, the consumer data lake administrator should be able to find the policy tag being shared via the grantee account Lake Formation console, under **Permissions, Administrative roles and tasks, LF-tags**.

4. Grant data permission to the external/grantee account.
 - a. In the navigation pane, under **Permissions, Data lake permissions**, choose **Grant**.
 - b. For **Principals**, choose **External accounts**, and enter the target AWS account ID or the IAM role of the principal or the Amazon Resource Name (ARN) for the principal (principal ARN).
 - c. For **LF-tags or catalog resources**, choose the **key** and **values** of the **LF-tag** that is being shared with the consumer account (**key** Confidentiality and **value** public).
 - d. For **Permissions**, under **Resources matched by LF-tags (recommended)** choose **Add LF-tag**.
 - e. Select the **key** and **value** of the tag that is being shared with the grantee account (**key** Confidentiality and **value** public).
 - f. For **Database permissions**, select **Describe** under **Database permissions** to grant access permissions at the database level.
 - g. The consumer data lake administrator should be able to find the policy tag being shared via the consumer account on the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>, under **Permissions, Administrative roles and tasks, LF-tags**.
 - h. Select **Describe** under **Grantable permissions** so the consumer account can grant database-level permissions to its users.

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.

Note

Principals who receive direct cross-account grants will not have the **Grantable permissions** option.

- i. For **Table and column permissions**, select **Select** and **Describe** under **Table permissions**.
- j. Select **Select** and **Describe** under **Grantable permissions**.
- k. Choose **Grant**.

Set up required on the receiving/grantee account

1. When you share a resource with another account, the resource still belongs to the producer account and is not visible within the Athena console. To make the resource visible in the Athena console, you need to create a resource link pointing to the shared resource. For instructions on creating a resource link, see [Creating a resource link to a shared Data Catalog table \(p. 159\)](#) and [Creating a resource link to a shared Data Catalog database \(p. 161\)](#)
2. You need to create a separate set of LF-tags in the consumer account to use LF tag-based access control when sharing the resource links. Create and assign the required LF-tags to the shared database/tables and the resource links.
3. Grant permissions on these LF-tags to the IAM principals in the grantee account.

Data sharing using the named resource method

You can grant permissions to directly to principals in the another AWS account, or to external AWS accounts or AWS Organizations. Granting Lake Formation permissions to Organizations or organizational units is equivalent to granting the permission to every AWS account in that organization or organizational unit.

When you grant permissions to external accounts or organizations, you must include the **Grantable permissions** option. Only the data lake administrator in the external account can access the shared resources until the administrator grants permissions on the shared resources to other principals in the external account.

Note

Grantable permissions option is not supported when granting permissions directly to IAM principals from external accounts.

Follow instructions in [Granting database permissions using the Lake Formation console and the named resource method \(p. 181\)](#) to grant cross-account permissions using the named resource method.

Granting permissions on a database or table shared with your account

After a Data Catalog resource belonging to another AWS account is shared with your AWS account, as a data lake administrator, you can grant permissions on the shared resource to other principals in your account. You can't, however, grant permissions on the resource to other AWS accounts or organizations.

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

To grant permissions on a shared database (named resource method, console)

- Follow the instructions in [Granting database permissions using the Lake Formation console and the named resource method \(p. 181\)](#). In the **Database** list under **LF-Tags or catalog resources**, ensure that you select the database in the external account, not a resource link for the database.

If you don't see the database in the list of databases, ensure that you have accepted the AWS Resource Access Manager (AWS RAM) resource share invitation for the database. For more information, see [Accepting a resource share invitation from AWS RAM \(p. 154\)](#).

Also, for the CREATE_TABLE and ALTER permissions, follow the instructions in [Granting data location permissions \(same account\) \(p. 174\)](#), 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 \(p. 185\)](#). In the **Database** list under **LF-Tags or catalog resources**, ensure that you select the database in the external account, not a resource link for the database.

If you don't see the table in the list of tables, ensure that you have accepted the AWS RAM resource share invitation for the table. For more information, see [Accepting a resource share invitation from AWS RAM \(p. 154\)](#).

Also, for the ALTER permission, follow the instructions in [Granting data location permissions \(same account\) \(p. 174\)](#), and be sure to enter the owning account ID in the **Registered account location** field.

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

- Follow the instructions in [Granting Data Catalog permissions using the Lake Formation console and the LF-TBAC Method \(p. 192\)](#). In the **LF-Tags or catalog resources** section, grant the exact LF-tag expression that the external account granted to your account, or a subset of that expression.

For example, if an external account granted the LF-tag expression module=customers AND environment=production to your account with the grant option, as a data lake administrator, you can grant that same expression, or module=customers or environment=production to a principal in your account. You can grant only the same or a subset of the Lake Formation permissions (for example, SELECT, ALTER, and so on) that were granted on resources through the LF-tag expression.

To grant permissions on a shared table (named resource method, AWS CLI)

- Enter a command similar to the following. In this example:
 - Your AWS account ID is 1111-2222-3333.
 - The account that owns the table and that granted it to your account is 1234-5678-9012.
 - The SELECT permission is being granted on the shared table pageviews to user datalake_user1. That user is a principal in your account.
 - The pageviews table is in the analytics database, which is owned by account 1234-5678-9012.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
--permissions "SELECT" --resource '{ "Table": { "CatalogId": "123456789012",
"DatabaseName": "analytics", "Name": "pageviews" }}'
```

Note that the owning account must be specified in the CatalogId property in the **resource** argument.

Granting resource link permissions

Follow these steps to grant AWS Lake Formation permissions on one or more resource links to a principal in your AWS account.

After you create a resource link, only you can view and access it. (This assumes that **Use only IAM access control for new tables in this database** is not enabled for the database.) To permit other principals in your account to access the resource link, grant at least the DESCRIBE permission.

Important

Granting permissions on a resource link doesn't grant permissions on the target (linked) database or table. You must grant permissions on the target separately.

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

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 \(p. 181\)](#) to do the following:
 1. Open the **Grant Permissions** page ([p. 181](#)).
 2. [Specify the databases \(p. 182\)](#). Specify one or more database resource links.
 3. [Specify the principals \(p. 181\)](#).
 - For table resource links, follow the steps in [Granting table permissions using the Lake Formation console and the named resource method \(p. 185\)](#) to do the following:
 1. [Open the Grant permissions page \(p. 186\)](#).
 2. [Specify the tables \(p. 187\)](#). Specify one or more table resource links.
 3. [Specify the principals \(p. 186\)](#).
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 \(p. 157\)](#)
- [Lake Formation Permissions Reference \(p. 219\)](#)

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, Lake Formation uses AWS Resource Access Manager (AWS RAM)

capacity for cross-account grants that your account can make with the named resource method. To maximize the AWS RAM capacity, follow these best practices for the named resource method:

- Use the new cross-account grant mode (**Version 3** under **Cross account version settings**) to share a resource with an external AWS account. For more information, see [Updating cross-account version settings \(p. 203\)](#).
- 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 \(p. 130\)](#).

- 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 permissions on Data Catalog resources \(p. 180\)](#).

Note

For more information about requesting a higher limit for the number of resource shares in AWS RAM, see [AWS service quotas](#) in the *AWS General Reference*.

- You must create a resource link to a shared database for that database to appear in the Amazon Athena and Amazon Redshift Spectrum query editors. Similarly, to be able to query shared tables using Athena and Redshift Spectrum, you must create resource links to the tables. The resource links then appear in the tables list of the query editors.

Instead of creating resource links for many individual tables for querying, you can use the **All tables** wildcard to grant permissions on all tables in a database. Then, when you create a resource link for that database and select that database resource link in the query editor, you'll have access to all tables in that database for your query. For more information, see [Creating resource links \(p. 157\)](#).

- When you share resources directly with principals in another account, the IAM principal in the recipient account may not have permission to create resource links to be able to query the shared tables using Athena and Amazon Redshift Spectrum. Instead of creating a resource link for each table that is shared, the data lake administrator can create a placeholder database and grant `CREATE_TABLE` permission to the `ALLIAMPPrincipal` group. Then, all IAM principals in the recipient account can create resource links in the placeholder database and start querying the shared tables.

See the example CLI command for granting permissions to `ALLIAMPPrincipal`s in [Granting database permissions using the AWS CLI and the named resources method \(p. 184\)](#).

- Athena and Redshift Spectrum support column-level access control, but only for inclusion, not exclusion. Column-level access control is not supported in AWS Glue ETL jobs.
- When a resource is shared with your AWS account, you can grant permissions on the resource only to users in your account. You can't grant permissions on the resource to other AWS accounts, to organizations (not even your own organization), or to the `IAMAllowedPrincipals` group.
- You can't grant `DROP` or `Super` on a database to an external account.
- Revoke cross-account permissions before you delete a database or table. Otherwise, you must delete orphaned resource shares in AWS Resource Access Manager.

See Also

- [Lake Formation Tag-based access control notes and restrictions \(p. 239\)](#)
- [CREATE_TABLE \(p. 224\)](#) in the [Lake Formation Permissions Reference \(p. 219\)](#) 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 Lake Formation tag-based access control method.)
- The data lake administrator must re-grant to the principal the Lake Formation SELECT permission that account A granted on the shared table.
- The principal must have the following IAM permissions on the table, the database that contains it, and the account A Data Catalog.

Note

In the following IAM policy:

- Replace `<account-id-A>` with the AWS account ID of account A.
- Replace `<region>` with a valid Region.
- Replace `<database>` with the name of the database in account A that contains the shared table.
- Replace `<table>` with the name of the shared table.

```
{
    "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:

- [Accepting a resource share invitation from AWS RAM \(p. 154\)](#)

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 \(p. 213\)](#)
- [Querying CloudTrail logs for Amazon S3 cross-account access \(p. 214\)](#)

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": "AROAQGFTBBBG0BWV2EMZA: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": "AROAQGFTBBBG0BWV2EMZA:GlueJobRunnerSession",  
        "accountId": "111122223333"  
    },  
    "eventSource": "lakeformation.amazonaws.com",  
    "eventName": "GetDataAccess",  
    ...  
    ...  
    "additionalEventData": {  
        "requesterService": "GLUE_JOB",  
        "lakeFormationPrincipal": "arn:aws:iam::111122223333:role/ETL-Glue-Role",  
        "lakeFormationRoleSessionName": "AWSLF-00-GL-111122223333-G13T0Rmng2"  
    },  
    ...  
}
```

To opt in to include principal ARNs in cross-account CloudTrail logs

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
Sign in as the Administrator user, or a user with the Administrator Access IAM policy.
2. In the navigation pane, choose **Settings**.
3. On the **Data catalog settings** page, in the **Default permissions for AWS CloudTrail** section, for **Resource owners**, enter one or more AWS resource owner account IDs.
Press **Enter** after each account ID.
4. Choose **Save**.

Now cross-account CloudTrail events stored in the logs for both the shared resource recipient and the resource owner contain the principal ARN.

Querying CloudTrail logs for Amazon S3 cross-account access

As a shared resource owner, you can query S3 CloudTrail logs to determine the accounts that have accessed your Amazon S3 buckets (provided that you enabled object-level logging in Amazon S3). This applies only to S3 locations that you registered with Lake Formation. If shared resource consumers opt in to include principal ARNs in Lake Formation CloudTrail logs, you can determine the roles or users that accessed the buckets.

When running queries with Amazon Athena, you can join Lake Formation CloudTrail events and S3 CloudTrail events on the session name property. Queries can also filter Lake Formation events on `eventName="GetDataAccess"`, and S3 events on `eventName="Get Object"` or `eventName="Put Object"`.

The following is an excerpt from a Lake Formation cross-account CloudTrail event where data in a registered S3 location was accessed.

```
{
  "eventSource": "lakeformation.amazonaws.com",
  "eventName": "GetDataAccess",
  .....
  .....
  "additionalEventData": {
    "requesterService": "GLUE_JOB",
    "lakeFormationPrincipal": "arn:aws:iam::111122223333:role/ETL-Glue-Role",
    "lakeFormationRoleSessionName": "AWSLF-00-GL-111122223333-B8JSAjo5QA"
  }
}
```

The `lakeFormationRoleSessionName` key value, `AWSLF-00-GL-111122223333-B8JSAjo5QA`, can be joined with the session name in the `principalId` key of the S3 CloudTrail event. The following is an excerpt from the S3 CloudTrail event. It shows the location of the session name.

```
{
  "eventSource": "s3.amazonaws.com",
  "eventName": "Get Object"
  .....
  .....
  "principalId": "AROAQSOX5XXUR7D6RMYLR:AWSLF-00-GL-111122223333-B8JSAjo5QA",
  "arn": "arn:aws:s3:::111122223333:assumed-role/Deformationally/AWSLF-00-
GL-111122223333-B8JSAjo5QA",
  "session Context": {
    "session Issuer": {
      "type": "Role",
      "principalId": "AROAQSOX5XXUR7D6RMYLR",
      "arn": "arn:aws:iam::111122223333:role/aws-service-role/lakeformation.amazonaws.com/
Deformationally",
      "accountId": "111122223333",
      "user Name": "Deformationally"
    },
    .....
  }
}
```

The session name is formatted as follows:

```
AWSLF-<version-number>-<query-engine-code>-<account-id>-<suffix>
```

version-number

The version of this format, currently `00`. If the session name format changes, the next version will be `01`.

query-engine-code

Indicates the entity that accessed the data. Current values are:

| | |
|----|--------------------------|
| GL | AWS Glue ETL job |
| AT | Athena |
| RE | Amazon Redshift Spectrum |

account-id

The AWS account ID that requested credentials from Lake Formation.

suffix

A randomly generated string.

Managing cross-account permissions using both AWS Glue and Lake Formation

It's possible to grant cross-account access to Data Catalog resources and underlying data by using either AWS Glue or AWS Lake Formation.

In AWS Glue you grant cross-account permissions by creating or updating a Data Catalog resource policy. In Lake Formation, you grant cross-account permissions by using the Lake Formation GRANT/REVOKE permissions model and the `Grant Permissions` API operation.

Tip

We recommend that rely solely on Lake Formation permissions to secure your data lake.

You can view Lake Formation cross-account grants by using the Lake Formation console or, 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` field 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": "Cataloguers",
        "Effect": "Allow",
        "Action": [
            "glue:*"
        ],
        "Principal": {
            "AWS": CONSUMER_ACCOUNT_IDS
        },
        "Resource": [
            "arn:aws:glue:{REGION}:{PRODUCER_ACCOUNT_ID}:catalog",
            "arn:aws:glue:{REGION}:{PRODUCER_ACCOUNT_ID}:database/*",
            "arn:aws:glue:{REGION}:{PRODUCER_ACCOUNT_ID}:table/*/*"
        ]
    }
]
}

policy = json.dumps(policy)
glue.put_resource_policy(PolicyInJson=policy, 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

When using the Lake Formation tag-based access control (LF-TBAC) method to make cross-account grants, if the **Cross account version settings** is **Version 1** or **Version 2** you must have a Data Catalog resource policy with at least the permissions specified in [Cross-account data sharing prerequisites \(p. 200\)](#).

See Also:

- [Metadata access control \(p. 292\)](#) (for a discussion of the named resource method versus the Lake Formation tag-based access control (LF-TBAC) method).
- [Viewing shared Data Catalog tables and databases \(p. 156\)](#)

- [Working with Data Catalog Settings on the AWS Glue Console](#) in the *AWS Glue Developer Guide*
- [Granting Cross-Account Access](#) in the *AWS Glue Developer Guide* (for sample Data Catalog resource policies)

Viewing all cross-account grants using the GetResourceShares API operation

If your enterprise grants cross-account permissions using both an AWS Glue Data Catalog resource policy and Lake Formation grants, the only way to view all cross-account grants in one place is to use the `glue:GetResourceShares` API operation.

When you grant Lake Formation permissions across accounts by using the named resource method, AWS Resource Access Manager (AWS RAM) creates an AWS Identity and Access Management (IAM) resource policy and stores it in your AWS account. The policy grants the permissions required to access the resource. AWS RAM creates a separate resource policy for each cross-account grant. You can view all of these policies by using the `glue:GetResourceShares` API operation.

Note

This operation also returns the Data Catalog resource policy. However, if you enabled meta data encryption in Data Catalog settings, and you don't have permission on the AWS KMS key, the operation won't return the Data Catalog resource policy.

To view all cross-account grants

- Enter the following AWS CLI command.

```
aws glue get-resource-policies
```

The following is an example resource policy that AWS RAM creates and stores when you grant permissions on table `t` in database `db1` to AWS account `1111-2222-3333`.

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

See Also:

- [GetResourceShares Action \(Python: get_resource_policies\)](#) in the *AWS Glue Developer Guide*

Revoking permission using the Lake Formation console

You can use the console to revoke all types of Lake Formation permissions—Data Catalog permissions, policy tag permissions, data filter permissions, and location permissions.

To revoke Lake Formation permissions on a resource (console)

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
Sign in as a data lake administrator or as a user who has been granted permissions with the grant option on the resource.
2. In the navigation pane, under **Permissions**, choose **Tag permissions**, **Data permissions**, or **Data locations**.
3. Select the permission or location, and then choose **Revoke**.
4. In the dialog box that opens, choose **Revoke**.

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” \(p. 290\)](#). 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” \(p. 180\)](#) and [the section called “Granting data location permissions” \(p. 174\)](#).

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 “Overview of Lake Formation permissions” \(p. 171\)](#).

Topics

- [Lake Formation Grant and Revoke AWS CLI Commands \(p. 220\)](#)
- [ALTER \(p. 222\)](#)
- [CREATE_DATABASE \(p. 223\)](#)
- [CREATE_TABLE \(p. 224\)](#)
- [DATA_LOCATION_ACCESS \(p. 225\)](#)
- [DELETE \(p. 225\)](#)
- [DESCRIBE \(p. 226\)](#)
- [DROP \(p. 227\)](#)
- [INSERT \(p. 227\)](#)
- [SELECT \(p. 228\)](#)
- [Super \(p. 229\)](#)

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/abcdefhijkl
```

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/abcdefhijkl/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. | CREATE_DATABASE is sufficient. |

| Create Database Use Case | Permissions Needed |
|--|---|
| The location property is specified, and the location is not managed by Lake Formation (is not registered). | CREATE_DATABASE is sufficient. |
| The location property is specified, and the location is managed by Lake Formation (is registered). | CREATE_DATABASE is required plus data location permissions on the specified location. |

CREATE_TABLE

| Permission | Granted on This Resource | Grantee Also Needs |
|--------------|--------------------------|--------------------|
| CREATE_TABLE | DATABASE | glue:CreateTable |

A principal with this permission can create a metadata table or resource link in the Data Catalog within the specified database.

Example

The following example grants the user `datalake_user1` permission to create tables in the `retail` database in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1
--permissions "CREATE_TABLE" --resource '{ "Database": { "Name": "retail" }}'
```

When a principal creates a table in the Data Catalog, all Lake Formation permissions on the table are granted to the principal, with the ability to grant these permissions to others.

Cross-Account Grants

If a database owner account grants CREATE_TABLE to a recipient account, and a user in the recipient account successfully creates a table in the owner account's database, the following rules apply:

- The user and data lake administrators in the recipient account have all Lake Formation permissions on the table. They can grant permissions on the table to other principals in their account. They can't grant permissions to principals in the owner account or any other accounts.
- Data lake administrators in the owner account can grant permissions on the table to other principals in their account.

Data Location Permissions

When you attempt to create a table that points to an Amazon S3 location, depending on whether you have data location permissions, the CREATE_TABLE permission might not be sufficient to create a table. It's important to keep the following three cases in mind.

| Create Table Use Case | Permissions Needed |
|--|-----------------------------|
| The specified location is not managed by Lake Formation (is not registered). | CREATE_TABLE is sufficient. |

| 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 \(p. 295\)](#).

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 | Table resource link | glue:GetTable |
| | Database resource link | glue:GetDatabase |
| DESCRIBE | DATABASE | glue:GetDatabase |
| DESCRIBE | TABLE | glue:GetTable |

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 | Database resource link | glue:DeleteDatabase |
| | Table resource link | glue:DeleteTable |

A principal with this permission can drop a database, table, or resource link in the Data Catalog. You can't grant DROP on a database to an external account or organization.

Warning

Dropping a database drops all tables in the database.

Example

The following example grants the DROP permission to the user `datalake_user1` on the database `retail` in AWS account `1111-2222-3333`.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions
"DROP" --resource '{ "Database": { "Name": "retail" }}'
```

Example

The following example grants DROP to the user `datalake_user1` on the table `inventory` in the database `retail`.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions
"DROP" --resource '{ "Table": { "DatabaseName": "retail", "Name": "inventory" }}'
```

Example

The following example grants DROP to the user `datalake_user1` on the table resource link `inventory-link` in the database `retail`.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions
"DROP" --resource '{ "Table": { "DatabaseName": "retail", "Name": "inventory-link" }}'
```

INSERT

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

A principal with this permission can insert, update, and read underlying data at the Amazon S3 location specified by the table. The principal can also view the table in the Lake Formation console and retrieve information about the table with the AWS Glue API.

Example

The following example grants the INSERT permission to the user `datalake_user1` on the table `inventory` in the database `retail` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions
"INSERT" --resource '{ "Table": { "DatabaseName": "retail", "Name": "inventory" }}'
```

This permission applies only to data in Amazon S3, and not to data in other data stores such as Amazon RDS.

SELECT

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

A principal with this permission can view a table in the Data Catalog, and can query the underlying data in Amazon S3 at the location specified by the table. The principal can view the table in the Lake Formation console and retrieve information about the table with the AWS Glue API. If column filtering was applied when this permission was granted, the principal can view the metadata only for the included columns and can query data only from the included columns.

Note

It is the responsibility of the integrated analytics service to apply the column filtering when processing a query.

Example

The following example grants the SELECT permission to the user `datalake_user1` on the table `inventory` in the database `retail` in AWS account 1111-2222-3333.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions
"SELECT" --resource '{ "Table": { "DatabaseName": "retail", "Name": "inventory" }}'
```

This permission applies only to data in Amazon S3, and not to data in other data stores such as Amazon RDS.

You can filter (restrict the access to) specific columns with an optional inclusion list or an exclusion list. An inclusion list specifies the columns that can be accessed. An exclusion list specifies the columns that can't be accessed. In the absence of an inclusion or exclusion list, all table columns are accessible.

The results of `glue:GetTable` return only the columns that the caller has permission to view. Integrated services such as Amazon Athena and Amazon Redshift honor column inclusion and exclusion lists.

Example

The following example grants SELECT to the user `datalake_user1` on the table `inventory` using an inclusion list.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions
"SELECT" --resource '{ "TableWithColumns": { "DatabaseName": "retail", "Name": "inventory",
"ColumnNames": [ "prodcode", "location", "period", "withdrawals" ] } }'
```

Example

This next example grants SELECT on the inventory table using an exclusion list.

```
aws lakeformation grant-permissions --principal
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --permissions
"SELECT" --resource '{ "TableWithColumns": { "DatabaseName": "retail", "Name": "inventory",
"ColumnWildcard": { "ExcludedColumnNames": [ "intkey", "prodcode" ] } } }'
```

The following restrictions apply to the SELECT permission:

- When granting SELECT, you can't include the grant option if column filtering is applied.
- You cannot restrict access control on columns that are partition keys.
- A principal with the SELECT permission on a subset of columns in a table cannot be granted the ALTER, DROP, DELETE, or INSERT permission on that table. Similarly, a principal with the ALTER, DROP, DELETE, or INSERT permission on a table cannot be granted the SELECT permission with column filtering.

The SELECT permission always appears on the **Data permissions** page of the Lake Formation console as a separate row. This following image shows that SELECT is granted to the users `datalake_user2` and `datalake_user3` on all columns in the `inventory` table.

| Data permissions (8) | | | | | |
|---|----------------|---------------|--------------------|------------------|----------------|
| Choose a database or table for which to review, grant or revoke user permissions. | | | | | |
| <input type="button" value="Find by properties"/> <input type="button" value="Revoke"/> <input type="button" value="Grant"/> | | | | | |
| <input type="button" value="Database: retail"/> <input type="button" value="Table: inventory"/> <input type="button" value="Clear filter"/> | | | | | |
| Principal | Principal type | Resource type | Resource | Owner account ID | Permissions |
| <input type="radio"/> <code>datalake_user3</code> | IAM user | Table | inventory | 111122223333 | Insert |
| <input type="radio"/> <code>datalake_user3</code> | IAM user | Column | retail.inventory.* | 111122223333 | Select |
| <input type="radio"/> <code>datalake_user2</code> | AD user | Table | inventory | 111122223333 | Delete, Insert |
| <input type="radio"/> <code>datalake_user2</code> | 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.

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

- 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. 299)
- [Upgrading AWS Glue data permissions to the Lake Formation model](#) (p. 23)

Lake Formation Tag-based access control

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

Topics

- [Overview of Lake Formation tag-based access control \(p. 231\)](#)
- [How Lake Formation tag-based access control works \(p. 232\)](#)
- [Lake Formation tag-based access control permissions model \(p. 237\)](#)
- [Lake Formation Tag-based access control notes and restrictions \(p. 239\)](#)
- [Managing LF-Tags for metadata access control \(p. 240\)](#)
- [Granting, revoking, and listing LF-Tag permissions \(p. 254\)](#)

Overview of Lake Formation tag-based access control

Lake Formation tag-based access control (LF-TBAC) works with IAM's attribute-based access control (ABAC) to provide fine-grained access to your data lake resources and data.

Note

IAM tags are not the same as LF-tags. These tags are not interchangeable. LF-tags are used to grant Lake Formation permissions and IAM tags are used to define IAM policies.

What is Lake Formation tag-based access control?

Lake Formation tag-based access control (LF-TBAC) is an authorization strategy that defines permissions based on attributes. In Lake Formation, these attributes are called *LF-tags*. You can attach LF-tags to Data Catalog resources, Lake Formation principals, and table columns. You can assign and revoke permissions on Lake Formation resources using these LF-tags. Lake Formation allows operations on those resources when the principal's tag matches the resource tag. LF-TBAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

Comparison of Lake Formation tag-based access control to IAM attribute-based access control

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM resources, including IAM entities (users or roles) and to AWS resources. You can create a single ABAC policy or small set of policies for your IAM principals. These ABAC policies can be designed to allow operations when the principal's tag matches the resource tag. ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

Cloud security and governance teams use IAM to define access policies and security permissions for all resources including Amazon S3 buckets, Amazon EC2 instances and any resources you can reference with an ARN. The IAM policies define broad (coarse-grained) permissions to your data lake resources, for example, to allow or deny access at Amazon S3 bucket or prefix level or database level. For more information about IAM ABAC, see [What is ABAC for AWS?](#) in the *IAM User Guide*.

For example, you can create three roles with the project-access tag key. Set the tag value of the first role to Dev, the second to Marketing, and the third to Support. Assign tags with the appropriate value to resources. You can then use a single policy that allows access when the role and the resource are tagged with the same value for project-access.

Data governance teams use Lake Formation to define fine-grained permissions to specific data lake resources. LF-tags are assigned to Data Catalog resources (databases, tables, and columns) and are granted to principals. A principal with LF-tags that match the LF-tags of a resource can access that resource. Lake Formation permissions are secondary to IAM permissions. For example, if IAM permissions don't allow a user access to a data lake, Lake Formation doesn't grant access to any resource within that data lake to that user, even if the principal and resource have matching LF-tags.

Lake Formation tag-based access control (LF-TBAC) works with IAM ABAC to provide additional levels of permissions for your Lake Formation data and resources.

- **Lake Formation TBAC permissions scale with innovation.** It's no longer necessary for an administrator to update existing policies to allow access to new resources. For example, assume that you use an IAM ABAC strategy with the project-access tag to provide access to specific databases within Lake Formation. Using LF-TBAC, the LF-tag Project=SuperApp is assigned to specific tables or columns, and the same LF-tag is granted to a developer for that project. Through IAM, the developer can access the database, and LF-TBAC permissions grant the developer further access to specific tables or columns within tables. If a new table is added to the project, the Lake Formation administrator only needs to assign the tag to the new table for the developer to be given access to the table.
- **Lake Formation TBAC requires fewer IAM policies.** Because you use IAM policies to grant high level access to Lake Formation resources and Lake Formation TBAC for managing more precise data access, you create fewer IAM policies.
- **Using Lake Formation TBAC, teams can change and grow quickly.** This is because permissions for new resources are automatically granted based on attributes. For example, if a new developer joins the project, it's easy to grant this developer access by associating the IAM role to the user and then assigning the required LF-tags to the user. You don't have to change the IAM policy to support a new project or to create new LF-tags.
- **Finer-grained permissions are possible using Lake Formation TBAC.** IAM policies grant access to the top-level resources, such as Data Catalog databases or tables. Using Lake Formation TBAC, you can grant access to specific tables or columns that contain specific data values.

Note

IAM tags are not the same as LF-tags. These tags are not interchangeable. LF-tags are used to grant Lake Formation permissions and IAM tags are used to define IAM policies.

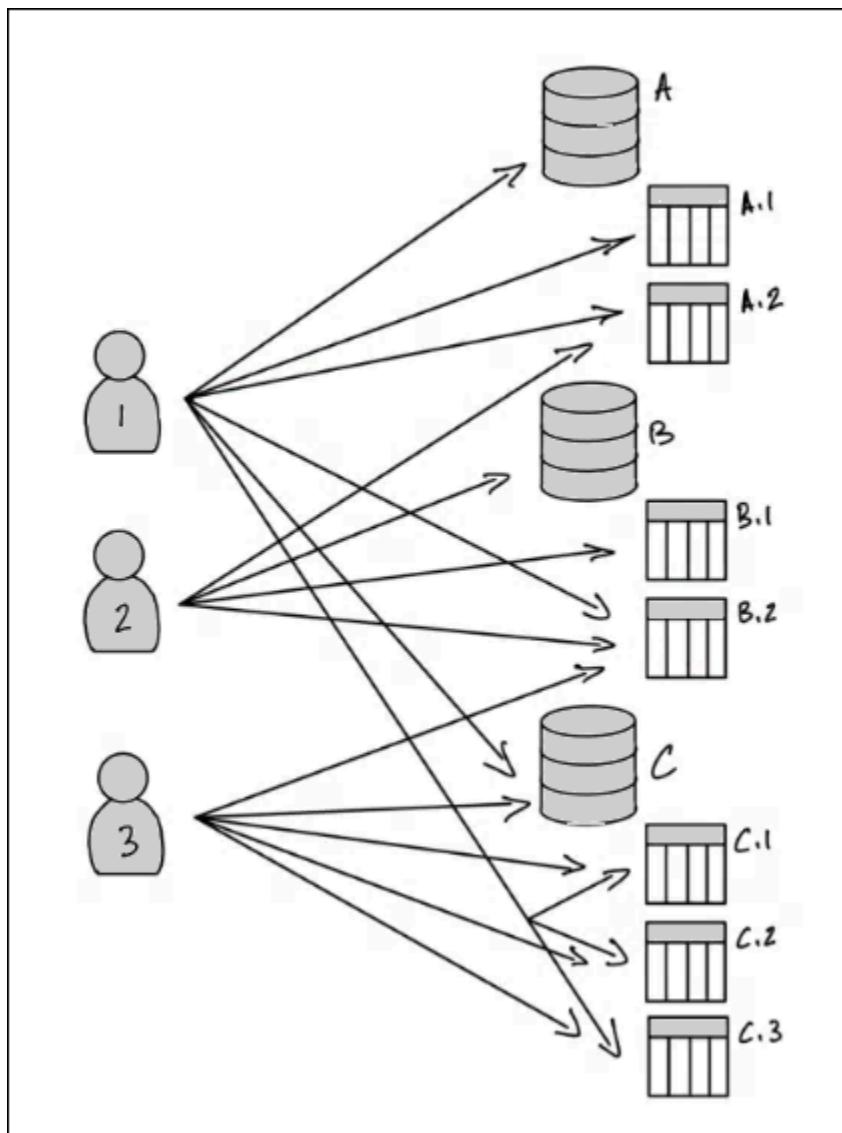
How Lake Formation tag-based access control works

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

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

| Task | Task Details |
|--|---|
| 1. Define the properties and relationships of LF-tags. | - |
| 2. Create the LF-tags in Lake Formation. | Creating LF-Tags (p. 241) |
| 3. Assign LF-tags to Data Catalog resources. | Assigning LF-Tags to Data Catalog resources (p. 246) |
| 4. Grant permissions to other principals to assign LF-tags to resources, optionally with the grant option. | Granting, revoking, and listing LF-Tag permissions (p. 254) |
| 5. Grant LF-tag expressions to principals, optionally with the grant option. | Granting Data Catalog permissions using the LF-TBAC method (p. 192) |
| 6. (Recommended) After verifying that principals have access to the correct resources through the LF-TBAC method, revoke permissions that were granted by using the named resource method. | - |

Consider the case where 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
```

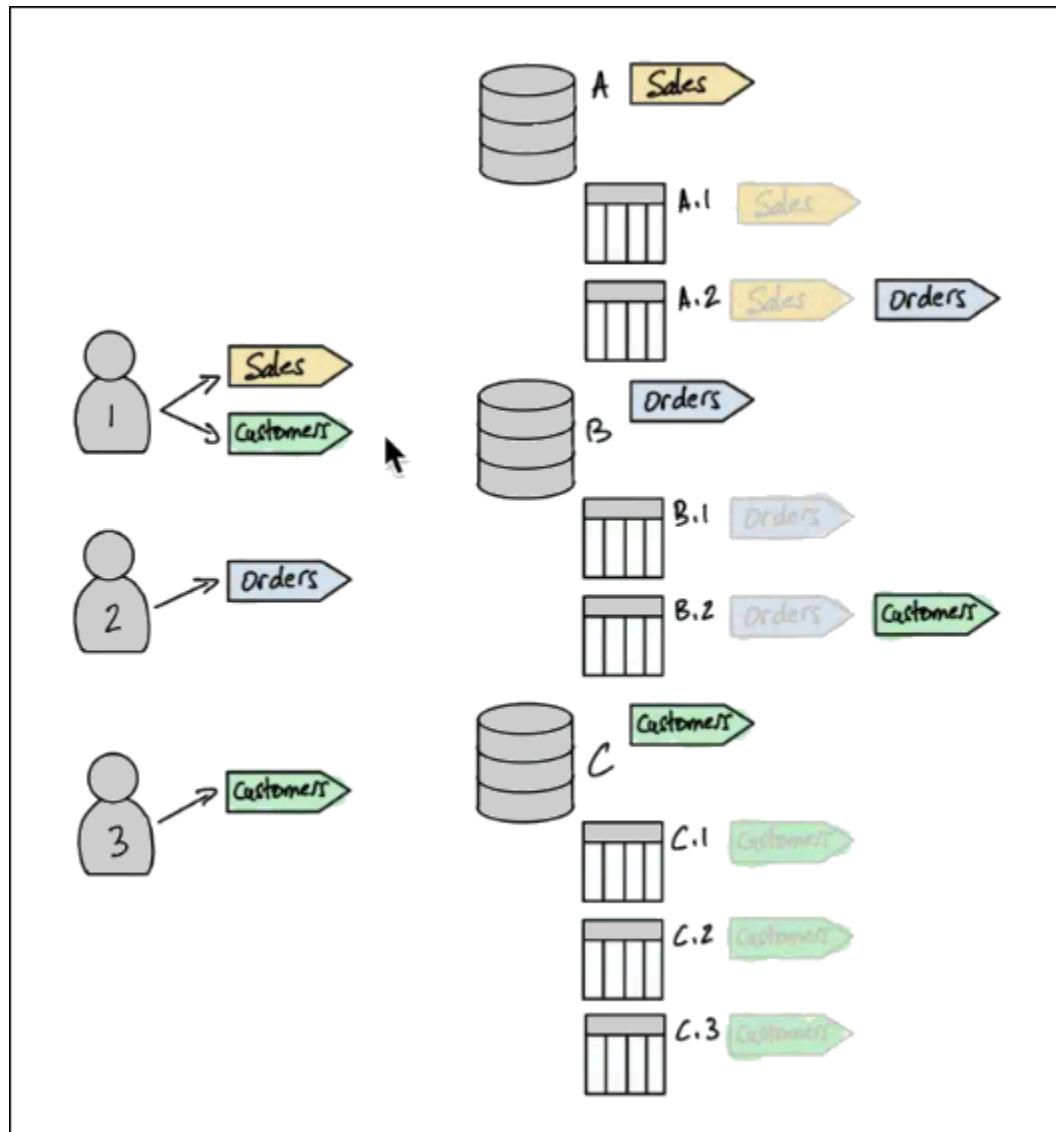
Now consider how the data lake administrator would grant permissions by using LF-TBAC. The following diagram indicates that the data lake administrator has assigned LF-tags to databases and tables, and has granted permissions on LF-tags to principals.

In this example, the LF-tags represent areas of the data lake that contain analytics for different modules of an enterprise resource planning (ERP) application suite. The data lake administrator wants to control

access to the analytics data for the various modules. All LF-tags have the key module and possible values Sales, Orders, and Customers. An example LF-tag looks like this:

```
module=Sales
```

The diagram shows only the LF-tag values.



Tag Assignments to Data Catalog Resources and Inheritance

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

Because of inheritance, the data lake administrator needs to make only the five following LF-tag assignments to resources (in pseudo-code).

```
ASSIGN TAGS module=Sales TO database A
ASSIGN TAGS module=Orders TO table A.2
ASSIGN TAGS module=Orders TO database B
```

```
ASSIGN TAGS module=Customers TO table B.2
ASSIGN TAGS module=Customers TO database C
```

Tag Grants to Principals

After assigning LF-tags to the databases and tables, the data lake administrator must make only four grants of LF-tags to principals, as follows (in pseudo-code).

```
GRANT TAGS module=Sales TO Principal 1
GRANT TAGS module=Customers TO Principal 1
GRANT TAGS module=Orders TO Principal 2
GRANT TAGS module=Customers TO Principal 3
```

Now, a principal with the `module=Sales` LF-tag can access Data Catalog resources with the `module=Sales` LF-tag (for example, database A), a principal with the `module=Customers` LF-tag can access resources with the `module=Customers` LF-tag, and so on.

The preceding grant commands are incomplete. This is because although they indicate through LF-tags the Data Catalog resources that the principals have permissions on, they don't indicate exactly which Lake Formation permissions (such as `SELECT`, `ALTER`) the principals have on those resources. Therefore, the following pseudo-code commands are a more accurate representation of how Lake Formation permissions are granted on Data Catalog resources through LF-tags.

```
GRANT (CREATE_TABLE ON DATABASES) ON TAGS module=Sales TO Principal 1
GRANT (SELECT, INSERT ON TABLES) ON TAGS module=Sales TO Principal 1
GRANT (CREATE_TABLE ON DATABASES) ON TAGS module=Customers TO Principal 1
GRANT (SELECT, INSERT ON TABLES) ON TAGS module=Customers TO Principal 1
GRANT (CREATE_TABLE ON DATABASES) ON TAGS module=Orders TO Principal 2
GRANT (SELECT, INSERT ON TABLES) ON TAGS module=Orders TO Principal 2
GRANT (CREATE_TABLE ON DATABASES) ON TAGS module=Customers TO Principal 3
GRANT (SELECT, INSERT ON TABLES) ON TAGS module=Customers TO Principal 3
```

Putting It Together - Resulting Principal Permissions on Resources

Given the LF-tags assigned to the databases and tables in the preceding diagram, and the LF-tags granted to the principals in the diagram, the following table lists the Lake Formation permissions that the principals have on the databases and tables.

| Principal | Permissions Granted Through LF-tags |
|-------------|--|
| Principal 1 | <ul style="list-style-type: none"> • <code>CREATE_TABLE</code> on database A • <code>SELECT, INSERT</code> on table A.1 • <code>SELECT, INSERT</code> on table A.2 • <code>SELECT, INSERT</code> on table B.2 • <code>CREATE_TABLE</code> on database C • <code>SELECT, INSERT</code> on table C.1 • <code>SELECT, INSERT</code> on table C.2 • <code>SELECT, INSERT</code> on table C.3 |
| Principal 2 | <ul style="list-style-type: none"> • <code>SELECT, INSERT</code> on table A.2 • <code>CREATE_TABLE</code> on database B • <code>SELECT, INSERT</code> on table B.1 • <code>SELECT, INSERT</code> on table B.2 |

| Principal | Permissions Granted Through LF-tags |
|-------------|--|
| Principal 3 | <ul style="list-style-type: none">• SELECT, INSERT on table B.2• CREATE_TABLE on database C• SELECT, INSERT on table C.1• SELECT, INSERT on table C.2• SELECT, INSERT on table C.3 |

Bottom Line

In this simple example, using five assignment operations and eight grant operations, the data lake administrator was able to specify 17 permissions. When there are tens of databases and hundreds of tables, the advantage of the LF-TBAC method over the named resource method becomes clear. In the hypothetical case of the need to grant every principal access to every resource, and where $n(P)$ is the number of principals and $n(R)$ is the number of resources:

- With the named resource method, the number of grants required is $n(P) \times n(R)$.
- With the LF-TBAC method, using a single LF-tag, the total of the number of grants to principals and assignments to resources is $n(P) + n(R)$.

See Also

- [Managing LF-Tags for metadata access control \(p. 240\)](#)
- [Granting Data Catalog permissions using the LF-TBAC method \(p. 192\)](#)

Lake Formation tag-based access control permissions model

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

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

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

- You can assign multiple LF-tags to Data Catalog resources. Only one value for a particular key can be assigned to a particular resource.

For example, you can assign `module=Orders, region=West, division=Consumer`, and so on to a database, table, or column. You can't assign `module=Orders, Customers`.

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

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

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

A principal that is granted this LF-tag expression can access only Data Catalog resources (databases, tables, and columns) that are assigned `module=sales` and either `division=consumer` or `division=commercial`. If you want the principal to be able to access resources that have `module=sales` or `division=commercial`, don't include both in the same grant. Make two grants, one for `module=sales` and one for `division=commercial`.

The simplest LF-tag expression consists of just one LF-tag, such as `module=sales`.

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

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

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 LF-TBAC method. If the data lake administrator grants data permissions with LF-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 LF-TBAC method, but only using the same LF-tag expression.

For example, assume that the data lake administrator makes the following grant (in pseudo-code).

```
GRANT (SELECT ON TABLES) ON TAGS module=customers, region=west,south TO user1 WITH
GRANT OPTION
```

In this case, `user1` can grant `SELECT` on tables to other principals by using the LF-TBAC method, but only with the complete LF-tag expression `module=customers, region=west,south`.

- Although data lake administrators have implicit Lake Formation permissions to create, update, and delete LF-tags, to assign LF-tags to resources, and to grant LF-tags to principals, data lake administrators also need the following LF-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 LF-tags to resources and grant LF-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 \(p. 371\)](#).

- If a principal is granted permissions on a resource with both the LF-TBAC method and the named resource method, the permissions that the principal has on the resource is the union of the permissions granted by both methods.
- Lake Formation supports granting DESCRIBE and ASSOCIATE on LF-tags across accounts, and granting permissions on Data Catalog resources across accounts using the LF-TBAC method. In both cases, the principal is an AWS account ID.

Note

Lake Formation supports cross-account grants to organizations and organizational units using LF-TBAC method. To use this capability, you need to update **Cross account version settings** to **Version 3**.

For more information, see [Cross-account data sharing in Lake Formation \(p. 199\)](#).

Example – Life Cycle of a LF-Tag

1. The data lake administrator Michael creates a LF-tag module=Customers.
2. Michael grants ASSOCIATE on the LF-tag to the data engineer Eduardo. Granting ASSOCIATE implicitly grants DESCRIBE.
3. Michael grants Super on the table Custs to Eduardo with the grant option, so that Eduardo can assign LF-tags to the table. For more information, see [Assigning LF-Tags to Data Catalog resources \(p. 246\)](#).
4. Eduardo assigns the LF-tag module=customers to the table Custs.
5. Michael makes the following grant to data engineer Sandra (in pseudo-code).

```
GRANT (SELECT, INSERT ON TABLES) ON TAGS module=customers TO Sandra WITH GRANT OPTION
```

6. Sandra makes the following grant to data analyst Maria.

```
GRANT (SELECT ON TABLES) ON TAGS module=customers TO Maria
```

Maria can now run queries on the Custs table.

See Also

- [Metadata access control \(p. 292\)](#)

Lake Formation Tag-based access control notes and restrictions

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

- Using Lake Formation tag-based access control (LF-TBAC) to grant cross-account access to Data Catalog resources requires additions to the Data Catalog resource policy for your AWS account. For more information, see [Cross-account data sharing prerequisites \(p. 200\)](#).

- LF-Tag keys and LF-tag values can't exceed 50 characters in length.
- The maximum number of LF-tags that can be assigned to a Data Catalog resource is 50.
- The following limits are soft limits:
 - The maximum number of LF-tags that can be created is 1000.
 - The maximum number of values that can be defined for a LF-tag is 1000.
- Tags keys and values are converted to all lower case when they are stored.
- Only one value for a LF-tag can be assigned to a particular resource.
- If multiple LF-tags are granted to a principal with a single grant, the principal can access only Data Catalog resources that have all of the LF-tags.
- AWS Glue ETL jobs require full table access. The jobs will fail if AWS Glue ETL role does not have access to all columns in a table. It is possible to apply LF-tags at a column-level, but it may cause AWS Glue ETL roles to lose full table access and have jobs fail. Using data filters for column and/or row filtering is not affected by this limitation.
- If a LF-tag expression evaluation results in access to only a subset of table columns, but the Lake Formation permission granted when there is a match is one of the permissions that required full column access, namely ALTER, DROP, INSERT, or DELETE, then none of those permissions is granted. Instead, only DESCRIBE is granted. If the granted permission is ALL (Super), then only SELECT and DESCRIBE are granted.

Managing LF-Tags for metadata access control

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

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

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

Topics

- [Creating LF-Tags \(p. 241\)](#)
- [Updating LF-Tags \(p. 242\)](#)
- [Deleting LF-Tags \(p. 242\)](#)
- [Listing LF-Tags \(p. 243\)](#)
- [Assigning LF-Tags to Data Catalog resources \(p. 246\)](#)
- [Viewing LF-Tags assigned to a resource \(p. 250\)](#)
- [Viewing the resources that a LF-Tag is assigned to \(p. 252\)](#)

See also

- [Granting, revoking, and listing LF-Tag permissions \(p. 254\)](#)
- [Granting Data Catalog permissions using the LF-TBAC method \(p. 192\)](#)
- [Lake Formation Tag-based access control \(p. 231\)](#)

Creating LF-Tags

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

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

Console

To create a LF-tag

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

Sign in as a data lake administrator.

2. In the navigation pane, under **Permissions**, choose **LF-Tags**.

The **LF-Tags** page appears.

| LF-Tags (3) | | | |
|-----------------------|-------------|--------------------------|--------------|
| | | 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 |

3. Choose **Add tag**.
4. In the **Add LF-Tag** dialog box, enter a key and one or more values.

Each key must have at least one value. To enter multiple values, either enter a comma-delimited list and then press **Enter**, or enter one value at a time and choose **Add** after each one. The maximum number of values permitted is 15.

5. Choose **Add tag**.

AWS CLI

To create a LF-tag

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

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

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

Updating LF-Tags

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

When you delete a LF-tag value, no check is performed for the presence of that LF-tag value on any Data Catalog resource. If the deleted LF-tag value is associated with a resource, it is no longer visible for the resource, and any principals that were granted permissions on that key-value pair no longer have the permissions.

Before deleting a LF-tag value, you can optionally use the [remove-lf-tags-from-resource command \(p. 250\)](#) command to remove the LF-tag from Data Catalog resources that have the value that you want to delete, and then retag the resource with the values that you want to keep.

Only data lake administrators can update a LF-tag.

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

Console

To update a LF-tag (console)

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
Sign in as a data lake administrator.
2. In the navigation pane, under **Permissions**, choose **LF-Tags**.
3. On the **LF-Tags** page, select a LF-tag, and then choose **Edit**.
4. In the **Edit LF-Tag** dialog box, add or remove LF-tag values.

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

5. Choose **Save**.

AWS CLI

To update a LF-tag (AWS CLI)

- Enter an `update-lf-tag` command. Provide one or both of the following arguments:
 - `--tag-values-to-add`
 - `--tag-values-to-delete`

Example

The following example replaces the value `vp` with the value `vice-president` for the LF-tag key `level`.

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

Deleting LF-Tags

You can delete LF-tags that are no longer in use. No check is performed for the presence of the LF-tag on a Data Catalog resource. If the deleted LF-tag is associated with a resource, it is no longer visible

for the resource, and any principals that were granted permissions on that LF-tag no longer have the permissions.

Before deleting a LF-tag, you can optionally use the [remove-lf-tags-from-resource \(p. 250\)](#) command to remove the LF-tag from all resources.

Only data lake administrators can delete a LF-tag.

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

Console

To delete a LF-tag (console)

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
Sign in as a data lake administrator.
2. In the navigation pane, under **Permissions**, choose **LF-Tags**.
3. On the **LF-Tags** page, select a LF-tag, and then choose **Delete**.
4. In the **Delete tag environment?** dialog box, to confirm the deletion, enter the LF-tag key value in the designated field and then choose **Delete**.

AWS CLI

To delete a LF-tag (AWS CLI)

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

Example

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

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

Listing LF-Tags

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

Data lake administrators can see all LF-tags that are defined in the local AWS account and all LF-tags for which the DESCRIBE and ASSOCIATE permissions have been granted to the local account from external accounts. The data lake administrator can see all values for all LF-tags.

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

Console

To list LF-tags (console)

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

Sign in as a data lake administrator or as a principal that has been granted permissions on LF-tags and that has the `lakeformation>ListLFTags` IAM permission.

2. In the navigation pane, under **Permissions**, choose **LF-Tags**.

The **LF-Tags** page appears.

| LF-Tags (3) | | | <input type="button" value="C"/> | <input type="button" value="Delete"/> | <input type="button" value="Edit"/> | <input type="button" value="Add tag"/> | |
|-----------------------|-------------|--------------------------|----------------------------------|---------------------------------------|-------------------------------------|--|------------------------|
| | | | <input type="text"/> Find tag | < | 1 | > | <input type="button"/> |
| | 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 LF-tags that were shared with your account from an external account.

AWS CLI

To list LF-tags (AWS CLI)

- Run the following command as a data lake administrator or as a principal that has been granted permissions on LF-tags and that has the `lakeformation>ListLFTags` IAM permission.

```
aws lakeformation list-lf-tags
```

The output is similar to the following.

```
{
  "LFTags": [
    {
      "CatalogId": "111122223333",
      "TagKey": "level",
      "TagValues": [
        "director",
        "vp",
        "c-level"
      ]
    },
    {
      "CatalogId": "111122223333",
      "TagKey": "module",
      "TagValues": [
        "Orders",
        "Sales",
        "Customers"
      ]
    }
  ]
}
```

To also see LF-tags that were granted from external accounts, include the command option --resource-share-type ALL.

```
aws lakeformation list-lf-tags --resource-share-type ALL
```

The output is similar to the following. Note the NextToken key, which indicates that there is more to list.

```
{  
    "LFTags": [  
        {  
            "CatalogId": "111122223333",  
            "TagKey": "level",  
            "TagValues": [  
                "director",  
                "vp",  
                "c-level"  
            ]  
        },  
        {  
            "CatalogId": "111122223333",  
            "TagKey": "module",  
            "TagValues": [  
                "Orders",  
                "Sales",  
                "Customers"  
            ]  
        }  
    ],  
    "NextToken": "eyJleHBpcmF0aW...ZXh0Ijp0cnVlfQ=="  
}
```

Repeat the command, and add the --next-token argument to view any remaining local LF-tags and LF-tags that were granted from external accounts. LF-tags from external accounts are always on a separate page.

```
aws lakeformation list-lf-tags --resource-share-type ALL  
--next-token eyJleHBpcmF0aW...ZXh0Ijp0cnVlfQ==
```

```
{  
    "LFTags": [  
        {  
            "CatalogId": "123456789012",  
            "TagKey": "region",  
            "TagValues": [  
                "central",  
                "south"  
            ]  
        }  
    ]  
}
```

API

You can use the SDKs available for Lake Formation to lists the tags that the requester has permission to view.

```
import boto3

client = boto3.client('lakeformation')
...

response = client.list_lf_tags(
    CatalogId='string',
    ResourceShareType='ALL',
    MaxResults=50
)
```

This command returns a dict object with the following structure:

```
{
    'LFTags': [
        {
            'CatalogId': 'string',
            'TagKey': 'string',
            'TagValues': [
                'string',
            ]
        },
    ],
    'NextToken': 'string'
}
```

For more information about the required permissions, see [Lake Formation Personas and IAM Permissions Reference \(p. 371\)](#).

Assigning LF-Tags to Data Catalog resources

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

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

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

Topics

- [Requirements for managing tags assigned to resources \(p. 246\)](#)
- [Assign LF-tags to a table column \(p. 247\)](#)
- [Assign LF-tags to a Data Catalog resource \(p. 248\)](#)
- [Updating LF-tags for a resource \(p. 250\)](#)
- [Removing LF-tag from a resource \(p. 250\)](#)

Requirements for managing tags assigned to resources

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

- Have the Lake Formation ASSOCIATE permission on the LF-tag.
- Have the IAM `lakeformation:AddLFTagsToResource` permission.
- Have `glue:GetDatabase` permission on a Glue database.

- Be the resource owner (creator), have the Super Lake Formation permission on the resource with the GRANT option, or have the following permissions with the GRANT option:
 - For databases in the same AWS account: DESCRIBE, CREATE_TABLE, ALTER, and DROP
 - For databases in an external account: DESCRIBE, CREATE_TABLE and ALTER
 - For tables (and columns): DESCRIBE, ALTER, DROP, INSERT, SELECT, and DELETE

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

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

Assign LF-tags to a table column

To assign LF-tags to a table column (console)

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
Sign in as a user who meets the requirements listed above.
2. In the navigation pane, choose **Tables**.
3. Choose a table name (not the option button next to the table name).
4. On the table details page, in the **Schema** section, choose **Edit schema**.
5. On the **Edit schema** page, select one or more columns, and then choose **Edit tags**.

Note

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

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

| Inherited keys | Values |
|----------------|----------------------|
| level | director (inherited) |
| module | Orders (inherited) |

Assign new LF-Tag

You can add 50 more tags.

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 LF-Tag**. Then for **Assigned keys**, choose a key, and for **Values**, choose a value for the key.

Edit LF-Tags: product_id [Learn More](#)

LF-Tags

After they are associated with catalog resources, LF-Tags allow you to create scalable permissions.

| Inherited keys | Values |
|-------------------------------------|------------------------|
| <input type="text" value="level"/> | director (inherited) ▾ |
| <input type="text" value="module"/> | Orders (inherited) ▾ |

| Assigned keys | Values |
|--|---------------------------------------|
| <input type="text" value="environment"/> | Production Production Development |

You can add 49 more tags.

Assign new LF-Tag

Cancel **Save**

8. (Optional) Choose **Assign new LF-Tag** again to add another LF-tag.

9. Choose **Save**.

Assign LF-tags to a Data Catalog resource

Console

To assign LF-tags to a Data Catalog database or table

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

Sign in as a user who meets the requirements listed earlier.

2. In the navigation pane, under **Data catalog**, do one of the following:

- To assign LF-tags to databases, choose **Databases**.
- To assign LF-tags to tables, choose **Tables**.

3. Choose a database or table, and on the **Actions** menu, choose **Edit tags**.

The **Edit LF-Tags: resource-name** dialog box appears.

If a table inherits LF-tags from its containing database, the window displays the inherited LF-tags. Otherwise, it displays the text "There are no inherited LF-Tags associated with the resource."

LF-Tags

After they are associated with catalog resources, LF-Tags allow you to create scalable permissions.

| Inherited keys | Values |
|----------------|----------------------|
| level | director (inherited) |

| Assigned keys | Values |
|--|--|
| module | <input type="text" value="Enter LF-Tag value"/> ▲ Orders Sales Customers |
| Assign new LF-Tag Remove You can add 49 more tags. | |

[Cancel](#) [Save](#)

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

AWS CLI

To assign LF-tags to a Data Catalog resource

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

The following example assigns the LF-tag `module=orders` to the table `orders` in the database `erp`. It uses the shortcut syntax for the `--lf-tags` argument. The `CatalogID` property for `--lf-tags` is optional. If not provided, the catalog ID of the resource (in this case, the table) is assumed.

```
aws lakeformation add-lf-tags-to-resource --resource '{ "Table": { "DatabaseName": "erp", "Name": "orders" } }' --lf-tags CatalogId=111122223333,TagKey=module,TagValues=orders
```

The following is the output if the command succeeds.

```
{
  "Failures": []
}
```

This next example assigns two LF-tags to the sales table, and uses the JSON syntax for the --lf-tags argument.

```
aws lakeformation add-lf-tags-to-resource --resource '{ "Table": { "DatabaseName": "erp", "Name": "sales" } }' --lf-tags '[{"TagKey": "module", "TagValues": ["sales"]}, {"TagKey": "environment", "TagValues": ["development"]} ]'
```

This next example assigns the LF-tag level=director to the total column of the table sales.

```
aws lakeformation add-lf-tags-to-resource --resource '{ "TableWithColumns": { "DatabaseName": "erp", "Name": "sales", "ColumnNames": [ "total" ] } }' --lf-tags TagKey=level,TagValues=director
```

Updating LF-tags for a resource

To update a LF-tag for a Data Catalog resource (AWS CLI)

- Use the add-lf-tags-to-resource command, as described in the previous procedure.
Adding a LF-tag with the same key as an existing LF-tag but with a different value updates the existing value.

Removing LF-tag from a resource

To remove a LF-tag for a Data Catalog resource (AWS CLI)

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

If a table has a LF-tag value that overrides the value that is inherited from the parent database, removing that LF-tag from the table restores the inherited value. This behavior also applies to a column that overrides key values inherited from the table.

The following example removes the LF-tag level=director from the total column of the sales table. The CatalogID property for --lf-tags is optional. If not provided, the catalog ID of the resource (in this case, the table) is assumed.

```
aws lakeformation remove-lf-tags-from-resource  
--resource '{ "TableWithColumns": { "DatabaseName": "erp", "Name": "sales", "ColumnNames": [ "total" ] } }'  
--lf-tags CatalogId=111122223333,TagKey=level,TagValues=director
```

Viewing LF-Tags assigned to a resource

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

Console

To view the LF-tags that are assigned to a resource (console)

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

Sign in as the data lake administrator, the resource owner, or a user who has been granted Lake Formation permissions on the resource.

2. In the navigation pane, under the heading **Data catalog**, do one of the following:
 - To view LF-tags assigned to a database, choose **Databases**.
 - To view LF-tags assigned to a table, choose **Tables**.
3. On the **Tables** or **Databases** page, choose the name of the database or table. Then on the details page, scroll down to the **LF-Tags** section.

The following screenshot shows the LF-tags assigned to a `customers` table, which is contained in the `retail` database. The module LF-tag is inherited from the database. The `credit_limit` column has the `level=vp` LF-tag assigned.

| Resource | Key | Value | Inherited from |
|-----------------------|-------------|------------|----------------|
| customers (table) | module | Customers | retail |
| customers (table) | environment | Production | - |
| credit_limit (column) | level | vp | - |

AWS CLI

To view the LF-tags that are assigned to a resource (AWS CLI)

- Enter a command similar to the following.

```
aws lakeformation get-resource-lf-tags --show-assigned-lf-tags --resource
'{"Table": {"CatalogId": "111122223333", "DatabaseName": "erp", "Name": "sales"}}'
```

The command returns the following output.

```
{
  "TableTags": [
    {
      "CatalogId": "111122223333",
      "TagKey": "module",
      "TagValues": [
        "sales"
      ]
    },
    {
      "CatalogId": "111122223333",
      "TagKey": "environment",
      "TagValues": [
        "Production"
      ]
    }
  ]
}
```

```
        "TagKey": "environment",
        "TagValues": [
            "development"
        ]
    },
    "ColumnTags": [
        {
            "Name": "total",
            "Tags": [
                {
                    "CatalogId": "111122223333",
                    "TagKey": "level",
                    "TagValues": [
                        "director"
                    ]
                }
            ]
        }
    ]
}
```

This output shows only LF-tags that are explicitly assigned, not inherited. If you want to see all LF-tags on all columns, including inherited LF-tags, omit the `--show-assigned-lf-tags` option.

Viewing the resources that a LF-Tag is assigned to

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

- DESCRIBE or ASSOCIATE on the LF-tag.
- DESCRIBE or any other Lake Formation permission on the resource.

In addition, you need the following AWS Identity and Access Management (IAM) permissions:

- `lakeformation:SearchDatabasesByLFTags`
- `lakeformation:SearchTablesByLFTags`

Console

To view the resources that a LF-tag is assigned to (console)

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
Sign in as a data lake administrator or as a user who meets the requirements listed earlier.
2. In the navigation pane, under the headings **Permissions** and **Administrative roles and tasks**, choose **LF-Tags**.
3. Choose a LF-tag key (not the option button next to the key name).

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

The screenshot shows the AWS Lake Formation LF-Tag details page. At the top, there is a header "module". Below it, a table titled "LF-Tag" shows one entry: "Key" (module) and "Values" (Orders, Sales, Customers). There are "Delete" and "Edit" buttons next to the table. Below this, a section titled "Associated data catalog resources (12)" lists various resources. A search bar "Find resource" is at the top of the list. The table columns are "Key", "Values", "Resource type", and "Resource". The data includes:

| Key | Values | Resource type | Resource |
|--------|-----------|---------------|-----------------------------|
| module | Customers | DATABASE | retail |
| module | Customers | TABLE | customers |
| module | Orders | TABLE | inventory |
| module | Customers | COLUMN | customers.cust_first_name |
| module | Customers | COLUMN | customers.work_phone_number |
| module | Customers | COLUMN | customers.company_name |
| module | Customers | COLUMN | customers.credit_limit |

AWS CLI

To view the resources that a LF-tag is assigned to

- Run a `search-tables-by-lf-tags` or `search-databases-by-lf-tags` command.

Example

The following example lists tables and columns that have the `level=vp` LF-tag assigned. For each table and column listed, all assigned LF-tags for the table or column are output, not just the search expression.

```
aws lakeformation search-tables-by-lf-tags --expression TagKey=level,TagValues=vp
```

For more information about the required permissions, see [Lake Formation Personas and IAM Permissions Reference \(p. 371\)](#).

Granting, revoking, and listing LF-Tag permissions

You can grant the DESCRIBE and ASSOCIATE Lake Formation permissions on LF-tags to principals so that they can view the LF-tags and assign them to Data Catalog resources (databases, tables, and columns). When LF-tags are assigned to Data Catalog resources, you can use the Lake Formation tag-based access control (LF-TBAC) method to secure those resources. For more information, see [Lake Formation Tag-based access control \(p. 231\)](#).

At first, only the data lake administrator can grant these permissions. If the data lake administrator grants these permissions with the grant option, other principals can grant them. The DESCRIBE and ASSOCIATE permissions are explained in [Lake Formation tag-based access control permissions model \(p. 237\)](#).

You can grant the DESCRIBE and ASSOCIATE permissions on a LF-tag to an external AWS account. A data lake administrator in that account can then grant those permissions to other principals in the account. Principals to whom the data lake administrator in the external account grants the ASSOCIATE permission can then assign LF-tags to Data Catalog resources that you shared with their account.

When granting to an external account, you must include the grant option.

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

Topics

- [Listing LF-Tag permissions using the console \(p. 254\)](#)
- [Granting LF-Tag permissions using the console \(p. 255\)](#)
- [Granting, Revoking, and Listing LF-Tag Permissions Using the AWS CLI \(p. 257\)](#)

For more information see [Managing LF-Tags for metadata access control \(p. 240\)](#) and [Lake Formation Tag-based access control \(p. 231\)](#).

Listing LF-Tag permissions using the console

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

To list LF-tag permissions (console)

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

Sign in as a data lake administrator or as a user to whom the ASSOCIATE or DESCRIBE permissions on LF-tags have been granted.

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

The **Grant LF-Tag permissions** page appears.

| Tag permissions | | | | | | <input type="button" value="C"/> | <input type="button" value="View"/> | <input type="button" value="Revoke"/> | <input type="button" value="Grant"/> |
|-----------------------|---|----------------|-------------|---------------|-------------|----------------------------------|-------------------------------------|---------------------------------------|--------------------------------------|
| | Principal | Principal type | Keys | values | Permissions | Grantable | | | |
| <input type="radio"/> | arn:aws:iam::111122223333:user/datalake_admin | IAM User | environment | All values | DESCRIBE | DESCRIBE | | | |
| <input type="radio"/> | arn:aws:iam::111122223333:user/datalake_admin | IAM User | environment | All values | ASSOCIATE | ASSOCIATE | | | |
| <input type="radio"/> | arn:aws:iam::111122223333:user/datalake_user1 | IAM User | module | Orders, Sales | ASSOCIATE | | | | |
| <input type="radio"/> | arn:aws:iam::111122223333:user/datalake_admin | IAM User | module | All values | DESCRIBE | DESCRIBE | | | |
| <input type="radio"/> | arn:aws:iam::111122223333:user/datalake_admin | IAM User | module | All values | ASSOCIATE | ASSOCIATE | | | |

Granting LF-Tag permissions using the console

The following steps explain how to grant permissions on LF-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.
- **LF-Tags** – The LF-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 LF-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 account, AWS organization or IAM principal 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 \(p. 220\)](#).

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.

An organization ID consists of "o-" followed by 10 to 32 lower-case letters or digits.

An organizational unit ID starts with "ou-" followed by 4 to 32 lowercase letters or digits (the ID of the root that contains the OU). This string is followed by a second "-" dash and 8 to 32 additional lowercase letters or digits.

For IAM principal, enter the ARN for the IAM user or role.

Specify the LF-Tags

In the **LF-Tags** section, specify the LF-tags to grant permissions on.

LF-Tags

Tag permission scope

Choose to grant permissions on all or a subset of LF-Tags.

Key

module X

Values

▼

Remove

Orders X Sales X

Enter a tag key

▼

Remove

Add LF-Tag

1. Choose **Add LF-Tag** to reveal the first row of fields for specifying a LF-tag.
2. Position the cursor in the **Key** field, optionally start typing to narrow down the selection list, and select a LF-tag key.
3. In the **Values** list, select one or more values, and then press **Tab** or click or tap outside the field to save the selected values.

Note

If one of the rows in the **Values** list has focus, pressing **Enter** selects or clears the check box.

The selected values appear as tiles below the **Values** list. Choose the **X** to remove a value. Choose **Remove** to remove the entire LF-tag.

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

Specify the 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 LF-Tag Permissions Using the AWS CLI

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

To list LF-tag permissions (AWS CLI)

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

The following command requests all LF-tags that you have permissions on.

```
aws lakeformation list-permissions --resource-type LF_TAG
```

The following is sample output for a data lake administrator, who sees all LF-tags granted to all principals. Non-administrative users see only LF-tags granted to them. LF-tag permissions granted from an external account appear on a separate results page. To see them, repeat the command and supply the `--next-token` argument with the token returned from the previous command run.

```
{  
    "PrincipalResourcePermissions": [  
        {  
            "Principal": {  
                "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/  
datalake_admin"  
            },  
            "Resource": {  
                "LFTag": {  
                    "CatalogId": "111122223333",  
                    "Name": "test-tag"  
                }  
            }  
        }  
    ]  
}
```

```

        "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": "eyJzaG91bGRRdWV...Wlzc2lvbnMiOnRydWV9"
}

```

You can list all grants for a specific LF-tag key. The following command returns all permissions granted on the LF-tag module.

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

You can also list LF-tag values granted to a specific principal for a specific LF-tag. When supplying the --principal argument, you must supply the --resource argument. Therefore, the command can only effectively request the values granted to a specific principal for a specific LF-tag key. The following command shows how to do this for the principal `datalake_user1` and the LF-tag key `module`.

```
aws lakeformation list-permissions --principal DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --resource-type LF_TAG --resource '{ "LFTag": {"CatalogId": "111122223333", "TagKey": "module", "TagValues": ["*"]}}'
```

The following is sample output.

```
{
    "PrincipalResourcePermissions": [
        {

```

```
"Principal": {  
    "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/  
datalake_user1"  
},  
"Resource": {  
    "LFTag": {  
        "CatalogId": "111122223333",  
        "TagKey": "module",  
        "TagValues": [  
            "Orders",  
            "Sales"  
        ]  
    },  
    "Permissions": [  
        "ASSOCIATE"  
    ],  
    "PermissionsWithGrantOption": []  
}  
]  
}
```

To grant permissions on LF-tags (AWS CLI)

- Enter a command similar to the following. This example grants to user `datalake_user1` the `ASSOCIATE` permission on the LF-tag with the key `module`. It grants permissions to view and assign all values for that key, as indicated by the asterisk (*).

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/  
datalake_user1 --permissions "ASSOCIATE" --resource '{ "LFTag":  
{"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

Granting the `ASSOCIATE` permission implicitly grants the `DESCRIBE` permission.

The next example grants `ASSOCIATE` to the external AWS account 1234-5678-9012 on the LF-tag with the key `module`, with the grant option. It grants permissions to view and assign only the values `sales` and `orders`.

```
aws lakeformation grant-permissions --principal  
DataLakePrincipalIdentifier=123456789012 --permissions "ASSOCIATE"  
--permissions-with-grant-option "ASSOCIATE" --resource '{ "LFTag":  
{"CatalogId":"111122223333","TagKey":"module","TagValues":["sales", "orders"]}}'
```

To revoke permissions on LF-tags (AWS CLI)

- Enter a command similar to the following. This example revokes the `ASSOCIATE` permission on the LF-tag with the key `module` from user `datalake_user1`.

```
aws lakeformation revoke-permissions --principal  
DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/  
datalake_user1 --permissions "ASSOCIATE" --resource '{ "LFTag":  
{"CatalogId":"111122223333","TagKey":"module","TagValues":["*"]}}'
```

Data filtering and cell-level security in Lake Formation

When you grant Lake Formation permissions on a Data Catalog table, you can include data filtering specifications to restrict access to certain data in query results and engines integrated with Lake Formation. Lake Formation uses data filtering to achieve column-level security, row-level security, and cell-level security.

Topics

- [Overview of data filtering \(p. 260\)](#)
- [Data filters in Lake Formation \(p. 261\)](#)
- [PartiQL support in row filter expressions \(p. 263\)](#)
- [Notes and restrictions for column-level filtering \(p. 265\)](#)
- [Notes and restrictions for row-Level and cell-level filtering \(p. 265\)](#)
- [Permissions required for querying tables with cell-level filtering \(p. 266\)](#)
- [Managing data filters \(p. 267\)](#)

Overview of data filtering

With the data filtering capabilities of Lake Formation, you can implement the following levels of data security.

Column-level security

Granting permissions on a Data Catalog table with column-level security (column filtering) allows users to view only specific columns that they have access to in the table. Consider a persons table that is used in multiple applications for a large multi-region communications company. Granting permissions on Data Catalog tables with column filtering can restrict users who don't work in the HR department from seeing personally identifiable information (PII) such as a social security number or birth date.

Row-level security

Granting permissions on a Data Catalog table with row-level security (row filtering) allows users to view only specific rows of data that they have access to in the table. The filtering is based on the values of one or more columns. For example, if different regional offices of the communications company have their own HR departments, you can limit the person records that HR employees can see to only records for employees in their region.

Cell-level security

Cell-level security combines row filtering and column filtering for a highly flexible permissions model. If you view the rows and columns of a table as a grid, by using cell-level security, you can restrict access to individual elements (cells) of the grid anywhere in the two dimensions. That is, you can restrict access to different columns depending on the row. This is illustrated by the following diagram, in which restricted columns are shaded.

| | Col1 | Col2 | Col3 | Col4 | Col5 | Col6 |
|------|------|------|------|------|------|------|
| Row1 | | | | | | |
| Row2 | | | | | | |
| Row3 | | | | | | |
| Row4 | | | | | | |
| Row5 | | | | | | |

Continuing the example of the persons table, you can create a *data filter* at the cell-level that restricts access to the street address column if the row has the country column set to "UK", but allows access to the street address column if the row has the country column set to "US".

Filters apply only to read operations. Therefore, you can grant only the SELECT Lake Formation permission with filters.

Data filters in Lake Formation

You can implement column-level, row-level, and cell-level security by creating *data filters*. You select a data filter when you grant the SELECT Lake Formation permission on tables.

Each data filter belongs to a specific table in your Data Catalog. A data filter includes the following information:

- Filter name
- The Catalog IDs of the table associated with the filter
- Table name
- Name of the database that contains the table
- Column specification – a list of columns to include or exclude in query results.
- Row filter expression – an expression that specifies the rows to include in query results. With some restrictions, the expression has the syntax of a WHERE clause in the PartiQL language. To specify all rows, enter true in the console or use AllRowsWildcard in API calls.

For more information about what is supported in row filter expressions, see [PartiQL support in row filter expressions \(p. 263\)](#).

The level of filtering that you get depends on how you populate the data filter.

- When you specify the "all columns" wildcard and provide a row filter expression, you are establishing row-level security (row filtering) only.
- When you include or exclude specific columns and specify "all rows" using the all-rows wildcard, you are establishing column-level security (column filtering) only.
- When you include or exclude specific columns and also provide a row filter expression, you are establishing cell-level security (cell filtering).

The following screenshot from the Lake Formation console shows a data filter that performs cell-level filtering. For queries against the orders table, it restricts access to the customer_name column in rows where the product_type column contains 'pharma'. In restricted rows, the query results return NULL for the customer_name column.

Create data filter



Data filter name

Enter a name that describes this data access filter.

restrict-pharma

Name may contain letters (A-Z), numbers (0-9), hyphens (-), or under-scores (_), and be less than 256 characters.

Target database

Select the database that contains the target table.

Choose databases



Load more

sales



054881201579

Target table

Select the table for which the data filter will be created.

Choose tables



Load more

orders



054881201579

Column-level access

Choose whether this filter should have column-level restrictions.

Access to all columns

Filter won't have any column restrictions.

Include columns

Filter will only allow access to specific columns.

Exclude columns

Filter will allow access to all but specific columns.

Select columns

Choose one or more columns



customer_name



string

Note the use of single quotes to enclose the string literal, 'pharma'.

You can use the Lake Formation console to create this data filter, or you can supply the following request object to the `CreateDataCellsFilter` API operation.

```
{
    "Name": "restrict-pharma",
    "DatabaseName": "sales",
    "TableName": "orders",
    "TableCatalogId": "111122223333",
    "RowFilter": {"FilterExpression": "product_type='pharma'" },
    "ColumnWildcard": {
        "ExcludedColumnNames": ["customer_name"]
    }
}
```

You can create as many data filters as you need for a table. In order to do so, you require `SELECT` permission with the `grant` option on a table. Data Lake Administrators by default have the permission to create *data filters* on all tables in that account. You typically only use a subset of the possible data filters when granting permissions on the table to a principal. For example, you could create a second data filter for the `orders` table that is a row-security-only data filter. Referring to the preceding screenshot, you could choose the **Access to all columns** option and include a row filter expression of `product_type<>pharma`. The name of this data filter could be `no-pharma`. It restricts access to all rows that have the `product_type` column set to 'pharma'.

The request object for the `CreateDataCellsFilter` API operation for this data filter is the following.

```
{
    "Name": "no-pharma",
    "DatabaseName": "sales",
    "TableName": "orders",
    "TableCatalogId": "111122223333",
    "RowFilter": {"FilterExpression": "product_type<>'pharma'" },
    "ColumnNames": ["customer_id", "customer_name", "order_num",
                    "product_id", "purchase_date", "product_type",
                    "product_manufacturer", "quantity", "price"]
}
```

You could then grant `SELECT` on the `orders` table with the `restrict-pharma` data filter to an administrative user, and `SELECT` on the `orders` table with the `no-pharma` data filter to non-administrative users. For users in the healthcare sector, you would grant `SELECT` on the `orders` table with full access to all rows and columns (no data filter), or perhaps with yet another data filter that restricts access to pricing information.

See also

- [Managing data filters \(p. 267\)](#)

PartiQL support in row filter expressions

You can construct row filter expressions using a subset of PartiQL data types, operators, and aggregations. Lake Formation does not allow any user defined or standard partiQL functions in the filter expression. You can use comparison operators to compare columns with constants (for example, `views >= 10000`), but you can't compare columns with other columns.

A Row filter expression may be a simple expression or a composite expression. Total length of the expression must be less than 2048 characters.

Simple expression

A simple expression will be of the format: <column name > <comparison operator ><value >

- **Column name**

It must be either a top level data column or a partition column present in the table schema and must belong to the [Supported data types \(p. 264\)](#) listed below.

- **Comparison operator**

The following are the supported operators: =, >, <, >=, <=, <>, !=, BETWEEN, IN, LIKE All string comparisons and LIKE pattern matches are case-sensitive.

- **Column value**

The Column value must match the data type of the column name.

Composite expression

A composite expression will be of the format: (<simple expression >) <AND/OR >(<simple expression >). Composite expressions can be further combined using logical operators AND/OR.

Supported data types

Row filters that refer to an AWS Glue Data Catalog table that contains an unsupported data types will result in an error. The following are the supported data types for table columns and constants, which are mapped to Amazon Redshift data types:

- STRING, CHAR, VARCHAR
- INT, LONG, BIGINT, FLOAT, DECIMAL, DOUBLE
- BOOLEAN

For more information about data types in Amazon Redshift, see [Data types in Amazon Redshift Database Developer Guide](#).

Row filter expressions

Example

The following are examples of valid row filter expressions for a table with columns: country (String), id (Long), year (partition column of type Integer), month (partition column of type Integer)

- year > 2010 and country != 'US'
- (year > 2010 and country = 'US') or (month < 8 and id > 23)
- (country between 'Z' and 'U') and (year = 2018)
- (country like '%ited%') and (year > 2000)

String constants must be enclosed in single-quotes.

Reserved keywords

If your row filter expression contains PartiQL keywords, you will receive a parsing error as column names may conflict with the keywords. When this happens, escape the column names by using double quotes.

Some examples of reserved keywords are “first”, “last”, “asc”, “missing”. See PartiQL specification for a list of reserved keywords.

PartiQL reference

For more information about PartiQL, see <https://partiql.org/>.

Notes and restrictions for column-level filtering

There are three ways to specify column filtering:

- By using data filters, as described earlier.
- By using simple column filtering.
- By using TAGs.

Simple column filtering just specifies a list of columns to include or exclude. Both the Lake Formation console, the API, and the AWS CLI support simple column filtering. For an example, see [Grant with Simple Column Filtering \(p. 191\)](#).

The following notes and restrictions apply to column filtering:

- AWS Glue ETL jobs support column filtering only by using data filters (cell-level security). The job fails if simple column filtering is applied to any table that the job references. If you want only column filtering, grant access to tables using data filters and enter true for the row filter expression in the console, or use AllRowsWildcard in your API calls.
- To grant SELECT with the grant option and column filtering, you must use an include list, not an exclude list. Without the grant option, you can use either include or exclude lists.
- To grant SELECT on a table with column filtering, you must have been granted SELECT on the table with the grant option and without any row restrictions. You must have access to all rows.
- If you grant SELECT with the grant option and column filtering to a principal in your account, that principal must specify column filtering for the same columns or a subset of the granted columns when granting to another principal. If you grant SELECT with the grant option and column filtering to an external account, the data lake administrator in the external account can grant SELECT on all columns to another principal in their account. However, even with SELECT on all columns, that principal will have visibility only on the columns granted to the external account.
- You can't apply column filtering on partition keys.
- A principal with the SELECT permission on a subset of columns in a table can't be granted the ALTER, DROP, DELETE, or INSERT permission on that table. For a principal with the ALTER, DROP, DELETE, or INSERT permission on a table, if you grant the SELECT permission with column filtering, it has no effect.

Notes and restrictions for row-Level and cell-level filtering

Keep in mind the following notes and restrictions for row-level and cell-level filtering:

- SELECT INTO statements are not supported.

- The struct, array, and map data types aren't supported in row filter expressions.
 - There is no limit to the number of data filters that can be defined on a table, but there is a limit of 100 data filter SELECT permissions for a single principal on a table.
 - To apply a data filter with a row filter expression, you must have SELECT with the grant option on all table columns. This restriction doesn't apply to administrators in external accounts when the grant was made to the external account.
 - If a principal is a member of a group and both the principal and the group are granted permissions on a subset of rows, the principal's effective row permissions are the union of the principal's permissions and the group's permissions.
 - The following column names are restricted in a table for row-level and cell-level filtering:
 - ctid
 - oid
 - xmin
 - cmin
 - xmax
 - cmax
 - tableoid
 - insertxid
 - deletexid
 - importoid
 - redcatuniqueid
 - If you apply the all-rows filter expression on a table concurrently with other filter expressions with predicates, the all-rows expression will prevail over all other filter expressions.
 - When permissions on a subset of rows are granted to an external AWS account and the data lake administrator of the external account grants those permissions to a principal in that account, the principal's effective filter predicate is the intersection of the account's predicate and any predicate that was directly granted to the principal.

For example, if the account has row permissions with the predicate `dept='hr'` and the principal was separately granted permission for `country='us'`, the principal has access only to rows with `dept='hr'` and `country='us'`.

For more information about cell-level filtering, see [Data filtering and cell-level security in Lake Formation \(p. 260\)](#).

Permissions required for querying tables with cell-level filtering

The following AWS Identity and Access Management (IAM) permissions are required to run queries against tables with cell-level filtering.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "lakeformation:StartQueryPlanning",  
                "lakeformation:GetQueryState",
```

```
        "lakeformation:GetWorkUnits",
        "lakeformation:GetWorkUnitResults"
    ],
    "Resource": "*"
}
]
```

For more information about Lake Formation permissions, see [Lake Formation Personas and IAM Permissions Reference \(p. 371\)](#).

Managing data filters

To implement column-level, row-level, and cell-level security, you can create and maintain data filters. Each data filter belongs to a Data Catalog table. You can create multiple data filters for a table, and then use one or more of them when granting permissions on the table.

You require SELECT permission with the grant option to create or view a data filter. To allow other principals or other AWS accounts to view and use a data filter, you can grant the DESCRIBE permission on it.

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

For information about data filters, see [Data filters in Lake Formation \(p. 261\)](#)

Creating a data filter

You can create one or more data filters for each Data Catalog table.

To create a data filter for a Data Catalog table (console)

1. Open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
Sign as a data lake administrator, the target table owner, or a principal who has a Lake Formation permission on the target table.
2. In the navigation pane, under **Data catalog**, choose **Data filters**.
3. On the **Data filters** page, choose **Create new filter**.
4. In the **Create data filter** dialog box, enter the following information:
 - Data filter name
 - Target database – Specify the database that contains the table.
 - Target table
 - Column-level access – Leave this set to **Access to all columns** to specify row filtering only. Choose **Include columns** or **Exclude columns** to specify column or cell filtering, and then specify the columns to include or exclude.
 - Row filter expression – Enter a filter expression to specify row or cell filtering. For supported data types and operators, see [PartiQL support in row filter expressions \(p. 263\)](#). If you do not want to use a row filter expression, enter true in the field.

The following screenshot shows a data filter that implements cell filtering. In queries against the `orders` table, it denies access to the `customer_name` column in any rows that have 'pharma' in the `product_type` column.

Create data filter

X

Data filter name

Enter a name that describes this data access filter.

restrict-pharma

Name may contain letters (A-Z), numbers (0-9), hyphens (-), or under-scores (_), and be less than 256 characters.

Target database

Select the database that contains the target table.

Choose databases



Load more

sales X

054881201579

Target table

Select the table for which the data filter will be created.

Choose tables



Load more

orders X

054881201579

Column-level access

Choose whether this filter should have column-level restrictions.

- Access to all columns
Filter won't have any column restrictions.
- Include columns
Filter will only allow access to specific columns.
- Exclude columns
Filter will allow access to all but specific columns.

Select columns

Choose one or more columns



customer_name X

string

Row filter expression

Enter the rest of the following query statement "SELECT * FROM orders WHERE..."

Please see the documentation for examples of filter expressions.

product_type='pharma'

5. Choose **Create filter**.

Granting data filter permissions

You can grant the SELECT, DESCRIBE and DROP Lake Formation permissions on data filters to principals.

At first, only you can view the data filters that you create for a table. To enable another principal to view a data filter and grant Data Catalog permissions with the data filter, you must either:

- Grant SELECT on a table to the principal with the grant option, and apply the data filter to the grant.
- Grant the DESCRIBE or DROP permission on the data filter to the principal.

You can grant the SELECT permission to an external AWS account. A data lake administrator in that account can then grant that permission to other principals in the account. When granting to an external account, you must include the grant option so that administrator of the external account can further cascade the permission to other users in his/her account. When granting to a principal in your account, granting with the grant option is optional.

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

Console

1. Sign in to the AWS Management Console and open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
2. In the navigation pane, under **Permissions**, choose **Data lake permissions**.
3. On the **Permissions** page, in the **Data permissions** section, choose **Grant**.
4. On the **Grant data permissions** page, choose the principals to grant the permissions to.
5. In the LF-Tags or catalog resources section, choose **Named data catalog resources**. Then choose the database, table, and data filter for which you want to grant permissions.

LF-Tags or catalog resources

Resources matched by LF-Tags (recommended)
Manage permissions indirectly for resources or data matched by a specific set of LF-Tags.

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

Databases
Select one or more databases.

Choose databases ▾ Load more

cloudtrail X
106567286946

Tables - optional
Select one or more tables.

Choose tables ▾ Load more

cloudtrail_logs_awslogs X
106567286946

Data filters - optional
Select one or more data filters.

Choose data filters ▾ Load more Create new

cloudtrail_lakeformation_filter X
106567286946

Manage data filters

- In the **Data filter permissions** section, choose the permissions you want to grant to the selected principals.

Data filter permissions

Data filter permissions
Choose specific access permissions to grant.

Select Describe Drop

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

Select Describe Drop

AWS CLI

- Enter a `grant-permissions` command. Specify `DataCellsFilter` for the `resource` argument, and specify `DESCRIBE` or `DROP` for the `Permissions` argument and, optionally, for the `PermissionsWithGrantOption` argument.

The following example grants `DESCRIBE` with the grant option to user `datalake_user1` on the data filter `restrict-pharma`, which belongs to the `orders` table in the `sales` database in AWS account `1111-2222-3333`.

```
aws lakeformation grant-permissions --cli-input-json file://grant-params.json
```

The following are the contents of file grant-params.json.

```
{  
    "Principal": {"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/datalake_user1"},  
    "Resource": {  
        "DataCellsFilter": {  
            "TableCatalogId": "111122223333",  
            "DatabaseName": "sales",  
            "TableName": "orders",  
            "Name": "restrict-pharma"  
        }  
    },  
    "Permissions": ["DESCRIBE"],  
    "PermissionsWithGrantOption": ["DESCRIBE"]  
}
```

Granting data permissions provided by data filters

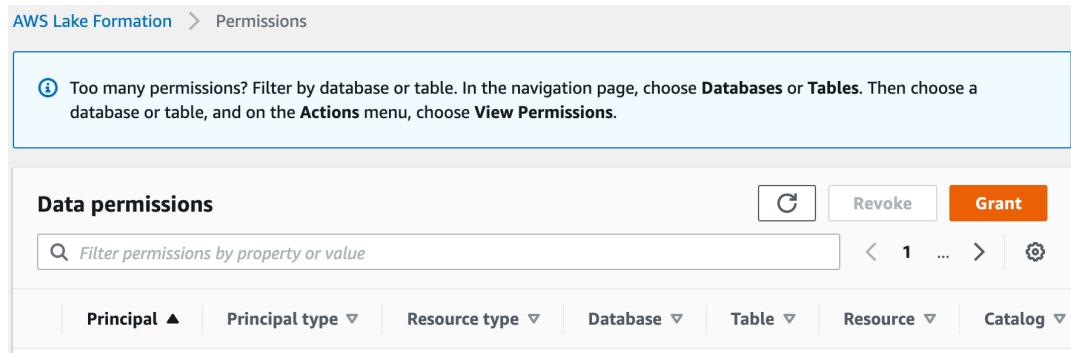
Data filters represent a subset of data within a table. To provide data access to principals, SELECT permissions need to be granted to those principals. With this permission the principals can:

- View the actual table name in list of tables shared with their account.
- Create data filters on the shared table and grant permissions to their users on those data filters.

Console

To grant SELECT permissions

1. Go to the **Permissions** page in the Lake Formation console, and then choose **Grant**.



2. Select the principals you want to provide access to, and select **Named data catalog resources**.

LF-Tags or catalog resources

Resources matched by LF-Tags (recommended)
Manage permissions indirectly for resources or data matched by a specific set of LF-Tags.

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

Databases
Select one or more databases.

cloudtrail X
106567286946
Load more

Tables - optional
Select one or more tables.

cloudtrail_logs_awslogs X
106567286946
Load more

Data filters - optional
Select one or more data filters.

cloudtrail_lakeformation_filter X
106567286946
Load more
Create new

[Manage data filters](#)

3. To provide access to the data that the filter represents, choose **Select** under **Data filter permissions**.

Data filter permissions

Data filter permissions
Choose specific access permissions to grant.

Select Describe Drop

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

Select Describe Drop

i Select permissions on data filters will grant access to the table 'cloudtrail_logs_awslogs'.

CLI

Enter a `grant-permissions` command. Specify `DataCellsFilter` for the resource argument, and specify `SELECT` for the Permissions argument.

The following example grants SELECT with the grant option to user `datalake_user1` on the data filter `restrict-pharma`, which belongs to the `orders` table in the `sales` database in AWS account `1111-2222-3333`.

```
aws lakeformation grant-permissions --cli-input-json file://grant-params.json
```

The following are the contents of file `grant-params.json`.

```
{  
    "Principal": {  
        "DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/datalake_user1"  
    },  
    "Resource": {  
        "DataCellsFilter": {  
            "TableCatalogId": "111122223333",  
            "DatabaseName": "sales",  
            "TableName": "orders",  
            "Name": "restrict-pharma"  
        }  
    },  
    "Permissions": ["SELECT"]  
}
```

Viewing data filters

You can use the Lake Formation console, AWS CLI, or the Lake Formation API to view data filters.

To view data filters, you must be a Data Lake administrator or have the required permissions on the data filters.

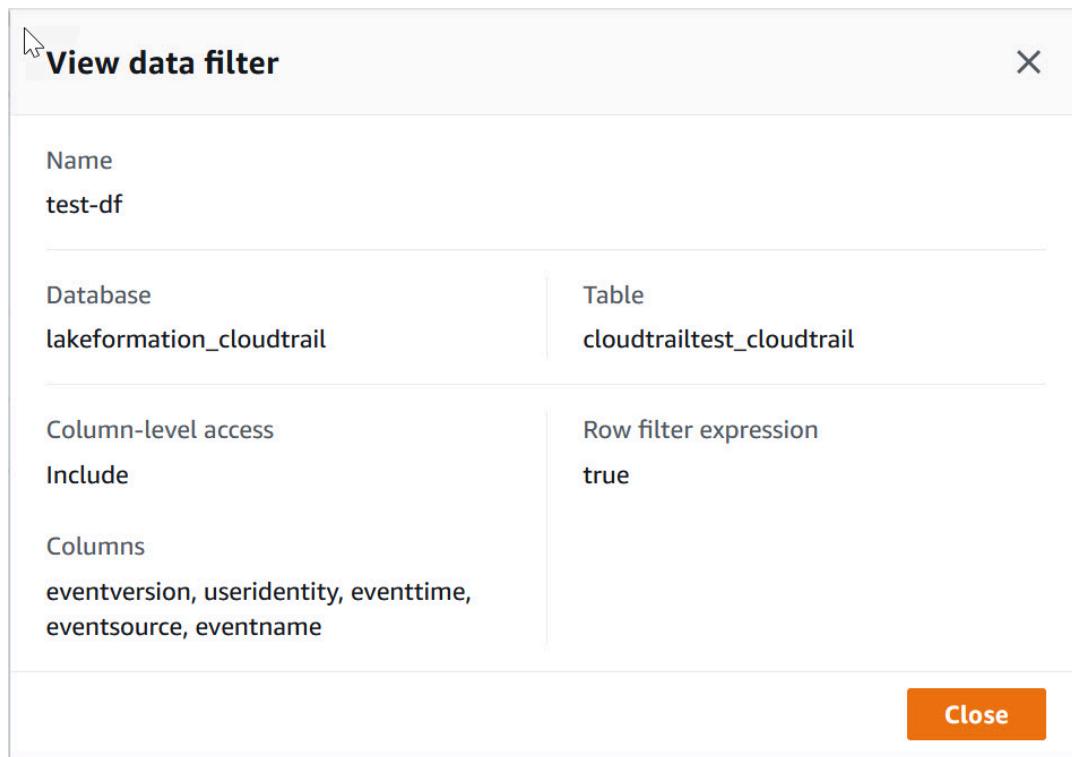
Console

1. Sign in to the AWS Management Console and open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
2. In the navigation pane, under **Data catalog**, choose **Data filters**.

The page displays the data filters you have access to.

| Data filters (1) | | | | <input type="button" value="Create new filter"/> |
|--|---------------------------|--------------------------|------------------|--|
| <input type="text"/> Find filter | | | | <input type="button" value="View"/> |
| Filter name | Table | Database | Table catalog ID | |
| <input checked="" type="radio"/> test-df | cloudtrailtest_cloudtrail | lakeformation_cloudtrail | | <input type="button" value=""/> |

3. To view the data filter details, choose the data filter, and then choose View. A new window appears with the data filter detailed information.



AWS CLI

Enter a `list-data-cells-filter` command and specify a table resource.

The following example lists the data filters for the `cloudtrailtest_cloudtrail` table.

```
aws lakeformation list-data-cells-filter --table '{ "CatalogId": "123456789012", "DatabaseName": "lakeformation_cloudtrail", "Name": "cloudtrailtest_cloudtrail" }'
```

API/SDK

Use the `ListDataCellsFilter` API and specify a table resource.

The following example uses Python to list the first 20 data filters for the `myTable` table.

```
response = client.list_data_cells_filter(
    Table = {
        'CatalogId': '111122223333',
        'DatabaseName': 'mydb',
        'Name': 'myTable'
    },
    MaxResults=20
)
```

Listing data filter permissions

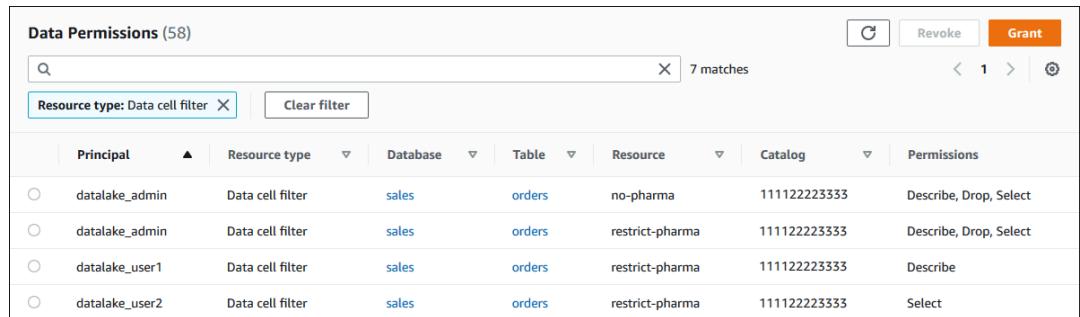
You can use the Lake Formation console to view the permissions granted on data filters.

To view permissions on a data filter, you must be a Data Lake administrator or have the required permissions on the data filter.

Console

1. Sign in to the AWS Management Console and open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
2. In the navigation pane, under **Permissions**, choose **Data permissions**.
3. On the **Data Permissions** page, click or tap in the search field, and on the **Properties** menu, choose **Resource type**.
4. On the **Resource type** menu, choose **Resource type: Data cell filter**.

The data filters that you have permissions on are listed. You might have to scroll horizontally to see the **Permissions** and **Grantable** columns.



The screenshot shows a table titled "Data Permissions (58)" with a search bar and filter buttons. The table has columns: Principal, Resource type, Database, Table, Resource, Catalog, and Permissions. There are 7 matches. The data shows four entries for "datalake_admin" and two for "datalake_user1" and "datalake_user2", all granting "Describe, Drop, Select" or "Select" permissions on tables in the "sales" database.

| Principal | Resource type | Database | Table | Resource | Catalog | Permissions |
|----------------|------------------|----------|--------|-----------------|--------------|------------------------|
| datalake_admin | Data cell filter | sales | orders | no-pharma | 111122223333 | Describe, Drop, Select |
| datalake_admin | Data cell filter | sales | orders | restrict-pharma | 111122223333 | Describe, Drop, Select |
| datalake_user1 | Data cell filter | sales | orders | restrict-pharma | 111122223333 | Describe |
| datalake_user2 | Data cell filter | sales | orders | restrict-pharma | 111122223333 | Select |

AWS CLI

- Enter a `list-permissions` command. Specify `DataCellsFilter` for the `resource` argument, and specify `DESCRIBE` or `DROP` for the `Permissions` argument and, optionally, for the `PermissionsWithGrantOption` argument.

The following example lists `DESCRIBE` permissions with the grant option on the data filter `restrict-pharma`. The results are limited to permissions granted for the principal `datalake_user1` and the `orders` table in the `sales` database in AWS account `1111-2222-3333`.

```
aws lakeformation list-permissions --cli-input-json file://list-params.json
```

The following are the contents of file `grant-params.json`.

```
{
    "Principal": {"DataLakePrincipalIdentifier": "arn:aws:iam::111122223333:user/datalake_user1"},
    "Resource": {
        "DataCellsFilter": {
            "TableCatalogId": "111122223333",
            "DatabaseName": "sales",
            "TableName": "orders",
            "Name": "restrict-pharma"
        }
    },
    "Permissions": ["DESCRIBE"],
    "PermissionsWithGrantOption": ["DESCRIBE"]
}
```

Reading from and writing to the data lake within transactions

AWS Lake Formation supports ACID (atomic, consistent, isolated, and durable) transactions when reading and writing governed tables composed of Amazon S3 objects, and when creating and updating table metadata in your Data Catalog. Transactions maintain the integrity of governed table manifests (*transaction data operations*) and of other table metadata such as schema (*transactional metadata operations*). The following are typical use cases for transactions against governed tables:

- **ETL into new tables** – In this use case, you might have an AWS Glue extract, transform, and load (ETL) job that starts a transaction, reads from a data source, writes to a data sink that is a registered Amazon S3 location in the data lake, and creates a governed table in the Data Catalog for the data sink. If the ETL script detects a failure at some point, the script can cancel the transaction, and as a result, the following occurs:
 - The governed table is deleted from the catalog.
 - If the script invokes the Lake Formation `DeleteObjectsOnCancel` API operation before each new object is written to Amazon S3, then Lake Formation also deletes all the objects that were written to Amazon S3 within the transaction. For more information, see [Rolling back Amazon S3 writes \(p. 277\)](#).
- **Table updates** – Assume that for an existing governed table, your ETL job starts a transaction, writes new objects to Amazon S3 and updates the table manifest by using the `UpdateTableObjects` API operation. If the script detects a failure, it can cancel the transaction, and as a result, the following occurs:
 - The table manifest is restored to its state before the transaction started.
 - If the script invokes the Lake Formation `DeleteObjectsOnCancel` API operation before each new object is written to Amazon S3, Lake Formation also deletes all the objects that were written to Amazon S3 within the transaction.
- **Schema updates** – For an existing governed table with a streaming data sink in Amazon S3, if the streaming ETL job determines that there are additional table columns in the data, it can update the table schema within a transaction. If a failure occurs, the job can cancel the transaction, in which case the table schema is restored to its state before the transaction started.
- **Time-travel queries** – Lake Formation maintains multiple versions (*snapshots*) of table metadata as the data in the data lake changes. You can travel back in time and query the data even if the schema has changed.

For more information about governed tables, see [Governed tables in Lake Formation \(p. 120\)](#).

Topics

- [Transactional data operations \(p. 277\)](#)
- [Commit process in governed table \(p. 277\)](#)
- [Rolling back Amazon S3 writes \(p. 277\)](#)
- [Transactional metadata operations \(p. 278\)](#)
- [Lake Formation API operations that support transactions \(p. 280\)](#)
- [Coding best practices for transactions \(p. 280\)](#)
- [Data lake transactions code samples \(p. 281\)](#)

Transactional data operations

Lake Formation ACID transactions provide snapshot isolation, which enables multiple jobs to concurrently and reliably update governed tables by adding and removing Amazon S3 objects, while maintaining read consistency for each query against the data lake.

An example of the read consistency that ACID transactions provide is when a user starts a query in Amazon Athena that scans all partitions of a table to aggregate sales figures. While the scan is proceeding and before it completes, an ETL job adds a partition to the table. If the user performed the query within a transaction, they will see query results that reflect the state of the table at the start of the query. The results won't include data from the new partition created by the job.

Lake Formation operations that take part in transactions access and modify governed tables in the AWS Glue Data Catalog. A transaction can involve multiple governed tables.

Transactions also provide ACID properties for changes involving the metadata that defines the Amazon S3 objects that make up a governed table (the *manifest*), as well as the table schema. For more information about governed tables, see [Governed tables in Lake Formation \(p. 120\)](#).

Integrated AWS services such as Athena automatically perform queries within transactions when governed tables are involved. To use transactions in your AWS Glue ETL jobs, the job script begins a transaction before performing any reads or writes to the data lake, references the transaction ID for any operation within the transaction, and commits the transaction upon completion. If a read or write operation fails, the job script can retry the operation, or optionally cancel the transaction. You can also use transactions in your queries by specifying the past point in time to read from.

Lake Formation might automatically cancel a transaction if Lake Formation detects certain failures, such as conflicting writes. Canceling a transaction causes all operations to be rolled back, as described in [Reading from and writing to the data lake within transactions \(p. 276\)](#).

To enable Lake Formation to distinguish long-running transactions (for example, Spark ETL jobs that run for many hours) from transactions that are abandoned due to crashes, your long-running write transactions should call the heartbeat API operation `ExtendTransaction` regularly. This isn't necessary for read-only transactions. Lake Formation automatically cancels transactions that are idle for too long.

Commit process in governed table

Modifications to governed tables must be made within the context of a transaction. If your ETL job performs an operation on a governed table without explicitly providing a transaction ID, Lake Formation automatically starts a transaction and commits (or cancels) the transaction at the end of the operation. This is referred to as a single-statement transaction.

For a transaction that has write operations, a call to `CommitTransaction` will move the transaction to `COMMIT_IN_PROGRESS` state. An internal background process works on applying the changes in the transaction to the governed table before moving the transaction to `COMMITTED` status. Therefore, a read operation made immediately after calling `CommitTransaction` may or may not reflect the result of the write operation. In order to deterministically read the result of the write operation, customers should wait until the status of transaction changes to `COMMITTED`. This can be checked by either calling `CommitTransaction` or `DescribeTransaction` API operation. Read operation for a single-statement transaction also demonstrates the same behavior.

Rolling back Amazon S3 writes

When a transaction is canceled, either automatically or by a call to `CancelTransaction`, Lake Formation never deletes data that was written to Amazon S3 without your permission. To grant permission to Lake

Formation to roll back writes made during a transaction, your code must call the [DeleteObjectsOnCancel API operation](#), which lists the Amazon S3 objects that can be deleted if the transaction is canceled. It's recommended that you call `DeleteObjectsOnCancel` before making the writes.

The AWS Glue ETL library function `write_dynamic_frame.from_catalog()` includes an option to automatically call `DeleteObjectsOnCancel` before writes. In the following example, the `callDeleteObjectsOnCancel` option is included in the `additional_options` argument. Because the value `False` is passed to the `read_only` argument of `start_transaction`, the transaction is not a read-only transaction.

```
transactionId = glueContext.start_transaction(False)

try:
    datasink0 = glueContext.write_dynamic_frame.from_catalog(
        frame = datasource0,
        database="MyDatabase",
        table_name="MyGovernedTable",
        additional_options={
            "partitionKeys": ["key1", "key2"],
            "transactionId": transactionId,
            "callDeleteObjectsOnCancel": "true"
        }
    )
    glueContext.commit_transaction(transactionId)

except:
    glueContext.cancel_transaction(transactionId)
```

Transactional metadata operations

The metadata for governed tables includes the following:

- Table schema
- Manifest of Amazon S3 table objects
- Version history
- Permissions

When making a change to the manifest for a governed table, applications can also change the schema in your Data Catalog transactionally. As a result of providing transactions on metadata, you can do time travel queries on data, and can query data as it existed at a previous time.

For example, the AWS Glue table API operations take an optional transaction ID to allow applications to create, update, or delete the metadata for a governed table. This is the transaction ID returned by the `StartTransaction` API.

When a transaction is committed, a *snapshot* is created of the table metadata and associated with the transaction ID. The API operations can refer to a specific snapshot by passing in the transaction ID. For example, you call `UpdateTable` with transaction ID abc followed by another update xyz to change the schema. You can then call `GetTable` with transaction ID abc, which receives the schema as it was when transaction ID abc was committed.

Another use case for transactional metadata is to perform a time travel query to read the metadata of a table as of a specified time using the `GetTable` or `GetTables` operations.

API Operations that support transaction metadata operations

The following AWS Glue API operations support transactional metadata operations. For each description, it's assumed that transaction IDs (returned by `StartTransaction` or `start_transaction`) are passed to the API call, and that at some point, there is a commit.

| API Operation | Description |
|---|---|
| CreateTable | Creates a table in the Data Catalog when you commit the transaction (when you call the <code>CommitTransaction</code> API with the transaction ID). |
| UpdateTable | Updates the metadata for a table. After the transaction is committed, creates a new snapshot of the table metadata. The snapshot is indexed by the transaction ID. |
| DeleteTable , BatchDeleteTable | Deletes tables when you commit the transaction. All snapshots are deleted. |
| GetTable | <p>Optionally takes either a <code>TransactionId</code> or an <code>QueryAsOfTime</code> parameter.</p> <p>If you pass a <code>TransactionId</code>, the operation retrieves that matching snapshot and returns the table metadata as of the time that transaction was committed.</p> <p>If you pass an <code>QueryAsOfTime</code> parameter, returns the table metadata as of the specified time.</p> |
| GetTables | <p>Optionally takes either a <code>TransactionId</code> or an <code>QueryAsOfTime</code> parameter.</p> <p>If neither parameter is passed, the operation returns both governed and ungoverned tables. If you pass either a <code>TransactionId</code> or an <code>QueryAsOfTime</code> parameter, returns only governed tables.</p> <p>If you pass a <code>TransactionId</code>, returns the metadata for the tables as of the time the transaction was committed.</p> <p>If you pass the <code>QueryAsOfTime</code> parameter, the operation returns the metadata for the tables as of the specified time.</p> |

Limitations on transactional metadata operations

The following are current limitations of transactional metadata operations:

- Lake Formation doesn't support transactional operations when adding or modifying databases in the Data Catalog, for example, `CreateDatabase`, `UpdateDatabase`, `GetDatabase`, etc.
- Lake Formation doesn't support time travel reads for modifications of databases in the Data Catalog.
- Lake Formation doesn't support transactional operations when adding or modifying table partitions, for example `CreatePartition`, `BatchUpdatePartition`, etc. Partition keys can be identified by looking at the table objects.

Lake Formation API operations that support transactions

Lake Formation makes available the following API operations to support transactions:

| API Operation | Description |
|---------------------------------------|--|
| StartTransaction | Starts a new transaction and returns its transaction ID. |
| CommitTransaction | Attempts to commit the transaction with the specified TransactionId. If successful, all modifications made during the transaction are persisted. |
| CancelTransaction | Stops the transaction associated with the specified TransactionId. All modifications made during the transaction are rolled back. |
| ExtendTransaction | Indicates that the specified transaction (TransactionId) is still active, and should not be canceled. The current timeout period for idle transactions is 30 seconds. Not needed for read-only transactions. |
| DescribeTransaction | Returns the status of the transaction identified by its TransactionId (ACTIVE, COMMITTED, or ABORTED). |
| ListTransactions | Returns the transaction ID, status, start and end time for uncommitted transactions and transactions available for time-travel. |
| DeleteObjectsOnCancel | Returns a list of Amazon S3 objects that will be written during the current transaction (specified by TransactionId). Lake Formation automatically deletes these objects if the transaction is canceled. |
| GetTableObjects | Returns the set of Amazon S3 objects that make up the specified governed table. A transaction ID or timestamp can be specified for time-travel queries. |
| UpdateTableObjects | Updates the manifest of Amazon S3 objects that make up the specified governed table. |

Coding best practices for transactions

The following are a few best practices for coding your AWS Glue ETL scripts for transactions.

- When you start a transaction, ensure that there is exception handling that will cancel transactions in the event of any exceptions. See [Rolling back Amazon S3 writes \(p. 277\)](#) for an example.
- If your job uses any of the APIs listed in [API Operations that support transaction metadata operations \(p. 278\)](#), specify a transaction ID to ensure the transaction can be cleaned up if it fails or is canceled.
- Keep in mind that the system can cancel your transaction for any number of reasons. Use the ExtendTransaction API to prevent long running transactions from being canceled.
- If you attempt an operation within a canceled transaction, you get a TransactionCanceledException, so your code should handle that exception. You can check the status of the transaction before attempting the operation with the DescribeTransaction API operation.
- For managing transactions within an AWS Glue ETL job script, see the transaction functions in the GlueContext for either [Python](#) or [Scala](#).

Data lake transactions code samples

The following PySpark and Java code examples demonstrate how to start and commit a transaction, and how to stop and roll back a transaction if an exception occurs.

Note

For writing Apache Parquet, AWS Glue ETL only supports writing to a governed table by specifying an option for a custom Parquet writer type optimized for Dynamic Frames.

When writing to a governed table with the parquet format, you should add the key `useGlueParquetWriter` with a value of `true` in the table parameters.

PySpark

This example shows the ETL script for an AWS Glue job. It copies data from a non-governed table to a governed table. The governed table needs to be created following the instructions at [Creating governed tables \(p. 123\)](#) or by using Amazon Athena.

```
import sys
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

db = "retail"
tbl = "inventory"
tx_id = glueContext.start_transaction(False)

datasource0 = glueContext.create_dynamic_frame.from_catalog(
    database = db, table_name = tbl,
    transformation_ctx = "datasource0")
datasource0.show()

dest_path = "s3://path_to_sales_retail_data/"

try:
    glueContext.write_dynamic_frame.from_catalog(
        frame = datasource0,
        database = "retail",
        table_name = "inventory-governed",
        additional_options = {
            "transactionId":tx_id
        }
    )
    glueContext.commit_transaction(tx_id)
except Exception:
    glueContext.cancel_transaction(tx_id)
    raise
job.commit()
```

This next example demonstrates the use of transactions in an AWS Glue streaming ETL job.

```
import sys
from awsglue.transforms import ApplyMapping
from awsglue.utils import getResolvedOptions
```

```

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.dynamicframe import DynamicFrame

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

data_frame_DataSource0 = glueContext.create_data_frame.from_catalog(
    database = "demo",
    table_name = "kinesis_cLOUDTRAIL_demo",
    transformation_ctx = "DataSource0",
    additional_options = {
        "startingPosition": "TRIM_HORIZON",
        "inferSchema": "true"
    }
)

def processBatch(data_frame, batchId):
    if (data_frame.count() > 0):
        dynamic_frame = DynamicFrame.fromDF(data_frame, glueContext, "from_data_frame")
        dynamic_frame = ApplyMapping.apply(
            frame = dynamic_frame,
            mappings = [
                ("eventversion", "string", "eventversion", "string"),
                ("eventtime", "string", "eventtime", "string"),
                ("eventsoure", "string", "eventsoure", "string"),
                ("eventname", "string", "eventname", "string"),
                ("awsregion", "string", "awsregion", "string"),
                ("sourceipaddress", "string", "sourceipaddress", "string"),
                ("useragent", "string", "useragent", "string"),
                ("errorcode", "string", "errorcode", "string"),
                ("errormessage", "string", "errormessage", "string"),
                ("requestid", "string", "requestid", "string"),
                ("eventid", "string", "eventid", "string"),
                ("eventtype", "string", "eventtype", "string"),
                ("apiversion", "string", "apiversion", "string"),
                ("readonly", "boolean", "readonly", "string"),
                ("recipientaccountid", "string", "recipientaccountid", "string"),
                ("sharedeventid", "string", "sharedeventid", "string"),
                ("vpcendpointid", "string", "vpcendpointid", "string")
            ],
            transformation_ctx = "ApplyMapping"
        )

        table_name = "cloudtrail_demo_sample"
        txId = glueContext.begin_transaction(False)
        try:
            glueContext.write_dynamic_frame.from_catalog(
                frame = dynamic_frame,
                database = "demo",
                table_name = table_name,
                additional_options = {
                    "transactionId":txId
                }
            )
            glueContext.commit_transaction(txId)
        except Exception:
            glueContext.cancel_transaction(txId)
            raise

```

```

glueContext.forEachBatch(
    frame = data_frame_DataSource0,
    batch_function = processBatch,
    options = {
        "windowSize": "100 seconds",
        "checkpointLocation": "s3://my_checkpoint_bucket/cloudtrail_demo_sample/"
    }
)
job.commit()

```

Java

The following two Java examples use the `getTableObjects` and `updateTableObjects` object API operations for governed tables within a transaction. For information about these API operations, see the [API documentation](#).

This first example adds information about a new partition to an existing governed table.

Imports

```

import com.amazonaws.services.lakeformation.AWSLakeFormation;
import com.amazonaws.services.lakeformation.AWSLakeFormationClientBuilder;
import com.amazonaws.services.lakeformation.model.CancelTransactionRequest;
import com.amazonaws.services.lakeformation.model.AddObjectInput;
import com.amazonaws.services.lakeformation.model.StartTransactionRequest;
import com.amazonaws.services.lakeformation.model.StartTransactionResult;
import com.amazonaws.services.lakeformation.model.CommitTransactionRequest;
import com.amazonaws.services.lakeformation.model.UpdateTableObjectsRequest;
import com.amazonaws.services.lakeformation.model.WriteOperation;

```

Code

```

AWSLakeFormation lake = AWSLakeFormationClientBuilder.standard().build();
// Begin transaction
BeginTransactionResult startTransactionResult = lake.startTransaction(new
    StartTransactionRequest());
String transactionId = startTransactionResult.getTransactionId();
// Construct a write operation
WriteOperation write = new WriteOperation()
    .withAddObject(new AddObjectInput()
        .withPartitionValues("par0", "par1")
        .withUri("s3://bucket/prefix")
        .withETag("eTag")
        .withSize(100L));
// Save a partition table object
try {
    lake.updateTableObjects(new UpdateTableObjectsRequest()
        .withDatabaseName(databaseName)
        .withTableName(tableName)
        .withTransactionId(transactionId)
        .withWriteOperations(write));
    // Commit transaction
    lake.commitTransaction(new
        CommitTransactionRequest().withTransactionId(transactionId));
} catch (Exception e) {
    // Abort transaction
    lake.cancelTransaction(new
        CancelTransactionRequest().withTransactionId(transactionId));
}

```

This next example retrieves all of the Amazon S3 objects associated with a governed table and processes them.

Imports

```
import com.amazonaws.services.lakeformation.AWSLakeFormation;
import com.amazonaws.services.lakeformation.AWSLakeFormationClientBuilder;
import com.amazonaws.services.lakeformation.model.StartTransactionRequest;
import com.amazonaws.services.lakeformation.model.StartTransactionResult;
import com.amazonaws.services.lakeformation.model.CommitTransactionRequest;
import com.amazonaws.services.lakeformation.model.GetTableObjectsRequest;
import com.amazonaws.services.lakeformation.model.GetTableObjectsResult;
import com.amazonaws.services.lakeformation.model.PartitionObjects;
import com.amazonaws.services.lakeformation.model.TableObject;
```

Code

```
AWSLakeFormation lake = AWSLakeFormationClientBuilder.standard().build();
// Start read only transaction
StartTransactionResult startTransactionResult = lake.startTransaction(new
    StartTransactionRequest().withReadOnly(true));
String transactionId = startTransactionResult.getTransactionId();
// Read all table objects from a table
GetTableObjectsResult getTableObjectsResult = lake.getTableObjects(
    new GetTableObjectsRequest()
        .withTransactionId(transactionId)
        .withDatabaseName(databaseName)
        .withTableName(tableName));
for (PartitionObjects partitionObjects: getTableObjectsResult.getObjects()) {
    for (TableObject tableObject: partitionObjects.getObjects()) {
        // do something with the data
    }
}
// Commit transaction
lake.commitTransaction(new
    CommitTransactionRequest().withTransactionId(transactionId));
```

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 \(p. 285\)](#)
- [Infrastructure Security in AWS Lake Formation \(p. 286\)](#)
- [Cross-service confused deputy prevention \(p. 288\)](#)
- [Security and access Control to metadata and data in Lake Formation \(p. 289\)](#)
- [Security Event Logging in AWS Lake Formation \(p. 302\)](#)
- [Using Service-Linked Roles for Lake Formation \(p. 302\)](#)

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 confidential or sensitive information, such as your customers' email addresses, into tags or 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 tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at Rest

AWS Lake Formation supports data encryption in the following areas:

- Data in your Amazon Simple Storage Service (Amazon S3) data lake.

Lake Formation supports data encryption with [AWS Key Management Service \(AWS KMS\)](#). Data is typically written to the data lake by means of AWS Glue extract, transform, and load (ETL) jobs. For information about how to encrypt data written by AWS Glue jobs, see [Encrypting Data Written by Crawlers, Jobs, and Development Endpoints](#) in the [AWS Glue Developer Guide](#).

- The AWS Glue Data Catalog, which is where Lake Formation stores metadata tables that describe data in the data lake.

For more information, see [Encrypting Your Data Catalog](#) in the [AWS Glue Developer Guide](#).

To add an Amazon S3 location as storage in your data lake, you *register* the location with AWS Lake Formation. You can then use Lake Formation permissions for fine-grained access control to AWS Glue Data Catalog objects that point to this location, and to the underlying data in the location.

Lake Formation supports registering an Amazon S3 location that contains encrypted data. For more information, see [Registering an encrypted Amazon S3 location \(p. 110\)](#).

Infrastructure Security in AWS Lake Formation

As a managed service, AWS Lake Formation is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

You use AWS published API calls to access Lake Formation through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service \(AWS STS\)](#) to generate temporary security credentials to sign requests.

Topics

- [AWS Lake Formation and interface VPC endpoints \(AWS PrivateLink\) \(p. 287\)](#)

AWS Lake Formation and interface VPC endpoints (AWS PrivateLink)

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

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

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

Each interface endpoint is represented by one or more [Elastic Network Interfaces](#) in your subnets.

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

Considerations for Lake Formation VPC endpoints

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

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

Creating an interface VPC endpoint for Lake Formation

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

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

- com.amazonaws.*region*.lakeformation

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

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

Creating a VPC endpoint policy for Lake Formation

Lake Formation supports VPC endpoint policies. A VPC endpoint policy is an AWS Identity and Access Management (IAM) resource policy that you attach to an endpoint when you create or modify the endpoint.

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

- The principal that can perform actions.
- The actions that can be performed.

- The resources on which actions can be performed.

For more information, see [Controlling access to services with VPC endpoints](#) in the *Amazon VPC User Guide*.

Example: VPC endpoint policy for Lake Formation actions

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

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

Note

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

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

- [What Is Amazon VPC?](#)
- [Create an Interface Endpoint](#)
- [Use VPC endpoint policies](#)

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that AWS Lake Formation gives another service to the resource. If you use both global condition context keys, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

Currently, Lake Formation only supports `aws:SourceArn` in the following format:

```
arn:aws:lakeformation:aws-region:account-id:*
```

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in Lake Formation to prevent the confused deputy problem.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "lakeformation.amazonaws.com"
    },
    "Action": [
      "sts:AssumeRole"
    ],
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "account-id"
      },
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:lakeformation:aws-region:account-id:/*"
      }
    }
  }
]
```

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 [AWS service integrations with Lake Formation \(p. 3\)](#).

Topics

- [Lake Formation access control overview \(p. 290\)](#)
- [AWS managed policies for Lake Formation \(p. 298\)](#)
- [Changing the default security settings for your data lake \(p. 299\)](#)
- [Permissions example scenario \(p. 301\)](#)

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*).

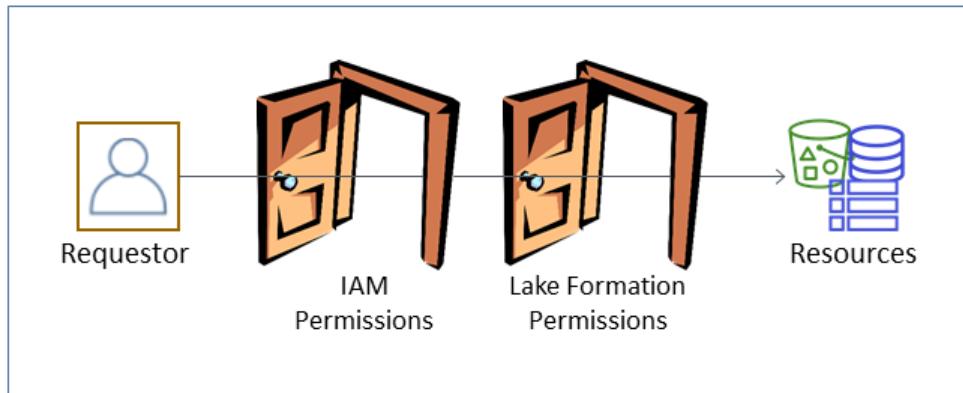
These permissions enable principals to create, read, update, and delete metadata databases and tables in the Data Catalog.

- **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 \(p. 291\)](#)
- [Metadata access control \(p. 292\)](#)
- [Underlying data access control \(p. 294\)](#)

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>This is the default method for backward compatibility with AWS Glue.</p> <ul style="list-style-type: none"> • <i>Open</i> means that the special permission Super is granted to the group IAMAllowedPrincipals, where IAMAllowedPrincipals is automatically created and includes any IAM users and roles that are allowed access to your Data Catalog resources by your IAM policies, and the Super permission enables a principal to perform every supported Lake Formation operation on the database or table on which it is granted. This effectively causes access to Data Catalog resources and Amazon S3 locations to be controlled solely by IAM policies. For more information, see Changing the default security settings for your data lake (p. 299) and Upgrading AWS Glue data permissions to the AWS Lake Formation model (p. 23). • <i>Fine-grained</i> means that IAM policies control all access to Data Catalog resources and to individual Amazon S3 buckets. <p>On the Lake Formation console, this method appears as Use only IAM access control.</p> |
| Method 2 | Fine-grained | Coarse-grained | <p>This is the recommended method.</p> <ul style="list-style-type: none"> • <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. • <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 |

| Method | Lake Formation Permissions | IAM Permissions | Comments |
|--------|----------------------------|-----------------|---|
| | | | "glue:/*" or "glue:Create*" rather than "glue:CreateTables", leaving Lake Formation permissions to control whether or not a principal can create catalog objects. It also means giving principals access to the APIs that they need to do their work, but locking down other APIs and resources. For example, you might create an IAM policy that enables a principal to create Data Catalog resources and create and run workflows, but doesn't enable creation of AWS Glue connections or user-defined functions. See the examples later in this section. |

Important

Be aware of the following:

- By default, Lake Formation has the **Use only IAM access control** settings enabled for compatibility with existing AWS Glue Data Catalog behavior. We recommend that you disable these settings after you transition to using Lake Formation permissions. For more information, see [Changing the default security settings for your data lake \(p. 299\)](#).
- Data lake administrators and database creators have implicit Lake Formation permissions that you must understand. For more information, see [Implicit Lake Formation permissions \(p. 173\)](#).

Metadata access control

For access control for Data Catalog resources, the following discussion assumes fine-grained access control with Lake Formation permissions and coarse-grained access control with IAM policies.

There are two distinct methods for granting Lake Formation permissions on Data Catalog resources:

- **Named resource access control** – With this method, you grant permissions on specific databases or tables by specifying database or table names. The grants have this form:

Grant permissions to principals on resources [with grant option].

With the grant option, you can allow the grantee to grant the permissions to other principals.

- **Tag-based access control** – With this method, you assign one or more *LF-tags* to Data Catalog databases, tables, and columns, and grant permissions on one or more LF-tags to principals. Each LF-tag is a key-value pair, such as department=sales. A principal that has LF-tags that match the LF-tags on a Data Catalog resource can access that resource. This method is recommended for data lakes with a large number of databases and tables. It's explained in detail in [Overview of Lake Formation tag-based access control \(p. 231\)](#).

The permissions that a principal has on a resource is the union of the permissions granted by both the methods.

The following table summarizes the available Lake Formation permissions on Data Catalog resources. The column headings indicate the resource on which the permission is granted.

| Catalog | Database | Table |
|-----------------|--------------|----------|
| CREATE_DATABASE | CREATE_TABLE | ALTER |
| | ALTER | DROP |
| | DROP | DESCRIBE |
| | DESCRIBE | SELECT* |
| | | INSERT* |
| | | DELETE* |

For example, the CREATE_TABLE permission is granted on a database. This means that the principal is allowed to create tables in that database.

The permissions with an asterisk (*) are granted on Data Catalog resources, but they apply to the underlying data. For example, the DROP permission on a metadata table enables you to drop the table from the Data Catalog. However, the DELETE permission granted on the same table enables you to delete the table's underlying data in Amazon S3, using, for example, a SQL DELETE statement. With these permissions, you also can view the table on the Lake Formation console and retrieve information about the table with the AWS Glue API. Thus, SELECT, INSERT, and DELETE are both Data Catalog permissions and data access permissions.

When granting SELECT on a table, you can add a filter that includes or excludes one or more columns. This permits fine-grained access control on metadata table columns, limiting the columns that users of integrated services can see when running queries. This capability is not available using just IAM policies.

There is also a special permission named Super. The Super permission enables a principal to perform every supported Lake Formation operation on the database or table on which it is granted. This permission can coexist with the other Lake Formation permissions. For example, you can grant Super, SELECT, and INSERT on a metadata table. The principal can perform all supported actions on the table, and when you revoke Super, the SELECT and INSERT permissions remain.

For details on each permission, see [Lake Formation Permissions Reference \(p. 219\)](#).

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 Persons and IAM Permissions Reference \(p. 371\)](#).

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 S3 location [with grant option]
```

If you include `with grant option`, the grantee can grant the permissions to other principals.

Recall that Lake Formation permissions always work in combination with AWS Identity and Access Management (IAM) permissions for fine-grained access control. For read/write permissions on underlying Amazon S3 data, IAM permissions are granted as follows:

When you register a location, you specify an IAM role that grants read/write permissions on that location. Lake Formation assumes that role when supplying temporary credentials to integrated AWS services. A typical role might have the following policy attached, where the registered location is the bucket `awsexamplebucket`.

```
{
    "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 [Using Service-Linked Roles for Lake Formation \(p. 302\)](#).

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 \(p. 371\)](#).

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

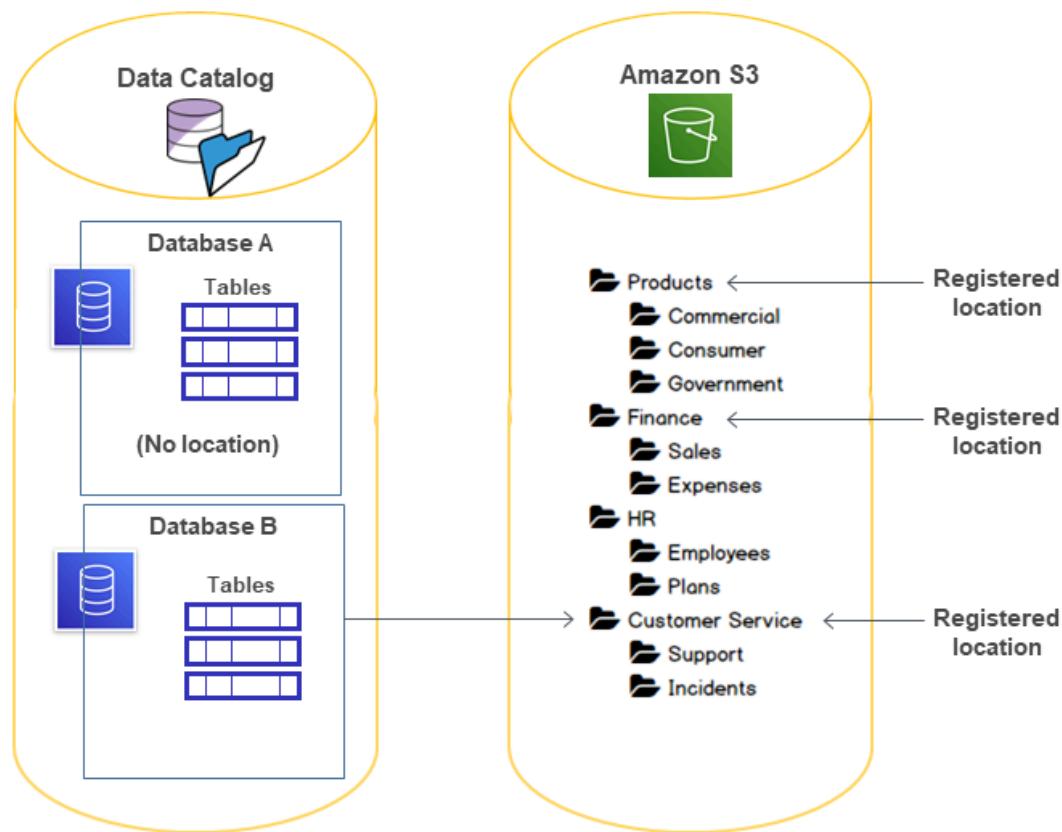
The Lake Formation permissions model doesn't prevent access to Amazon S3 locations through the Amazon S3 API or console if you have access to them through IAM or Amazon S3 policies. You can attach IAM policies to principals to block this access.

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 Finance/Sales | Fails | No data location permission |
| Database A at Products | Succeeds | Has data location permission |

| Database and Location | Succeeds or Fails | Reason |
|--|-------------------|--|
| Database A at HR/Plans | Succeeds | Location is not registered |
| Database B at Customer Service/ Incidents | Succeeds | Database has location property at Customer Service |

For more information, see the following:

- [Adding an Amazon S3 location to your data lake \(p. 107\)](#)
- [Lake Formation Permissions Reference \(p. 219\)](#)
- [Lake Formation Personas and IAM Permissions Reference \(p. 371\)](#)

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 | Description |
|--|-------------------------------------|--|
| Lake Formation user, including the data lake administrator | AWSGlueConsoleFullAccess | Allows the principal to conduct a variety of operations on the Lake Formation console. |
| Data lake administrators | AWSLakeFormationDataAdmin | Allows the data lake administrator to conduct administrative operations and view AWS CloudTrail logs. |
| Lake Formation user, including the data lake administrator | AWSLakeFormationCrossAccountManager | 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 [Create a data lake administrator \(p. 12\)](#) and [Using Service-Linked Roles for Lake Formation \(p. 302\)](#).

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.

Lake Formation updates to AWS managed policies

View details about updates to AWS managed policies for Lake Formation since this service began tracking these changes.

| Change | Description | Date |
|---|--|-------------|
| Lake Formation updated AWSLakeFormationCrossAccountPolicy policy. | Lake Formation enhanced the AWSLakeFormationCrossAccountManager policy to create only one resource share per recipient account when a resource is first shared. All resources shared thereafter with the same account are attached to the same resource share. | May 6, 2022 |
| Lake Formation started tracking changes. | Lake Formation started tracking changes for its AWS managed policies. | May 6, 2022 |

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 [Change the default permission model \(p. 15\)](#).
2. Change the settings for existing Data Catalog resources. For instructions, see [Upgrading AWS Glue data permissions to the AWS Lake Formation model \(p. 23\)](#).

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

You can also change default security settings by using the Lake Formation [PutDataLakeSettings Action \(Python: put_data_lake_settings\) \(p. 338\)](#). This action takes as arguments an optional catalog ID and a [DataLakeSettings Structure \(p. 336\)](#).

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, RoleARN )
```

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 \(p. 320\)](#).

Note

`GetTableObjects`, `UpdateTableObjects`, and `GetWorkUnitResults` are high-volume data plane operations. Calls to these APIs are not currently logged to CloudTrail. For more information about data plane operations in CloudTrail, see [Logging data events for trails](#) in the *AWS CloudTrail User Guide*.

Changes in Lake Formation to support additional CloudTrail events will be documented at [Document history for AWS Lake Formation \(p. 386\)](#).

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.

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.

Integrating third-party services with Lake Formation

Integrating with AWS Lake Formation enables third-party services to securely access data in their Amazon S3 based data lakes. You can use Lake Formation as your authorization engine to manage or enforce permissions to your data lake with integrated AWS services such as Amazon Athena, Amazon EMR, and Redshift Spectrum. Lake Formation provides two options for integrating services:

1. The Lake Formation credential vending engine: Lake Formation can vend scoped-down temporary credentials in the form of AWS STS tokens to registered Amazon S3 locations based on the effective permissions, so that authorized engines can access data on behalf of users.
2. Central enforcement: Lake Formation [querying API](#) operations retrieve data from Amazon S3 and filter the results based on effective permissions. The engine or application integrating with the querying API operation can depend on Lake Formation to evaluate the calling identity's permissions and securely filter the data based on these permissions. Third-party query engines only see and operate on filtered data.

Topics

- [Using Lake Formation credential vending \(p. 304\)](#)

Using Lake Formation credential vending

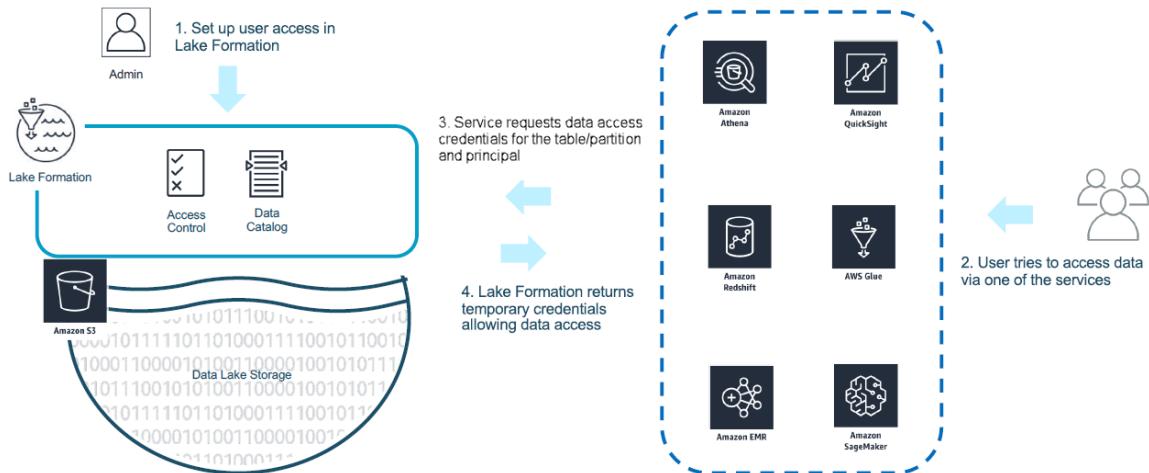
Lake Formation allows third-party services to integrate with Lake Formation by using [credential vending API](#) operations. This allows third-party services to use the same authorization and credential vending engine that the rest of AWS analytics services use. This section describes how to use credential vending API operations to integrate a third-party query engine with Lake Formation.

Topics

- [How Lake Formation credential vending works \(p. 304\)](#)
- [Roles and responsibilities in Lake Formation credential vending \(p. 306\)](#)
- [Lake Formation workflow for credential vending API operations \(p. 306\)](#)
- [Registering a third-party query engine \(p. 307\)](#)
- [Enabling permissions for a third-party query engine to call credential vending API operations \(p. 308\)](#)

How Lake Formation credential vending works

This section describes how to use credential vending API operations to integrate a third-party application (query engine) with Lake Formation.



1. The Lake Formation administrator performs the following activities:

- Registers an Amazon S3 location with Lake Formation by providing an IAM role (used for vending credentials) that has appropriate permissions to access data within the Amazon S3 location
- Registers a third-party application to be able to call Lake Formation's credential vending API operations. See [the section called "Registering a third-party query engine" \(p. 307\)](#)
- Grants permissions for users to access databases and tables

For example, if you want to publish a user sessions data set that includes some columns containing personally identifiable information (PII), to restrict access, you assign these columns an **LF-TBAC** tag named "classification" with a value of "sensitive". Next, you define a permission that allows a business analyst to access the user sessions data, but exclude those columns tagged with *classification = sensitive*.

- A principal (user) submits a query to an integrated service.
- The integrated application sends the request to Lake Formation asking for table information and credentials to access the table.
- If the querying principal is authorized to access the table, Lake Formation returns the credentials to the integrated application, which allows data access.
- The integrated service reads data from Amazon S3, filters columns based on the policies it received, and returns the results back to the principal.

Important

Lake Formation credential vending API operations enable a **distributed-enforcement with explicit deny on failure (fail-close) model**. This introduces a three-party security model between customers, third-party services and Lake Formation. Integrated services are trusted to properly enforce Lake Formation permissions (distributed-enforcement).

The integrated service is responsible for filtering the data read from Amazon S3 based on the policies returned from Lake Formation before the filtered data is returned back to the user. Integrated services follow a fail-close model, which means that they must fail the query if they are unable to enforce required Lake Formation permissions.

Roles and responsibilities in Lake Formation credential vending

| Role | Responsibility |
|--------------------|---|
| The customer | <ul style="list-style-type: none"> Enable Lake Formation credential vending (see the section called "Registering a third-party query engine" (p. 307)). Explicitly registers approved third parties with Lake Formation (see the section called "Registering a third-party query engine" (p. 307)). Tests and validates third-party solutions with Lake Formation permissions. Monitors and audits third-party usage of Lake Formation credential vending API operations. |
| The third-party | <ul style="list-style-type: none"> Publicly documents the supported capability for every software revision and provides instructions to enable it correctly. Accurately advertises the supported capabilities when calling Lake Formation credential vending API operations (according to the documentation). Securely stores and handles vended credentials to avoid credential leaks and privilege escalation. Enforces permissions based on supported capabilities and returns only filtered data to users Fails the query when unable to properly enforce required permissions |
| AWS Lake Formation | <ul style="list-style-type: none"> Correctly derives and returns effective permissions for a given principal. Validates third-party supported capabilities on an API operation call-by-call basis. Returns scoped-down IAM credentials only when the engine's advertised capabilities match those defined on the catalog resources, otherwise returns an error. |

Lake Formation workflow for credential vending API operations

The following is the work flow for credential vending API operations:

1. A user submits a query or request for data using an integrated third-party query engine. The query engine assumes an IAM role that represents the user or a group of users, and retrieves trusted credentials to be used when calling the credential vending API operations.
2. The query engine calls `GetUnfilteredTableMetadata`, and if it is a partitioned table, the query engine calls `GetUnfilteredPartitionsMetadata` to retrieve metadata and policy information from the Data Catalog.
3. Lake Formation performs authorization for the request. If the user doesn't have appropriate permissions on the table, then `AccessDeniedException` is thrown.
4. As part of the request, the query engine sends the filtering it supports. There are two flags that can be sent within an array: `COLUMN_PERMISSIONS` and `CELL_FILTER_PERMISSION`. If the query engine doesn't support any of these features, and a policy exists on the table for the feature, then a `PermissionTypeMismatchException` is thrown and the query fails. This is to avoid data leakage.
5. The returned response contains the following:

- The entire schema for the table so that query engines can use it to parse the data from storage.
 - A list of authorized columns that the user has access. If the authorized column list is empty, it indicates that the user has DESCRIBE permissions, but does not have SELECT permissions, and the query fails.
 - A flag, IsRegisteredWithLakeFormation, which indicates if Lake Formation can vend credentials to this resources data. If this returns false, then the customers' credentials should be used to access Amazon S3.
 - A list of CellFilters if any that should be applied to rows of data. This list contains columns and an expression to evaluate each row. This should only be populated if *CELL_FILTER_PERMISSION* is sent as part of the request and there is a data filter against the table for the calling user.
6. After the metadata is retrieved, the query engine calls `GetTemporaryGlueTableCredentials` or `GetTemporaryGluePartitionCredentials` to get AWS credentials to retrieve data from the Amazon S3 location.
 7. The query engine reads relevant objects from Amazon S3, filters the data based on the policies it received in step 2, and returns the results to the user.

The credential vending API operations for Lake Formation contain additional content for configuring integration with third-party query engines. You can see the operation details in the [Credential vending API operations section. \(p. 338\)](#)

Registering a third-party query engine

Before a third-party query engine can use credential vending API operations, you need to explicitly enable permissions for the query engine to call the API operations on your behalf. This is done in a few steps:

1. You need to specify the AWS accounts and IAM session tags that require permission to call the credential vending API operations through the AWS Lake Formation console, the AWS CLI or the API/SDK.
2. When the third-party query engine assumes the execution role in your account, the query engine must attach a session tag that is registered with Lake Formation representing the third-party engine. Lake Formation uses this tag to validate that if the request is coming from an approved engine. For more information about session tags, see [Session tags](#) in the IAM User Guide.
3. When setting up a third-party query engine execution role, you must have the following minimum set of permissions in the IAM policy:

```
{
  "Version": "2012-10-17",
  "Statement": {"Effect": "Allow",
    "Action": [
      "lakeformation:GetDataAccess",
      "glue:GetTable",
      "glue:GetTables",
      "glue:GetDatabase",
      "glue:GetDatabases",
      "glue>CreateDatabase",
      "glue:GetUserDefinedFunction",
      "glue GetUserDefinedFunctions",
      "glue:GetPartition",
      "glue:GetPartitions"
    ],
    "Resource": "*"
  }
}
```

- Set up a role trust policy on the query engine execution role to have fine access control on which session tag key value pair can be attached to this role. In the following example, this role is only allowed to have session tag key "LakeFormationAuthorizedCaller" and session tag value "engine1" to be attached, and no other session tag key value pair is allowed.

```
{  
    "Sid": "AllowPassSessionTags",  
    "Effect": "Allow",  
    "Principal": {  
        "AWS": "arn:aws:iam::111122223333:role/query-execution-role"  
    },  
    "Action": "sts:TagSession",  
    "Condition": {  
        "StringLike": {  
            "aws:RequestTag/LakeFormationAuthorizedCaller": "engine1"  
        }  
    }  
}
```

When `LakeFormationAuthorizedCaller` calls the `STS:AssumeRole` API operation to fetch credentials for the query engine to use, the session tag must be included in the [AssumeRole request](#). The returned temporary credential can be used to make Lake Formation credential vending API requests.

Lake Formation credential vending API operations require the calling principal to be an IAM role. The IAM role must include a session tag with a predetermined value that has been registered with Lake Formation. This tag allows Lake Formation to verify that the role used to call the credential vending API operations is allowed to do so.

Enabling permissions for a third-party query engine to call credential vending API operations

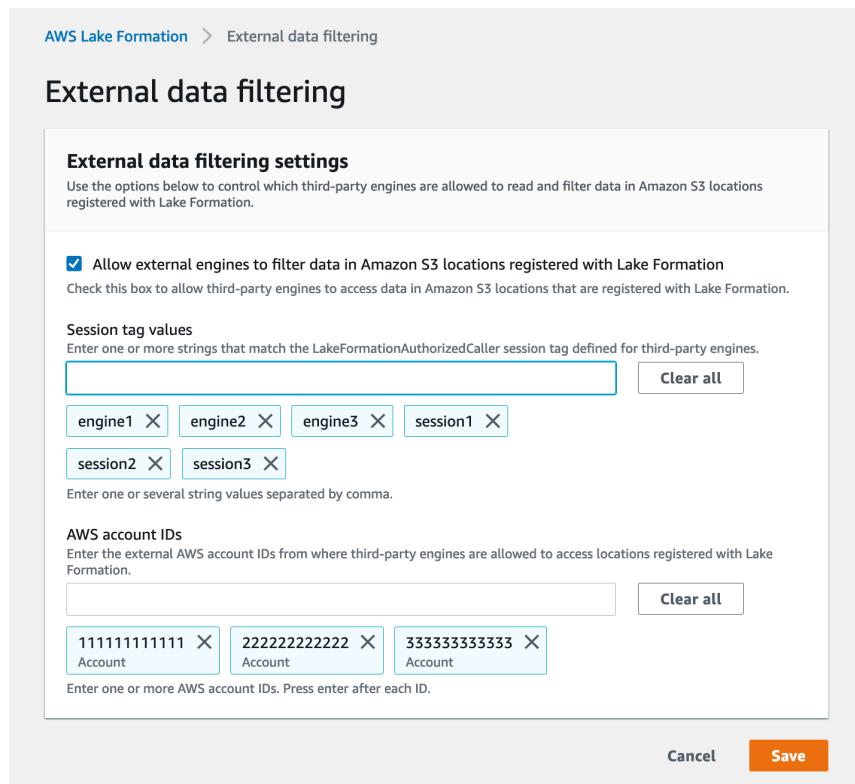
Follow these steps to allow a third-party query engine to call credential vending API operations through the AWS Lake Formation console, the AWS CLI or API/SDK.

Console

To register your account for external data filtering:

- Sign in to the AWS Management Console, and open the Lake Formation console at <https://console.aws.amazon.com/lakeformation/>.
- In the left-side navigation, expand **Permissions**, and then choose **External data filtering**.
- On the **External data filtering** page, choose the option **Allow external engines to filter data in Amazon S3 locations registered with Lake Formation**.
- Enter the session tags that you created for the third-party engine. For information about session tags, see [Passing session tags in AWS STS](#) in the *AWS Identity and Access Management User Guide*.
- Enter the account IDs for users that can use the third-party engine to access unfiltered metadata information and the data access credentials of resources in the current account.

You can also use the AWS account ID field for configuring cross-account access.



CLI

Use the `put-data-lake-settings` CLI command to set the following parameters.

There are three fields to configure when using this AWS CLI command:

- `allow-external-data-filtering` – (boolean) Indicates that a third-party engine can access unfiltered metadata information and data access credentials of resources in the current account.
- `external-data-filtering-allow-list` – (array) A list of account IDs that can access unfiltered metadata information and data access credentials of resources in the current account when using a third-party engine.
- `authorized-sessions-tag-value-list` – (array) A list of authorized session tag values (strings). If an IAM role credential has been attached with an authorized key-value pair, then if the session tag is included in the list, the session is granted access to unfiltered metadata information and data access credentials on resources in the configured account. The authorized session tag key is defined as `*LakeFormationAuthorizedCaller*`.

For example:

```
aws lakeformation put-data-lake-settings --cli-input-json file://datalakesettings.json

{
  "DataLakeSettings": {
    "DataLakeAdmins": [
      {
        "DataLakePrincipalIdentifier": "arn:aws:iam::111111111111:user/lakeAdmin"
      }
    ],
    "CreateDatabaseDefaultPermissions": []
  }
}
```

```
"CreateTableDefaultPermissions": [],
"TrustedResourceOwners": [],
"AllowExternalDataFiltering": true,
"ExternalDataFilteringAllowList": [
    {"DataLakePrincipalIdentifier": "111111111111"}
],
"AuthorizedSessionTagValueList": ["engine1"]
}
```

API/SDK

Use the PutDataLakeSetting API operation to set the following parameters.

There are three fields to configure when using this API operation:

- AllowExternalDataFiltering – (Boolean) Indicates whether a third-party engine can access unfiltered metadata information and data access credentials of resources in the current account.
- ExternalDataFilteringAllowList – (array) A list of account IDs that can access unfiltered metadata information and the data access credentials of resources in the current account using a third-party engine.
- AuthorizedSessionTagValueList – (array) A list of authorized tag values (strings). If an IAM role credential has been attached with an authorized tag, then the session is granted access to unfiltered metadata information and the data access credentials on resources in the configured account. The authorized session tag key is defined as *LakeFormationAuthorizedCaller*.

For example:

```
//Enable session tag on existing data lake settings
public void sessionTagSetUpForExternalFiltering(AWSLakeFormationClient lakeformation) {
    GetDataLakeSettingsResult getDataLakeSettingsResult =
        lfClient.getDataLakeSettings(new GetDataLakeSettingsRequest());
    DataLakeSettings dataLakeSettings =
        getDataLakeSettingsResult.getDataLakeSettings();

    //set account level flag to allow external filtering
    dataLakeSettings.setAllowExternalDataFiltering(true);

    //set account that are allowed to call credential vending or Glue
    GetFilteredMetadata API
    List<DataLakePrincipal> allowlist = new ArrayList<>();
    allowlist.add(new
    DataLakePrincipal().withDataLakePrincipalIdentifier("111111111111"));
    dataLakeSettings.setWhitelistedForExternalDataFiltering(allowlist);

    //set registered session tag values
    List<String> registeredTagValues = new ArrayList<>();
    registeredTagValues.add("engine1");
    dataLakeSettings.setAuthorizedSessionTagValueList(registeredTagValues);

    lakeformation.putDataLakeSettings(new
    PutDataLakeSettingsRequest().withDataLakeSettings(dataLakeSettings));
}
```

Working with other AWS services

AWS services such as Amazon Athena, AWS Glue, Amazon Redshift Spectrum, and Amazon EMR can use Lake Formation to securely access data in Amazon S3 locations registered with Lake Formation. With Lake Formation, you can define and manage fine-grained access control (FGAC) permissions for your data in the AWS Glue Data Catalog. Each of these AWS services is a trusted caller to Lake Formation, and Lake Formation provides access to data stored in Amazon S3 through temporary credentials. For more information, see [How Lake Formation credential vending works \(p. 304\)](#).

To avail these capabilities, Lake Formation requires you to first register the Amazon S3 location, and assign appropriate permissions to the IAM principal for accessing the table, the database, and the Amazon S3 location. For more information see, [Managing Lake Formation permissions \(p. 171\)](#).

Topics

- [Using AWS Lake Formation with Amazon Athena \(p. 311\)](#)
- [Using AWS Lake Formation with Amazon Redshift Spectrum \(p. 313\)](#)
- [Using AWS Lake Formation with AWS Glue \(p. 315\)](#)
- [Using AWS Lake Formation with Amazon EMR \(p. 317\)](#)
- [Using AWS Lake Formation with Amazon QuickSight \(p. 318\)](#)

Using AWS Lake Formation with Amazon Athena

[Amazon Athena](#) is a server-less query service that helps you analyze structured, semi-structured, and unstructured data stored in Amazon S3. Athena supports querying data from CSV, JSON, Parquet, and Avro data formats. Athena also supports table formats like [Apache Hive](#), [Apache Hudi](#), [Apache Iceberg](#) and [Lake Formation governed tables](#). Athena integrates with the AWS Glue Data Catalog to store metadata of your data sets in Amazon S3. Athena can use Lake Formation to define and maintain access control policies on those data sets.

Here are some common use cases where you can use Lake Formation with Athena.

- Grant IAM users Lake Formation permissions for accessing the Data Catalog resources (database and tables) from Athena. You can use either the named resource method or LF-tags to define permissions on database and tables. For more information, see:
 - [Granting database permissions using the Lake Formation console and the named resource method \(p. 181\)](#)
 - [Lake Formation Tag-based access control \(p. 231\)](#)

Lake Formation permissions support both read and write operations on databases and tables.

Note

You cannot apply data filters when you use LF-tags to manage permissions on Data Catalog resources.

- Control the query results by using [Data filters in Lake Formation \(p. 261\)](#) to secure tables in your Amazon S3 data lakes by granting permissions at column, row, and cell-levels. See the [limitation on partition projection in Athena user guide](#).
- Enforce fine-grained access control on the data available to the SAML-based Athena user when running federated queries.

Athena JDBC and ODBC drivers support configuring federated access to your data source using SAML-based Identity Provider (IdP). Use Amazon QuickSight integrated with Lake Formation with your existing IAM role or SAML users or groups to visualize Athena query results.

Note

Lake Formation permissions for SAML users and groups will apply only when you submit queries to Athena using the JDBC or ODBC driver.

For more information, see [Using Lake Formation and the Athena JDBC and ODBC drivers for federated access to Athena](#).

- Use [Cross-account data sharing in Lake Formation \(p. 199\)](#) to query tables in another account.

Note

For more information on limitations when using Lake Formation permissions to Views, see [Considerations and Limitations](#).

Support for transactional table formats

Applying Lake Formation permissions allows you to secure your transactional data in your Amazon S3 based data lakes. The table below lists transactional table formats supported in Athena and the Lake Formation permissions. Lake Formation enforces these permissions when Athena users run their queries.

| Table format | Description and allowed operations | Lake Formation permissions supported in Athena |
|--------------------------------|--|---|
| Lake Formation governed tables | <p>Governed tables support ACID (atomic, consistent, isolated, and durable) transactions, automatic data compaction, and time-travel queries. Athena supports only create and read operations on governed tables.</p> <p>Athena does not support delete, insert into, and update operations.</p> <p>For more information, see Governed tables in Lake Formation (p. 120).</p> <p>For step-by-step tutorial to learn how to create a governed table, see Creating a governed table in Lake Formation (p. 48).</p> | Use Data filtering and cell-level security in Lake Formation (p. 260) to secure governed tables using table, column, row, and cell-level permissions. |
| Apache Hudi | <p>A format used to simplify incremental data processing and data pipeline development.</p> <p>Athena supports create and read operations using Apache Hudi table formats on Amazon S3 data sets for both Copy on Write (CoW) and Merge On Read (MoR) Hudi table types.</p> | Table, column, row, and cell-level permissions are supported. |

| Table format | Description and allowed operations | Lake Formation permissions supported in Athena |
|-----------------------------|---|---|
| | <p>Athena does not support write operations on Hudi tables.</p> <p>Use Athena to query Hudi datasets.</p> | |
| Apache Iceberg | <p>An open table format that manages large collections of files as tables, and supports modern analytic data lake operations such as record-level insert, update, delete, and time travel queries.</p> <p>For more information on Athena's support for Iceberg tables, see Using Iceberg tables.</p> | Table, column, row, and cell-level permissions are supported. |
| Linux Foundation Delta Lake | <p>Delta Lake is an open-source project that helps to implement modern data lake architectures commonly built on Amazon S3 or Hadoop Distributed File System (HDFS).</p> <p>Amazon Athena does not support Delta lake tables. You can read Delta Lake tables using symlink manifest tables. For more information, see Crawl Delta Lake tables using AWS Glue crawlers</p> | Lake Formation integration is not available for Delta Lake table queries. |

Additional resources

Blog posts, videos, and workshops

- [Query an Apache Hudi dataset in an Amazon S3 data lake with Amazon Athena](#)
- [Build an Apache Iceberg data lake using Amazon Athena, Amazon EMR, and AWS Glue](#)
- [Insert, update, delete on Amazon S3 with Athena and Apache Iceberg](#)
- [LF-Tag based access control](#) Lake Formation workshop on querying a data lake.

Using AWS Lake Formation with Amazon Redshift Spectrum

[Amazon Redshift Spectrum](#) lets you to query and retrieve data in Amazon S3 data lakes without loading data into Amazon Redshift cluster nodes.

Redshift Spectrum supports two ways of registering an external AWS Glue data catalog enabled with Lake Formation.

- Using a cluster attached IAM role that has permission to the Data Catalog

To create an IAM role, follow the steps outlined in the below procedure.

[To create an IAM role for Amazon Redshift using an AWS Glue Data Catalog enabled for AWS Lake Formation](#)

- Using a federated IAM identity configured to manage access to external AWS Glue Data Catalog resources

Redshift Spectrum supports querying Lake Formation tables using federated IAM identities. The IAM identities can be an IAM user or an IAM role. For more information on IAM identity federation in Redshift Spectrum, see [Using a federated identity to manage Amazon Redshift access to local resources and Redshift Spectrum external tables](#).

With Lake Formation integration with Redshift Spectrum, you can define row, column, and cell-level access control permissions on tables after your data is registered with Lake Formation.

For more information see [Using Redshift Spectrum with AWS Lake Formation](#).

Redshift Spectrum supports reads or SELECT queries on the Lake Formation managed external schema tables. Inserts are only supported from an Amazon Redshift local table to an external schema table that is managed by Lake Formation.

For more information, see [Creating external schemas for Redshift Spectrum](#).

Support for transactional table types

This table lists transactional table formats supported in Redshift Spectrum and the applicable Lake Formation permissions.

Supported table formats

| Table format | Description and allowed operations | Lake Formation permissions supported in Redshift Spectrum |
|--------------------------------|--|--|
| Lake Formation governed tables | <p>Governed tables support ACID (atomic, consistent, isolated, and durable) transactions, automatic data compaction, and time-travel queries. Redshift Spectrum supports only create and read operations on governed tables.</p> <p>Redshift Spectrum does not support delete, insert into, and update operations.</p> <p>For more information, see Governed tables in Lake Formation (p. 120).</p> <p>For step-by-step tutorial to learn how to create a governed table, see Creating a governed table in Lake Formation (p. 48).</p> | <p>Use Data filtering and cell-level security in Lake Formation (p. 260) to secure governed tables using table, column, row, and cell-level permissions.</p> |

| Table format | Description and allowed operations | Lake Formation permissions supported in Redshift Spectrum |
|-----------------------------|---|--|
| Apache Hudi | <p>A format used to simplify incremental data processing and data pipeline development.</p> <p>Redshift Spectrum supports insert, delete, and upsert write operations using Apache Hudi Copy on Write (CoW) table format on Amazon S3.</p> <p>For more information, see Creating external tables for data managed in Apache Hudi.</p> | Table, column, row, and cell-level permissions are supported. |
| Apache Iceberg | An open table format that manages large collections of files as tables and supports modern analytic data lake operations such as record-level insert, update, delete, and time travel queries. | Redshift Spectrum does not support Apache Iceberg tables for querying. |
| Linux Foundation Delta Lake | <p>Delta Lake is an open-source project that helps implement modern data lake architectures commonly built on Amazon S3 or Hadoop Distributed File System (HDFS).</p> <p>Redshift Spectrum supports querying Delta Lake tables. For more information, see Creating external tables for data managed in Delta Lake.</p> | Table, column, row, and cell-level permissions are supported. |

Additional resources

Blog posts and workshops

- [Centralize governance for your data lake using AWS Lake Formation while enabling a modern data architecture with Amazon Redshift Spectrum](#)
- [Use Redshift Spectrum to query Apache Hudi Copy On Write \(CoW\) tables in Amazon S3 data lake](#)

Using AWS Lake Formation with AWS Glue

Data engineers and DevOps professionals use AWS Glue with Extract, Transform and Load (ETL) with Apache Spark to perform transformations on their data sets in Amazon S3 and load the transformed data into data lakes and data warehouses for analytics, machine learning, and application development. With different teams accessing the same data set in Amazon S3, it is imperative to grant and restrict permissions based on their roles.

AWS Lake Formation is built on AWS Glue, and the services interact in the following ways:

- Lake Formation and AWS Glue share the same Data Catalog.
- The following Lake Formation console features invoke the AWS Glue console:
 - Jobs – For more information, see [Adding Jobs](#) in the *AWS Glue Developer Guide*.
 - Crawlers – For more information, see [Cataloging Tables with a Crawler](#) in the *AWS Glue Developer Guide*.
- The workflows generated when you use a Lake Formation blueprint are AWS Glue workflows. You can view and manage these workflows in both the Lake Formation console and the AWS Glue console.
- Machine learning transforms are provided with Lake Formation and are built on AWS Glue API operations. You create and manage machine learning transforms on the AWS Glue console. For more information, see [Machine Learning Transforms](#) in the *AWS Glue Developer Guide*.

You can use the Lake Formation fine-grained access control to manage your existing Data Catalog resources and Amazon S3 data locations.

Note

AWS Glue ETL requires full access to the entire table while fetching data from underlying Amazon S3 location. AWS Glue ETL job fails if you apply column-level permissions on a table. However, you can create column-level and row-level security by defining data filters. For more information, see [Notes and restrictions for column-level filtering \(p. 265\)](#) Lake Formation evaluates the data filter defined on the table and retrieves only the filtered data from Amazon S3 required for the AWS Glue ETL job.

Support for transactional table types

This table lists transactional table formats supported in AWS Glue and the applicable Lake Formation permissions.

Supported table formats

| Table format | Description and allowed operations | Lake Formation permissions supported in AWS Glue |
|--------------------------------|--|---|
| Lake Formation governed tables | <p>Governed tables support ACID (atomic, consistent, isolated, and durable) transactions. AWS Glue supports transactions and time travel queries on governed tables.</p> <p>To use transactions in your AWS Glue ETL jobs, you begin a transaction before performing a read or write, and commit the transaction upon completion.</p> <p>AWS Glue ETL jobs can include a timestamp to indicate that you want to discover the state of the data at a particular date and time.</p> <p>For more information, see Governed tables in Lake Formation (p. 120).</p> | Use Data filtering and cell-level security in Lake Formation (p. 260) to secure governed tables using table, column, row, and cell-level permissions. |

| Table format | Description and allowed operations | Lake Formation permissions supported in AWS Glue |
|-----------------------------|---|--|
| | For step-by-step tutorial to learn how to create a governed table, see Creating a governed table in Lake Formation (p. 48) . | |
| Apache Hudi | <p>A open table format used to simplify incremental data processing and data pipeline development.</p> <p>Use AWS Marketplace Connector to work with Hudi tables.</p> | Lake Formation integration is not available for Hudi tables. |
| Apache Iceberg | <p>An open table format that manages large collections of files as tables.</p> <p>Use AWS Marketplace Connector to work with Iceberg tables.</p> | Lake Formation integration is not available for Iceberg tables. |
| Linux Foundation Delta Lake | <p>Delta Lake is an open-source project that helps implement modern data lake architectures commonly built on Amazon S3 or Hadoop Distributed File System (HDFS).</p> <p>AWS Glue supports reading and writing to Delta Lake tables using the marketplace connector.</p> <p>Use AWS Marketplace Connector to work with Delta Lake tables.</p> | Lake Formation integration is not available for Delta Lake tables. |

Additional resources

Blog posts and repositories

- [Use the AWS Glue connector to read and write Apache Iceberg tables with ACID transactions and perform time travel](#)
- [Writing to Apache Hudi tables using AWS Glue custom connector](#)
- AWS repository of [Cloudformation template and pyspark code sample to analyze streaming data using AWS Glue, Apache Hudi, and Amazon S3](#).

Using AWS Lake Formation with Amazon EMR

Amazon EMR is a flexible AWS managed cluster platform on which you can run any custom code on supported big data frameworks like Hadoop Map-Reduce, Spark, Hive, Presto, etc. Organizations also use Amazon EMR to run both batch and stream data processing applications across a highly distributed cluster. Using Amazon EMR, you can run your data transformations and custom code on database and tables whose permissions are managed by Lake Formation.

There are three options for deploying Amazon EMR:

- EMR on EC2
- EMR Serverless
- EMR on EKS

Lake Formation integration is available only with Amazon EMR on Amazon EC2. Lake Formation integrates with Amazon EMR on Amazon EC2 in two ways.

- Enable SAML-based authentication using your organization's Identity Provider (IdP) for Amazon EMR Notebooks or Zeppelin to run Spark jobs on Lake Formation managed databases, tables and columns. In this method, only Spark scripts are supported while Hive, Presto, etc are not supported.
- Use Lake Formation to manage access to your data lakes based on your [Amazon EMR runtime role](#) allowing users to submit Spark and Hive jobs to Amazon EMR clusters through Step APIs. The types of operations supported by this method are listed under [Additional considerations](#).

Note

Both methods do not support using data filters.

For more information, see [Integrate Amazon EMR with Lake Formation](#).

Support for transactional table formats

Amazon EMR provides support for [Apache Hudi](#), [Apache Iceberg](#) and [Delta Lake](#) table formats. However, Lake Formation permissions are currently not supported on Apache Hudi, Apache Iceberg and Linux Foundation Delta Lake tables.

Additional resources

User guide, blog posts, and workshops

- [Integration with Amazon EMR using Runtime Roles](#)
- [Get a quick start with Apache Hudi, Apache Iceberg, and Delta Lake with Amazon EMR on EKS](#)
- [Using Delta Lake OSS with Amazon EMR Serverless](#)

Using AWS Lake Formation with Amazon QuickSight

Amazon QuickSight supports exploring datasets managed by Lake Formation permissions in Amazon S3 using Athena.

Both Standard and Enterprise edition users of Amazon QuickSight integrate with Lake Formation, but slightly differently.

- Enterprise edition – Grant fine-grained access control (FGAC) permissions to individual Amazon QuickSight users, groups, and IAM roles to access databases and tables.
- Standard edition – Grant permissions to IAM roles to access databases and tables.

Note

By default, Amazon QuickSight uses a role named `aws-quicksight-service-role-v0`. You can also define custom roles with required permissions that enable Amazon QuickSight to access Athena.

For more information, see [Authorizing connections through AWS Lake Formation](#)

Additional resources

Blog posts

- [Enable fine-grained permissions for Amazon QuickSight authors in AWS Lake Formation](#)
- [Securely analyze your data with AWS Lake Formation and Amazon QuickSight](#)

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": "AROAQGFTBBBG0BWV2EMZA: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 \(p. 213\)](#)

AWS Lake Formation API

Note

Updated [API Reference](#) for the AWS Lake Formation service is now available.
The page you are viewing will be removed soon.

Contents

- [Permissions APIs \(p. 325\)](#)
 - [Data Types \(p. 325\)](#)
 - [Resource Structure \(p. 326\)](#)
 - [DatabaseResource Structure \(p. 326\)](#)
 - [TableResource Structure \(p. 327\)](#)
 - [TableWithColumnsResource Structure \(p. 327\)](#)
 - [DataCellsFilterResource Structure \(p. 328\)](#)
 - [DataLocationResource Structure \(p. 328\)](#)
 - [DataLakePrincipal Structure \(p. 328\)](#)
 - [ResourcePermissions Structure \(p. 329\)](#)
 - [ResourcePermissionsError Structure \(p. 329\)](#)
 - [PrincipalResourcePermissions Structure \(p. 329\)](#)
 - [DetailsMap Structure \(p. 330\)](#)
 - [PrincipalResourcePermissionsError Structure \(p. 330\)](#)
 - [ColumnWildcard Structure \(p. 330\)](#)
 - [BatchPermissionsRequestEntry Structure \(p. 330\)](#)
 - [BatchPermissionsFailureEntry Structure \(p. 331\)](#)
 - [PrincipalPermissions Structure \(p. 331\)](#)
 - [Operations \(p. 331\)](#)
 - [GrantPermissions Action \(Python: grant_permissions\) \(p. 331\)](#)
 - [RevokePermissions Action \(Python: revoke_permissions\) \(p. 332\)](#)
 - [BatchGrantPermissions Action \(Python: batch_grant_permissions\) \(p. 333\)](#)
 - [BatchRevokePermissions Action \(Python: batch_revoke_permissions\) \(p. 334\)](#)
 - [GetEffectivePermissionsForPath Action \(Python: get_effective_permissions_for_path\) \(p. 334\)](#)
 - [ListPermissions Action \(Python: list_permissions\) \(p. 335\)](#)
- [Data Lake Settings APIs \(p. 336\)](#)
 - [Data Types \(p. 336\)](#)
 - [DataLakeSettings Structure \(p. 336\)](#)
 - [Operations \(p. 337\)](#)
 - [GetDataLakeSettings Action \(Python: get_data_lake_settings\) \(p. 337\)](#)
 - [PutDataLakeSettings Action \(Python: put_data_lake_settings\) \(p. 338\)](#)
- [Credential Vending API \(p. 338\)](#)
 - [Data Types \(p. 338\)](#)
 - [FilterCondition Structure \(p. 338\)](#)
 - [ColumnNames list \(p. 339\)](#)
 - [ResourceInfo Structure \(p. 339\)](#)
 - [Operations \(p. 339\)](#)

- [RegisterResource Action \(Python: register_resource\) \(p. 339\)](#)
- [DeregisterResource Action \(Python: deregister_resource\) \(p. 340\)](#)
- [ListResources Action \(Python: list_resources\) \(p. 341\)](#)
- [Tagging API \(p. 341\)](#)
 - [Data Types \(p. 341\)](#)
 - [Tag Structure \(p. 342\)](#)
 - [LFTagKeyResource Structure \(p. 342\)](#)
 - [LFTagPolicyResource Structure \(p. 342\)](#)
 - [TaggedTable Structure \(p. 343\)](#)
 - [TaggedDatabase Structure \(p. 343\)](#)
 - [LFTag Structure \(p. 343\)](#)
 - [LFTagPair Structure \(p. 344\)](#)
 - [LFTagError Structure \(p. 344\)](#)
 - [ColumnLFTag Structure \(p. 344\)](#)
 - [Operations \(p. 344\)](#)
 - [AddLFTagsToResource Action \(Python: add_lf_tags_to_resource\) \(p. 345\)](#)
 - [RemoveLFTagsFromResource Action \(Python: remove_lf_tags_from_resource\) \(p. 345\)](#)
 - [GetResourceLFTags Action \(Python: get_resource_lf_tags\) \(p. 346\)](#)
 - [ListLFTags Action \(Python: list_lf_tags\) \(p. 347\)](#)
 - [CreateLFTag Action \(Python: create_lf_tag\) \(p. 348\)](#)
 - [GetLFTag Action \(Python: get_lf_tag\) \(p. 348\)](#)
 - [UpdateLFTag Action \(Python: update_lf_tag\) \(p. 349\)](#)
 - [DeleteLFTag Action \(Python: delete_lf_tag\) \(p. 350\)](#)
 - [SearchTablesByLFTags Action \(Python: search_tables_by_lf_tags\) \(p. 351\)](#)
 - [SearchDatabasesByLFTags Action \(Python: search_databases_by_lf_tags\) \(p. 352\)](#)
- [Transaction APIs \(p. 352\)](#)
 - [Data Types \(p. 353\)](#)
 - [TransactionDescription Structure \(p. 353\)](#)
 - [VirtualObject Structure \(p. 353\)](#)
 - [Operations \(p. 353\)](#)
 - [StartTransaction Action \(Python: start_transaction\) \(p. 354\)](#)
 - [CommitTransaction Action \(Python: commit_transaction\) \(p. 354\)](#)
 - [CancelTransaction Action \(Python: cancel_transaction\) \(p. 355\)](#)
 - [ExtendTransaction Action \(Python: extend_transaction\) \(p. 355\)](#)
 - [DescribeTransaction Action \(Python: describe_transaction\) \(p. 356\)](#)
 - [ListTransactions Action \(Python: list_transactions\) \(p. 356\)](#)
 - [DeleteObjectsOnCancel Action \(Python: delete_objects_on_cancel\) \(p. 357\)](#)
 - [Exceptions \(p. 358\)](#)
 - [TransactionCommitInProgressException Structure \(p. 358\)](#)
 - [TransactionAbortedException Structure \(p. 358\)](#)
 - [TransactionCommittedException Structure \(p. 359\)](#)
 - [TransactionCanceledException Structure \(p. 359\)](#)
 - [TransactionContentionException Structure \(p. 359\)](#)
 - [ResourceNotReadyException Structure \(p. 359\)](#)
- [Object APIs \(p. 359\)](#)

- [Data Types \(p. 360\)](#)
- [TableObject Structure \(p. 360\)](#)
- [PartitionObjects Structure \(p. 360\)](#)
- [AddObjectInput Structure \(p. 360\)](#)
- [DeleteObjectInput Structure \(p. 361\)](#)
- [WriteOperation Structure \(p. 361\)](#)
- [Operations \(p. 361\)](#)
- [GetTableObjects Action \(Python: get_table_objects\) \(p. 362\)](#)
- [UpdateTableObjects Action \(Python: update_table_objects\) \(p. 363\)](#)
- [Data Filter APIs \(p. 364\)](#)
 - [Data Types \(p. 364\)](#)
 - [DataCellsFilter Structure \(p. 364\)](#)
 - [RowFilter Structure \(p. 364\)](#)
 - [Operations \(p. 365\)](#)
 - [CreateDataCellsFilter Action \(Python: create_data_cells_filter\) \(p. 365\)](#)
 - [DeleteDataCellsFilter Action \(Python: delete_data_cells_filter\) \(p. 365\)](#)
 - [ListDataCellsFilter Action \(Python: list_data_cells_filter\) \(p. 366\)](#)
- [Storage APIs \(p. 367\)](#)
 - [Data Types \(p. 367\)](#)
 - [StorageOptimizer Structure \(p. 367\)](#)
 - [Operations \(p. 368\)](#)
 - [ListTableStorageOptimizers Action \(Python: list_table_storage_optimizers\) \(p. 368\)](#)
 - [UpdateTableStorageOptimizer Action \(Python: update_table_storage_optimizer\) \(p. 369\)](#)
- [Common Data Types \(p. 370\)](#)
 - [ErrorDetail Structure \(p. 370\)](#)
 - [String Patterns \(p. 370\)](#)

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 \(p. 326\)](#)
- [DatabaseResource Structure \(p. 326\)](#)
- [TableResource Structure \(p. 327\)](#)
- [TableWithColumnsResource Structure \(p. 327\)](#)
- [DataCellsFilterResource Structure \(p. 328\)](#)
- [DataLocationResource Structure \(p. 328\)](#)
- [DataLakePrincipal Structure \(p. 328\)](#)
- [ResourcePermissions Structure \(p. 329\)](#)
- [ResourcePermissionsError Structure \(p. 329\)](#)
- [PrincipalResourcePermissions Structure \(p. 329\)](#)
- [DetailsMap Structure \(p. 330\)](#)
- [PrincipalResourcePermissionsError Structure \(p. 330\)](#)

- [ColumnWildcard Structure \(p. 330\)](#)
- [BatchPermissionsRequestEntry Structure \(p. 330\)](#)
- [BatchPermissionsFailureEntry Structure \(p. 331\)](#)
- [PrincipalPermissions Structure \(p. 331\)](#)

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 \(p. 326\)](#) 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 \(p. 327\)](#) 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 \(p. 327\)](#) 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 \(p. 328\)](#) object.

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

- DataCellsFilter – A [DataCellsFilterResource \(p. 328\)](#) object.

A data cell filter.

- LFTag – A [LFTagKeyResource \(p. 342\)](#) object.

The LF-tag key and values attached to a resource.

- LFTagPolicy – A [LFTagPolicyResource \(p. 342\)](#) object.

A list of LF-tag conditions that define a resource's LF-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 \(p. 370\)](#).

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 \(p. 370\)](#).

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 \(p. 370\)](#).

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 \(p. 370\)](#).

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 \(p. 370\)](#).

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 \(p. 370\)](#).

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 \(p. 370\)](#).

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 \(p. 370\)](#).

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 \(p. 330\)](#) object.
A wildcard specified by a ColumnWildcard object. At least one of ColumnNames or ColumnWildcard is required.

DataCellsFilterResource Structure

A structure for a data cells filter resource.

Fields

- **TableCatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).
The ID of the catalog to which the table belongs.
- **DatabaseName** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).
A database in the AWS Glue Data Catalog.
- **TableName** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).
The name of the table.
- **Name** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).
The name of the data cells filter.

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 \(p. 370\)](#).
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 \(p. 326\)](#) 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 \(p. 329\)](#) object.
A list of resource permissions that had errors.
- **Error** – An [ErrorDetail \(p. 370\)](#) 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 \(p. 328\)](#) object.
The Data Lake principal to be granted or revoked permissions.
- **Resource** – A [Resource \(p. 326\)](#) 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 \(p. 330\)](#) object.
This attribute can be used to return any additional details of PrincipalResourcePermissions. Currently returns only as a AWS 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 AWS RAM resource share ARN.

Fields

- ResourceShare – An array of UTF-8 strings.

A resource share ARN for a catalog resource shared through AWS RAM.

PrincipalResourcePermissionsError Structure

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

Fields

- PrincipalResourcePermissions – A [PrincipalResourcePermissions \(p. 329\)](#) object.
The principal permissions that were to be granted or revoked.
- Error – An [ErrorDetail \(p. 370\)](#) 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 \(p. 328\)](#) object.
The principal to be granted a permission.
- Resource – A [Resource \(p. 326\)](#) 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. 330) object.
 - An identifier for an entry of the batch request.
- **Error** – An [ErrorDetail](#) (p. 370) object.
 - An error message that applies to the failure of the entry.

PrincipalPermissions Structure

Permissions granted to a principal.

Fields

- **Principal** – A [DataLakePrincipal](#) (p. 328) 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. 331)
- [RevokePermissions Action \(Python: revoke_permissions\)](#) (p. 332)
- [BatchGrantPermissions Action \(Python: batch_grant_permissions\)](#) (p. 333)
- [BatchRevokePermissions Action \(Python: batch_revoke_permissions\)](#) (p. 334)
- [GetEffectivePermissionsForPath Action \(Python: get_effective_permissions_for_path\)](#) (p. 334)
- [ListPermissions Action \(Python: list_permissions\)](#) (p. 335)

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. 370).

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 \(p. 328\)](#) 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 \(p. 326\)](#) 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. Lake Formation requires that each principal be authorized to perform a specific task on 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`
- `InvalidArgumentException`

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 \(p. 370\)](#).

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 \(p. 328\)](#) object.

The principal to be revoked permissions on the resource.

- **Resource** – *Required:* A [Resource \(p. 326\)](#) 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](#)
- [InvalidArgumentException](#)

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 \(p. 370\)](#).

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 \(p. 330\)](#) 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 \(p. 331\)](#) objects.

A list of failures to grant permissions to the resources.

Errors

- [InvalidArgumentException](#)
- [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 \(p. 370\)](#).

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 \(p. 330\)](#) 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 \(p. 331\)](#) 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 \(p. 370\)](#).

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 \(p. 329\)](#) 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

- [InvalidArgumentException](#)
- [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 \(p. 370\)](#).

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 \(p. 328\)](#) 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 \(p. 326\)](#) 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.

- **IncludeRelated** – UTF-8 string, not less than 1 or more than 5 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

Indicates that related permissions should be included in the results.

Response

- **PrincipalResourcePermissions** – An array of [PrincipalResourcePermissions \(p. 329\)](#) 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

- [InvalidArgumentException](#)
- [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 \(p. 336\)](#)

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 \(p. 328\)](#) objects.
A list of AWS Lake Formation principals. Supported principals are IAM users or IAM roles.
- **CreateDatabaseDefaultPermissions** – An array of [PrincipalPermissions \(p. 331\)](#) 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.

For more information, see [Changing the Default Security Settings for Your Data Lake](#).

- `CreateTableDefaultPermissions` – An array of [PrincipalPermissions \(p. 331\)](#) 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.

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 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 \(Python: get_data_lake_settings\) \(p. 337\)](#)
- [PutDataLakeSettings Action \(Python: put_data_lake_settings\) \(p. 338\)](#)

GetDataLakeSettings Action (Python: `get_data_lake_settings`)

Retrieves the list of the data lake administrators of a AWS 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 \(p. 370\)](#).

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 Lake Formation environment.

Response

- `DataLakeSettings` – A [DataLakeSettings \(p. 336\)](#) object.

A structure representing a list of 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 AWS 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 \(p. 370\)](#).

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 Lake Formation environment.

- DataLakeSettings – *Required*: A [DataLakeSettings \(p. 336\)](#) object.

A structure representing a list of 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. 338)
- ColumnNames list (p. 339)
- ResourceInfo Structure (p. 339)

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 \(p. 370\)](#).

The IAM role that registered a resource.

- **LastModified** – Timestamp.

The date and time the resource was last modified.

Operations

- [RegisterResource Action \(Python: register_resource\) \(p. 339\)](#)
- [DeregisterResource Action \(Python: deregister_resource\) \(p. 340\)](#)
- [ListResources Action \(Python: list_resources\) \(p. 341\)](#)

RegisterResource Action (Python: register_resource)

Registers the resource as managed by the Data Catalog.

To add or update data, AWS 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 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 \(p. 370\)](#).

The identifier for the role that registers the resource.

Response

- *No Response parameters.*

Errors

- `InvalidArgumentException`
- `InternalServiceException`
- `OperationTimeoutException`
- `AlreadyExistsException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `AccessDeniedException`

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

- [InvalidArgumentException](#)
- [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 \(p. 338\)](#) 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 \(p. 339\)](#) 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

- [InvalidArgumentException](#)
- [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

- [Tag Structure \(p. 342\)](#)
- [LFTagKeyResource Structure \(p. 342\)](#)
- [LFTagPolicyResource Structure \(p. 342\)](#)
- [TaggedTable Structure \(p. 343\)](#)

- [TaggedDatabase Structure \(p. 343\)](#)
- [LFTag Structure \(p. 343\)](#)
- [LFTagPair Structure \(p. 344\)](#)
- [LFTagError Structure \(p. 344\)](#)
- [ColumnLFTag Structure \(p. 344\)](#)

Tag Structure

A structure for a key-value pair LF-tag.

Fields

- **key** – UTF-8 string, not less than 1 or more than 128 bytes long.
The key for the LF-tag.
- **value** – UTF-8 string, not more than 256 bytes long.
The value of the LF-tag.

LFTagKeyResource Structure

A structure containing an LF-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 \(p. 370\)](#).
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 \(p. 370\)](#).
The key-name for the LF-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 LF-tag conditions that apply to a resource's LF-tag policy.

Fields

- **CatalogId** – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).
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 LF-tag policy applies.

- **Expression – Required:** An array of [LFTag \(p. 343\)](#) objects, not less than 1 or more than 5 structures.

A list of LF-tag conditions that apply to the resource's LF-tag policy.

TaggedTable Structure

A structure describing a table resource with LF-tags.

Fields

- **Table – A [TableResource \(p. 327\)](#) object.**

A table that has LF-tags attached to it.

- **LFTagOnDatabase – An array of [LFTagPair \(p. 344\)](#) objects, not less than 1 or more than 50 structures.**

A list of LF-tags attached to the database where the table resides.

- **LFTagsOnTable – An array of [LFTagPair \(p. 344\)](#) objects, not less than 1 or more than 50 structures.**

A list of LF-tags attached to the table.

- **LFTagsOnColumns – An array of [ColumnLFTag \(p. 344\)](#) objects.**

A list of LF-tags attached to columns in the table.

TaggedDatabase Structure

A structure describing a database resource with LF-tags.

Fields

- **Database – A [DatabaseResource \(p. 326\)](#) object.**

A database that has LF-tags attached to it.

- **LFTags – An array of [LFTagPair \(p. 344\)](#) objects, not less than 1 or more than 50 structures.**

A list of LF-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 that do not have the LF-tag 'PII' in tables that have the LF-tag 'Prod'.

Fields

- **TagKey – Required:** UTF-8 string, not less than 1 or more than 128 bytes long, matching the [Custom string pattern #9 \(p. 370\)](#).

The key-name for the LF-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 an LF-tag key-value pair.

Fields

- CatalogId – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

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, matching the [Custom string pattern #9 \(p. 370\)](#).

The key-name for the LF-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.

LFTagError Structure

A structure containing an error related to a TagResource or UnTagResource operation.

Fields

- LFTag – A [LFTagPair \(p. 344\)](#) object.

The key-name of the LF-tag.

- Error – An [ErrorDetail \(p. 370\)](#) object.

An error that occurred with the attachment or detachment of the LF-tag.

ColumnLFTag Structure

A structure containing the name of a column resource and the LF-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 \(p. 370\)](#).

The name of a column resource.

- LFTags – An array of [LFTagPair \(p. 344\)](#) objects, not less than 1 or more than 50 structures.

The LF-tags attached to a column resource.

Operations

- AddLFTagsToResource Action ([Python: add_lf_tags_to_resource](#)) (p. 345)
- RemoveLFTagsFromResource Action ([Python: remove_lf_tags_from_resource](#)) (p. 345)
- GetResourceLFTags Action ([Python: get_resource_lf_tags](#)) (p. 346)

- [ListLFTags Action \(Python: list_lf_tags\) \(p. 347\)](#)
- [CreateLFTag Action \(Python: create_lf_tag\) \(p. 348\)](#)
- [GetLFTag Action \(Python: get_lf_tag\) \(p. 348\)](#)
- [UpdateLFTag Action \(Python: update_lf_tag\) \(p. 349\)](#)
- [DeleteLFTag Action \(Python: delete_lf_tag\) \(p. 350\)](#)
- [SearchTablesByLFTags Action \(Python: search_tables_by_lf_tags\) \(p. 351\)](#)
- [SearchDatabasesByLFTags Action \(Python: search_databases_by_lf_tags\) \(p. 352\)](#)

AddLFTagsToResource Action (Python: add_lf_tags_to_resource)

Attaches one or more LF-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 \(p. 370\)](#).
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 \(p. 326\)](#) object.
The database, table, or column resource to which to attach an LF-tag.
- **LFTags** – *Required:* An array of [LFTagPair \(p. 344\)](#) objects, not less than 1 or more than 50 structures.
The LF-tags to attach to the resource.

Response

- **Failures** – An array of [LFTagError \(p. 344\)](#) objects.
A list of failures to tag the resource.

Errors

- `EntityNotFoundException`
- `InvalidArgumentException`
- `InternalServiceException`
- `OperationTimeoutException`
- `AccessDeniedException`
- `ConcurrentModificationException`

RemoveLFTagsFromResource Action (Python: remove_lf_tags_from_resource)

Removes an LF-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 \(p. 370\)](#).

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 \(p. 326\)](#) object.

The database, table, or column resource where you want to remove an LF-tag.

- LFTags – *Required*: An array of [LFTagPair \(p. 344\)](#) objects, not less than 1 or more than 50 structures.

The LF-tags to be removed from the resource.

Response

- Failures – An array of [LFTagError \(p. 344\)](#) 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 LF-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 \(p. 370\)](#).

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 \(p. 326\)](#) object.

The database, table, or column resource for which you want to return LF-tags.

- ShowAssignedLFTags – Boolean.

Indicates whether to show the assigned LF-tags.

Response

- LFTagOnDatabase – An array of [LFTagPair \(p. 344\)](#) objects, not less than 1 or more than 50 structures.
A list of LF-tags applied to a database resource.
- LFTagsOnTable – An array of [LFTagPair \(p. 344\)](#) objects, not less than 1 or more than 50 structures.
A list of LF-tags applied to a table resource.
- LFTagsOnColumns – An array of [ColumnLFTag \(p. 344\)](#) objects.
A list of LF-tags applied to a column resource.

Errors

- EntityNotFoundException
- InvalidInputException
- InternalServiceException
- OperationTimeoutException
- GlueEncryptionException
- AccessDeniedException

ListLFTags Action (Python: list_lf_tags)

Lists LF-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 \(p. 370\)](#).

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 LF-tags and shared LF-tags that the requester has permission to view. If resource share type is FOREIGN, returns all share LF-tags that the requester can view. If no resource share type is passed, lists LF-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 \(p. 344\)](#) objects, not less than 1 or more than 50 structures.

A list of LF-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`
- `AccessDeniedException`

CreateLFTag Action (Python: `create_lf_tag`)

Creates an LF-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 \(p. 370\)](#).

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, matching the [Custom string pattern #9 \(p. 370\)](#).

The key-name for the LF-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 an LF-tag definition.

Request

- CatalogId – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

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, matching the [Custom string pattern #9 \(p. 370\)](#).

The key-name for the LF-tag.

Response

- CatalogId – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

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, matching the [Custom string pattern #9 \(p. 370\)](#).

The key-name for the LF-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 LF-tag key. If the LF-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 LF-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 \(p. 370\)](#).

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, matching the [Custom string pattern #9 \(p. 370\)](#).

The key-name for the LF-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 LF-tag values to delete from the LF-tag.

- TagValuesToAdd – An array of UTF-8 strings, not less than 1 or more than 50 strings.

A list of LF-tag values to add from the LF-tag.

Response

- *No Response parameters.*

Errors

- EntityNotFoundException
- InvalidInputException
- InternalServiceException
- OperationTimeoutException
- ConcurrentModificationException
- AccessDeniedException

DeleteLFTag Action (Python: delete_lf_tag)

Deletes the specified LF-tag key name. If the attribute key does not exist or the LF-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 LF-tag key is still attached with resources. You can consider untagging resources with this LF-tag key.

Request

- CatalogId – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

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, matching the [Custom string pattern #9 \(p. 370\)](#).

The key-name for the LF-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 LF-tags. 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 \(p. 370\)](#).

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 \(p. 343\)](#) 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 \(p. 343\)](#) objects.
A list of tables that meet the LF-tag conditions.

Errors

- EntityNotFoundException
- InternalServiceException
- InvalidInputException
- OperationTimeoutException
- GlueEncryptionException
- AccessDeniedException

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 \(p. 370\)](#).

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 \(p. 343\)](#) 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 \(p. 343\)](#) objects.
A list of databases that meet the LF-tag conditions.

Errors

- [EntityNotFoundException](#)
- [InternalServiceException](#)
- [InvalidInputException](#)
- [OperationTimeoutException](#)
- [GlueEncryptionException](#)
- [AccessDeniedException](#)

Transaction APIs

The Transaction APIs describes operations for transactionally updating the contents of a table in AWS Lake Formation.

Data Types

- [TransactionDescription Structure \(p. 353\)](#)
- [VirtualObject Structure \(p. 353\)](#)

TransactionDescription Structure

A structure that contains information about a transaction.

Fields

- **TransactionId** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The ID of the transaction.

- **TransactionStatus** – UTF-8 string (valid values: ACTIVE | COMMITTED | ABORTED | COMMIT_IN_PROGRESS).

A status of ACTIVE, COMMITTED, or ABORTED.

- **TransactionStartTime** – Timestamp.

The time when the transaction started.

- **TransactionEndTime** – Timestamp.

The time when the transaction committed or aborted, if it is not currently active.

VirtualObject Structure

An object that defines an Amazon S3 object to be deleted if a transaction cancels, provided that `VirtualPut` was called before writing the object.

Fields

- **Uri** – *Required*: Uniform resource identifier (uri), not less than 1 or more than 1024 bytes long, matching the [URI address multi-line string pattern \(p. 370\)](#).

The path to the Amazon S3 object. Must start with `s3://`

- **ETag** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The ETag of the Amazon S3 object.

Operations

- [StartTransaction Action \(Python: start_transaction\) \(p. 354\)](#)
- [CommitTransaction Action \(Python: commit_transaction\) \(p. 354\)](#)
- [CancelTransaction Action \(Python: cancel_transaction\) \(p. 355\)](#)
- [ExtendTransaction Action \(Python: extend_transaction\) \(p. 355\)](#)
- [DescribeTransaction Action \(Python: describe_transaction\) \(p. 356\)](#)
- [ListTransactions Action \(Python: list_transactions\) \(p. 356\)](#)

- [DeleteObjectsOnCancel Action \(Python: delete_objects_on_cancel\) \(p. 357\)](#)

StartTransaction Action (Python: start_transaction)

Starts a new transaction and returns its transaction ID. Transaction IDs are opaque objects that you can use to identify a transaction.

Request

- **TransactionType** – UTF-8 string (valid values: READ_AND_WRITE | READ_ONLY).

Indicates whether this transaction should be read only or read and write. Writes made using a read-only transaction ID will be rejected. Read-only transactions do not need to be committed.

Response

- **TransactionId** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

An opaque identifier for the transaction.

Errors

- [InternalServiceException](#)
- [OperationTimeoutException](#)

CommitTransaction Action (Python: commit_transaction)

Attempts to commit the specified transaction. Returns an exception if the transaction was previously aborted. This API action is idempotent if called multiple times for the same transaction.

Request

- **TransactionId** – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The transaction to commit.

Response

- **TransactionStatus** – UTF-8 string (valid values: ACTIVE | COMMITTED | ABORTED | COMMIT_IN_PROGRESS).

The status of the transaction.

Errors

- [InvalidInputException](#)
- [EntityNotFoundException](#)
- [InternalServiceException](#)

- OperationTimeoutException
- TransactionCanceledException
- ConcurrentModificationException

CancelTransaction Action (Python: cancel_transaction)

Attempts to cancel the specified transaction. Returns an exception if the transaction was previously committed.

Request

- TransactionId – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The transaction to cancel.

Response

- *No Response parameters.*

Errors

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- TransactionCommittedException
- TransactionCommitInProgressException
- ConcurrentModificationException

ExtendTransaction Action (Python: extend_transaction)

Indicates to the service that the specified transaction is still active and should not be treated as idle and aborted.

Write transactions that remain idle for a long period are automatically aborted unless explicitly extended.

Request

- TransactionId – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The transaction to extend.

Response

- *No Response parameters.*

Errors

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `TransactionCommittedException`
- `TransactionCanceledException`
- `TransactionCommitInProgressException`

DescribeTransaction Action (Python: `describe_transaction`)

Returns the details of a single transaction.

Request

- `TransactionId` – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The transaction for which to return status.

Response

- `TransactionDescription` – A [TransactionDescription \(p. 353\)](#) object.

Returns a `TransactionDescription` object containing information about the transaction.

Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

ListTransactions Action (Python: `list_transactions`)

Returns metadata about transactions and their status. To prevent the response from growing indefinitely, only uncommitted transactions and those available for time-travel queries are returned.

This operation can help you identify uncommitted transactions or to get information about transactions.

Request

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The catalog for which to list transactions. Defaults to the account ID of the caller.

- `StatusFilter` – UTF-8 string (valid values: ALL | COMPLETED | ACTIVE | COMMITTED | ABORTED).

A filter indicating the status of transactions to return. Options are ALL | COMPLETED | COMMITTED | ABORTED | ACTIVE. The default is ALL.

- MaxResults – Number (integer), not less than 1 or more than 1000.

The maximum number of transactions to return in a single call.

- NextToken – UTF-8 string, not more than 4096 bytes long.

A continuation token if this is not the first call to retrieve transactions.

Response

- Transactions – An array of [TransactionDescription \(p. 353\)](#) objects.

A list of transactions. The record for each transaction is a TransactionDescription object.

- NextToken – UTF-8 string, not more than 4096 bytes long.

A continuation token indicating whether additional data is available.

Errors

- [InvalidArgumentException](#)
- [InternalServiceException](#)
- [OperationTimeoutException](#)

[DeleteObjectsOnCancel Action \(Python: delete_objects_on_cancel\)](#)

For a specific governed table, provides a list of Amazon S3 objects that will be written during the current transaction and that can be automatically deleted if the transaction is canceled. Without this call, no Amazon S3 objects are automatically deleted when a transaction cancels.

The AWS Glue ETL library function `write_dynamic_frame.from_catalog()` includes an option to automatically call `DeleteObjectsOnCancel` before writes. For more information, see [Rolling Back Amazon S3 Writes](#).

Request

- CatalogId – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The AWS Glue data catalog that contains the governed table. Defaults to the current account ID.

- DatabaseName – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The database that contains the governed table.

- TableName – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The name of the governed table.

- TransactionId – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

ID of the transaction that the writes occur in.

- **Objects – Required:** An array of [VirtualObject \(p. 353\)](#) objects, not less than 1 or more than 100 structures.

A list of VirtualObject structures, which indicates the Amazon S3 objects to be deleted if the transaction cancels.

Response

- *No Response parameters.*

Errors

- InternalServiceException
- InvalidInputException
- OperationTimeoutException
- EntityNotFoundException
- TransactionCommittedException
- TransactionCanceledException
- ResourceNotReadyException
- ConcurrentModificationException

Exceptions

- [TransactionCommitInProgressException Structure \(p. 358\)](#)
- [TransactionAbortedException Structure \(p. 358\)](#)
- [TransactionCommittedException Structure \(p. 359\)](#)
- [TransactionCanceledException Structure \(p. 359\)](#)
- [TransactionContentionException Structure \(p. 359\)](#)
- [ResourceNotReadyException Structure \(p. 359\)](#)

TransactionCommitInProgressException Structure

Contains details about an error related to a transaction commit that was in progress.

Fields

- **Message – UTF-8 string.**

A message describing the error.

TransactionAbortedException Structure

Contains details about an error where the specified transaction has already been aborted and cannot be used for `UpdateTableObjects`.

Fields

- Message – UTF-8 string.
A message describing the error.

TransactionCommittedException Structure

Contains details about an error where the specified transaction has already been committed and cannot be used for `UpdateTableObjects`.

Fields

- Message – UTF-8 string.
A message describing the error.

TransactionCanceledException Structure

Contains details about an error related to a transaction that was cancelled.

Fields

- Message – UTF-8 string.
A message describing the error.

TransactionContentionException Structure

Contains details about a retryable error indicating that the commit did not succeed due to a contention or conflict.

Fields

- Message – UTF-8 string.
A message describing the error.

ResourceNotReadyException Structure

Contains details about an error related to a resource which is not ready for a transaction.

Fields

- Message – UTF-8 string.
A message describing the error.

Object APIs

The Object APIs describes governed table objects in AWS Lake Formation.

Data Types

- [TableObject Structure \(p. 360\)](#)
- [PartitionObjects Structure \(p. 360\)](#)
- [AddObjectInput Structure \(p. 360\)](#)
- [DeleteObjectInput Structure \(p. 361\)](#)
- [WriteOperation Structure \(p. 361\)](#)

TableObject Structure

Specifies the details of a governed table.

Fields

- **Uri** – Uniform resource identifier (uri), not less than 1 or more than 1024 bytes long, matching the [URI address multi-line string pattern \(p. 370\)](#).

The Amazon S3 location of the object.

- **ETag** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The Amazon S3 ETag of the object. Returned by `GetTableObjects` for validation and used to identify changes to the underlying data.

- **Size** – Number (long).

The size of the Amazon S3 object in bytes.

PartitionObjects Structure

A structure containing a list of partition values and table objects.

Fields

- **PartitionValues** – An array of UTF-8 strings, not less than 1 or more than 100 strings.

A list of partition values.

- **Objects** – An array of [TableObject \(p. 360\)](#) objects.

A list of table objects

AddObjectInput Structure

A new object to add to the governed table.

Fields

- **Uri** – *Required:* Uniform resource identifier (uri), not less than 1 or more than 1024 bytes long, matching the [URI address multi-line string pattern \(p. 370\)](#).

The Amazon S3 location of the object.

- **ETag** – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The Amazon S3 ETag of the object. Returned by `GetTableObjects` for validation and used to identify changes to the underlying data.

- **Size** – *Required*: Number (long).

The size of the Amazon S3 object in bytes.

- **PartitionValues** – An array of UTF-8 strings, not less than 1 or more than 100 strings.

A list of partition values for the object. A value must be specified for each partition key associated with the table.

The supported data types are integer, long, date(yyyy-MM-dd), timestamp(yyyy-MM-dd HH:mm:ssXXX or yyyy-MM-dd HH:mm:ss"), string and decimal.

DeleteObjectInput Structure

An object to delete from the governed table.

Fields

- **Uri** – *Required*: Uniform resource identifier (uri), not less than 1 or more than 1024 bytes long, matching the [URI address multi-line string pattern \(p. 370\)](#).

The Amazon S3 location of the object to delete.

- **ETag** – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The Amazon S3 ETag of the object. Returned by `GetTableObjects` for validation and used to identify changes to the underlying data.

- **PartitionValues** – An array of UTF-8 strings, not less than 1 or more than 100 strings.

A list of partition values for the object. A value must be specified for each partition key associated with the governed table.

WriteOperation Structure

Defines an object to add to or delete from a governed table.

Fields

- **AddObject** – An [AddObjectInput \(p. 360\)](#) object.

A new object to add to the governed table.

- **DeleteObject** – A [DeleteObjectInput \(p. 361\)](#) object.

An object to delete from the governed table.

Operations

- [GetTableObjects Action \(Python: get_table_objects\) \(p. 362\)](#)

- [UpdateTableObjects Action \(Python: update_table_objects\) \(p. 363\)](#)

GetTableObjects Action (Python: get_table_objects)

Returns the set of Amazon S3 objects that make up the specified governed table. A transaction ID or timestamp can be specified for time-travel queries.

Request

- CatalogId – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The catalog containing the governed table. Defaults to the caller's account.

- DatabaseName – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The database containing the governed table.

- TableName – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The governed table for which to retrieve objects.

- TransactionId – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The transaction ID at which to read the governed table contents. If this transaction has aborted, an error is returned. If not set, defaults to the most recent committed transaction. Cannot be specified along with QueryAsOfTime.

- QueryAsOfTime – Timestamp.

The time as of when to read the governed table contents. If not set, the most recent transaction commit time is used. Cannot be specified along with TransactionId.

- PartitionPredicate – Predicate string, not more than 2048 bytes long, matching the [URI address multi-line string pattern \(p. 370\)](#).

A predicate to filter the objects returned based on the partition keys defined in the governed table.

- The comparison operators supported are: =, >, <, >=, <=
- The logical operators supported are: AND
- The data types supported are integer, long, date(yyyy-MM-dd), timestamp(yyyy-MM-dd HH:mm:ssXXX or yyyy-MM-dd HH:mm:ss"), string and decimal.
- MaxResults – Number (integer), not less than 1 or more than 1000.

Specifies how many values to return in a page.

- NextToken – UTF-8 string, not more than 4096 bytes long.

A continuation token if this is not the first call to retrieve these objects.

Response

- Objects – An array of [PartitionObjects \(p. 360\)](#) objects.

A list of objects organized by partition keys.

- NextToken – UTF-8 string, not more than 4096 bytes long.

A continuation token indicating whether additional data is available.

Errors

- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`
- `TransactionCommittedException`
- `TransactionCanceledException`
- `ResourceNotReadyException`

UpdateTableObjects Action (Python: update_table_objects)

Updates the manifest of Amazon S3 objects that make up the specified governed table.

Request

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The catalog containing the governed table to update. Defaults to the caller's account ID.

- `DatabaseName` – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The database containing the governed table to update.

- `TableName` – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The governed table to update.

- `TransactionId` – *Required:* UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Custom string pattern #11 \(p. 370\)](#).

The transaction at which to do the write.

- `WriteOperations` – *Required:* An array of [WriteOperation \(p. 361\)](#) objects, not less than 1 or more than 100 structures.

A list of `WriteOperation` objects that define an object to add to or delete from the manifest for a governed table.

Response

- *No Response parameters.*

Errors

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`
- `EntityNotFoundException`
- `TransactionCommittedException`

- `TransactionCanceledException`
- `TransactionCommitInProgressException`
- `ResourceNotReadyException`
- `ConcurrentModificationException`

Data Filter APIs

The Data Filter APIs describes how to manage data cell filters in AWS Lake Formation.

Data Types

- [DataCellsFilter Structure \(p. 364\)](#)
- [RowFilter Structure \(p. 364\)](#)

DataCellsFilter Structure

A structure that describes certain columns on certain rows.

Fields

- `TableCatalogId` – *Required*: Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The ID of the catalog to which the table belongs.

- `DatabaseName` – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

A database in the AWS Glue Data Catalog.

- `TableName` – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

A table in the database.

- `Name` – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The name given by the user to the data filter cell.

- `RowFilter` – A [RowFilter \(p. 364\)](#) object.

A PartiQL predicate.

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

A list of column names.

- `ColumnWildcard` – A [ColumnWildcard \(p. 330\)](#) object.

A wildcard with exclusions.

You must specify either a `ColumnNames` list or the `ColumnWildcard`.

RowFilter Structure

A PartiQL predicate.

Fields

- **FilterExpression** – Predicate string, not more than 2048 bytes long, matching the [URI address multi-line string pattern \(p. 370\)](#).
A filter expression.
- **AllRowsWildcard** – An empty-structure named AllRowsWildcard.
A wildcard for all rows.

Operations

- [CreateDataCellsFilter Action \(Python: create_data_cells_filter\) \(p. 365\)](#)
- [DeleteDataCellsFilter Action \(Python: delete_data_cells_filter\) \(p. 365\)](#)
- [ListDataCellsFilter Action \(Python: list_data_cells_filter\) \(p. 366\)](#)

CreateDataCellsFilter Action (Python: create_data_cells_filter)

Creates a data cell filter to allow one to grant access to certain columns on certain rows.

Request

- **TableData** – *Required:* A [DataCellsFilter \(p. 364\)](#) object.
A DataCellsFilter structure containing information about the data cells filter.

Response

- *No Response parameters.*

Errors

- `AlreadyExistsException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `AccessDeniedException`

DeleteDataCellsFilter Action (Python: delete_data_cells_filter)

Deletes a data cell filter.

Request

- **TableData** – A [DataCellsFilter \(p. 364\)](#) object.

A DataCellsFilter structure containing information about the data cells filter.

- TableCatalogId – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The ID of the catalog to which the table belongs.

- DatabaseName – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

A database in the AWS Glue Data Catalog.

- TableName – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

A table in the database.

- Name – UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The name given by the user to the data filter cell.

Response

- *No Response parameters.*

Errors

- `InvalidArgumentException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `AccessDeniedException`

ListDataCellsFilter Action (Python: `list_data_cells_filter`)

Lists all the data cell filters on a table.

Request

- Table – A [TableResource \(p. 327\)](#) object.

A table in the AWS Glue Data Catalog.

- NextToken – UTF-8 string.

A continuation token, if this is a continuation call.

- MaxResults – Number (integer), not less than 1 or more than 1000.

The maximum size of the response.

Response

- DataCellsFilters – An array of [DataCellsFilter \(p. 364\)](#) objects.

A list of DataCellFilter structures.

- **NextToken** – UTF-8 string.

A continuation token, if not all requested data cell filters have been returned.

Errors

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `AccessDeniedException`

Storage APIs

The Storage APIs allow you to manage storage optimization on governed tables in AWS Lake Formation.

Data Types

- [StorageOptimizer Structure \(p. 367\)](#)

StorageOptimizer Structure

A structure describing the configuration and details of a storage optimizer.

Fields

- **StorageOptimizerType** – UTF-8 string (valid values: `compaction="COMPACTION"` | `garbage_collection="GARBAGE_COLLECTION"` | `index="INDEX"` | `copy_on_write="COPY_ON_WRITE"` | `all="GENERIC"`).

The specific type of storage optimizer. The supported value is `compaction`.

- **Config** – A map array of key-value pairs.

Each key is a UTF-8 string.

Each value is a UTF-8 string.

A map of the storage optimizer configuration. Currently contains only one key-value pair: `is_enabled` indicates true or false for acceleration.

- **ErrorMessage** – UTF-8 string.

A message that contains information about any error (if present).

When an acceleration result has an enabled status, the error message is empty.

When an acceleration result has a disabled status, the message describes an error or simply indicates "disabled by the user".

- **Warnings** – UTF-8 string.

A message that contains information about any warnings (if present).

- **LastRunDetails** – UTF-8 string.

When an acceleration result has an enabled status, contains the details of the last job run.

Operations

- [ListTableStorageOptimizers Action \(Python: list_table_storage_optimizers\) \(p. 368\)](#)
- [UpdateTableStorageOptimizer Action \(Python: update_table_storage_optimizer\) \(p. 369\)](#)

ListTableStorageOptimizers Action (Python: list_table_storage_optimizers)

Returns the configuration of all storage optimizers associated with a specified table.

Request

- CatalogId – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The Catalog ID of the table.

- DatabaseName – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

Name of the database where the table is present.

- TableName – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

Name of the table.

- StorageOptimizerType – UTF-8 string (valid values: compaction="COMPACTION" | garbage_collection="GARBAGE_COLLECTION" | index="INDEX" | copy_on_write="COPY_ON_WRITE" | all="GENERIC").

The specific type of storage optimizers to list. The supported value is compaction.

- MaxResults – Number (integer), not less than 1 or more than 1000.

The number of storage optimizers to return on each call.

- NextToken – UTF-8 string.

A continuation token, if this is a continuation call.

Response

- StorageOptimizerList – An array of [StorageOptimizer \(p. 367\)](#) objects.

A list of the storage optimizers associated with a table.

- NextToken – UTF-8 string.

A continuation token for paginating the returned list of tokens, returned if the current segment of the list is not the last.

Errors

- [EntityNotFoundException](#)

- `InvalidArgumentException`
- `AccessDeniedException`
- `InternalServiceException`

UpdateTableStorageOptimizer Action (Python: update_table_storage_optimizer)

Updates the configuration of the storage optimizers for a table.

Request

- `CatalogId` – Catalog id string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

The Catalog ID of the table.

- `DatabaseName` – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

Name of the database where the table is present.

- `TableName` – *Required*: UTF-8 string, not less than 1 or more than 255 bytes long, matching the [Single-line string pattern \(p. 370\)](#).

Name of the table for which to enable the storage optimizer.

- `StorageOptimizerConfig` – *Required*: A map array of key-value pairs.

Each key is a UTF-8 string (valid values: `compaction="COMPACTION"` | `garbage_collection="GARBAGE_COLLECTION"` | `index="INDEX"` | `copy_on_write="COPY_ON_WRITE"` | `all="GENERIC"`).

Each value is a A map array of key-value pairs.

Each key is a UTF-8 string.

Each value is a UTF-8 string.

Name of the table for which to enable the storage optimizer.

Response

- `Result` – UTF-8 string.

A response indicating the success or failure of the operation.

Errors

- `EntityNotFoundException`
- `InvalidArgumentException`
- `AccessDeniedException`
- `InternalServiceException`

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 \(p. 370\)](#).

The code associated with this error.
- ErrorMessage – Description string, not more than 2048 bytes long, matching the [URI address multi-line string pattern \(p. 370\)](#).

A message describing the error.

String Patterns

The API uses the following regular expressions to define what is valid content for various string parameters and members:

- Single-line string pattern – "[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*"
- URI address multi-line string pattern – "[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\r\n\t]*"
- Custom string pattern #3 – "^\\w+\\.\\w+\\.\\w+\$"
- Custom string pattern #4 – "^\w+\\.\\w+\$"
- Custom string pattern #5 – "arn:aws:iam::[0-9]*:role/.*"
- Custom string pattern #6 – "arn:aws:iam::[0-9]*:user/.*"
- Custom string pattern #7 – "arn:aws:iam::[0-9]*:group/.*"
- Custom string pattern #8 – "arn:aws:iam::[0-9]*:saml-provider/.*"
- Custom string pattern #9 – "^(\\p{L}\\p{Z}\\p{N}_.:\\/=?\\-@%)*\$"
- Custom string pattern #10 – "^(\\p{L}\\p{Z}\\p{N}_.:*\\/=?\\-@%)*\$"
- Custom string pattern #11 – "[\\p{L}\\p{N}\\p{P}]*"

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" \(p. 219\)](#).

AWS Lake Formation Personas

The following table lists the suggested AWS Lake Formation personas.

Lake Formation Personas

| Persona | Description |
|-------------------------------|--|
| IAM administrator (superuser) | (Required) User who can create IAM users and roles. Has the AdministratorAccess AWS managed policy. Has all 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 Creating a Database (p. 119) . |
| 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 \(p. 372\)](#)
- [Data Engineer Permissions \(p. 373\)](#)
- [Data Analyst Permissions \(p. 375\)](#)

- [Workflow Role Permissions \(p. 375\)](#)

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 \(p. 375\)](#).

| Policy Type | Policy |
|--|---|
| AWS managed policies | <ul style="list-style-type: none"> • <code>AWSLakeFormationDataAdmin</code> • <code>AWSGlueConsoleFullAccess</code> (Optional) • <code>CloudWatchLogsReadOnlyAccess</code> (Optional) • <code>AWSLakeFormationCrossAccountManager</code> (Optional) • <code>AmazonAthenaFullAccess</code> (Optional) <p>For information about the optional AWS managed policies, see the section called "Create a data lake administrator" (p. 12).</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::<account- id>:role/aws-service-role/lakeformation.amazonaws.com/ AWSServiceRoleForLakeFormationDataAccess" }] }</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::<account-id>:role/<workflow_role>"] }] }</pre> |

| Policy Type | Policy |
|---|--|
| (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 | AWSGlueConsoleFullAccess |
| Inline policy (basic) | <pre>{ "Version": "2012-10-17", "Statement": [{ "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", "lakeformation:GetWorkUnits", "lakeformation:GetWorkUnitResults", "lakeformation:StartQueryPlanning", "lakeformation:GetQueryState", "lakeformation:GetQueryStatistics"], "Resource": "*" }] }</pre> |

| Policy Type | Policy |
|--|--|
| Inline policy (for operations on governed tables, including operations within transactions) | <pre data-bbox="556 255 1486 340">] }</pre> |
| Inline policy (for operations on governed tables, including operations within transactions) | <pre data-bbox="556 340 1486 903">{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["lakeformation:StartTransaction", "lakeformation:CommitTransaction", "lakeformation:CancelTransaction", "lakeformation:ExtendTransaction", "lakeformation:DescribeTransaction", "lakeformation>ListTransactions", "lakeformation:GetTableObjects", "lakeformation:UpdateTableObjects", "lakeformation>DeleteObjectsOnCancel"], "Resource": "*" }] }</pre> |
| Inline policy (for metadata access control using the Lake Formation tag-based access control (LF-TBAC) method) | <pre data-bbox="556 903 1486 1417">{ "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 data-bbox="556 1417 1486 1856">{ "Version": "2012-10-17", "Statement": [{ "Sid": "PassRolePermissions", "Effect": "Allow", "Action": ["iam:PassRole"], "Resource": ["arn:aws:iam::<account-id>:role/<workflow_role>"] }] }</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> |
| (Optional) Inline policy (for operations on governed tables, including operations within transactions) | <pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["lakeformation:StartTransaction", "lakeformation:CommitTransaction", "lakeformation:CancelTransaction", "lakeformation:ExtendTransaction", "lakeformation:DescribeTransaction", "lakeformation>ListTransactions", "lakeformation:GetTableObjects", "lakeformation:UpdateTableObjects", "lakeformation>DeleteObjectsOnCancel"], "Resource": "*" }] }</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 | <code>AWSGlueServiceRole</code> |
| Inline policy (data access) | <pre>{ "Version": "2012-10-17", "Statement": [{ "Sid": "Lakeformation", "Effect": "Allow", "Action": ["lakeformation:GetDataAccess", "lakeformation:GrantPermissions"], "Resource": "*" }] }</pre> |
| Inline policy (passrole policy for the workflow role) | <pre>{ "Version": "2012-10-17", "Statement": [{ "Sid": "PassRolePermissions", "Effect": "Allow", "Action": ["iam:PassRole"], "Resource": ["arn:aws:iam::<account-id>:role/<workflow_role>"] }] }</pre> |
| Inline policy (for ingesting data outside the data lake, for example, AWS CloudTrail logs) | <pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["s3:GetObject", "s3>ListBucket"], "Resource": ["arn:aws:s3:::<your-s3-cloudtrail-bucket>/*"] }] }</pre> |

Troubleshooting Lake Formation

If you encounter issues when working with AWS Lake Formation, consult the topics in this section.

Topics

- [General troubleshooting \(p. 377\)](#)
- [Troubleshooting cross-account access \(p. 378\)](#)
- [Troubleshooting blueprints and workflows \(p. 381\)](#)

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 \(p. 294\)](#) and [Granting data location permissions \(p. 174\)](#).

Error: "Insufficient encryption key permissions for Glue API"

An attempt was made to grant Lake Formation permissions without AWS Identity and Access Management (IAM) permissions on the AWS KMS encryption key for an encrypted Data Catalog.

My Amazon Athena or Amazon Redshift query that uses manifests is failing

Lake Formation does not support queries that use manifests.

Error: "Insufficient Lake Formation permission(s): Required create tag on catalog"

The user/role must be a data lake administrator.

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. 378](#))
- Principals in the recipient account can see the Data Catalog resource but can't access the underlying data ([p. 378](#))
- Error: "Association failed because the caller was not authorized" when accepting a AWS RAM resource share invitation ([p. 379](#))
- Error: "Not authorized to grant permissions for the resource" ([p. 379](#))
- Error: "Access denied to retrieve AWS Organization information" ([p. 379](#))
- Error: "Organization <organization-ID> not found" ([p. 379](#))
- Error: "Insufficient Lake Formation permissions: Illegal combination" ([p. 380](#))
- ConcurrentModificationException on grant/revoke requests to external accounts ([p. 380](#))
- Error when using Amazon EMR to access data shared via cross-account ([p. 380](#))

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" \(p. 154\)](#).

- 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" \(p. 216\)](#).

- 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 data sharing prerequisites" \(p. 200\)](#).

- 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 \(p. 212\)](#).

Error: "Association failed because the caller was not authorized" when accepting a AWS RAM resource share invitation

After granting access to a resource to a different account, when the receiving account attempts to accept the resource share invitation, the action fails.

```
$ aws ram get-resource-share-associations --association-type PRINCIPAL --resource-share-arns arn:aws:ram:aws-region:444444444444:resource-share/e1d1f4ba-xxxx-xxxx-xxxx-xxxxxxxxx5d8d
{
    "resourceShareAssociations": [
        {
            "resourceShareArn": "arn:aws:ram:aws-region:444444444444:resource-share/e1d1f4ba-xxxx-xxxx-xxxx-xxxxxxxxx5d8d",
            "resourceShareName": "LakeFormation-MMCC0XQBH3Y",
            "associatedEntity": "5815803XXXXX",
            "associationType": "PRINCIPAL",
            "status": "FAILED",
            "statusMessage": "Association failed because the caller was not authorized.",
            "creationTime": "2021-07-12T02:20:10.267000+00:00",
            "lastUpdatedTime": "2021-07-12T02:20:51.830000+00:00",
            "external": true
        }
    ]
}
```

The error occurs because the `glue:PutResourcePolicy` is invoked by AWS Glue when the receiving account accepts the resource share invitation. To resolve the issue, allow the `glue:PutResourcePolicy` action by the assumed role used by the receiving account.

Error: "Not authorized to grant permissions for the resource"

An attempt was made to grant cross-account permissions on a database or table that is owned by another account. When a database or table is shared with your account, as a data lake administrator, you can grant permissions on it only to users in your account.

Error: "Access denied to retrieve AWS Organization information"

Your account is an AWS Organizations management account and you do not have the required permissions to retrieve organization information, such as organizational units in the account.

For more information, see [Required permissions for cross-account grants \(p. 202\)](#).

Error: "Organization <organization-ID> not found"

An attempt was made to share a resource with an organization, but sharing with organizations is not enabled. Enable resource sharing with organizations.

For more information, see [Enable Sharing with AWS Organizations](#) in the *AWS RAM User Guide*.

Error: "Insufficient Lake Formation permissions: Illegal combination"

A user shared a Data Catalog resource while Lake Formation permissions were granted to the IAMAllowedPrincipals group for the resource. The user must revoke all Lake Formation permissions from IAMAllowedPrincipals before sharing the resource.

ConcurrentModificationException on grant/revoke requests to external accounts

When users make multiple concurrent grant and/or revoke permission requests for a principal on LF-tag policies, then Lake Formation throws ConcurrentModificationException. Users need to catch the exception and retry the failed grant/revoke request. Using batch versions of the GrantPermissions/RevokePermissions API operations - [the section called "BatchGrantPermissions \(batch_grant_permissions\)" \(p. 333\)](#) and [the section called "BatchRevokePermissions \(batch_revoke_permissions\)" \(p. 334\)](#) alleviates this problem to an extent by reducing the number of concurrent grant/revoke requests.

Error when using Amazon EMR to access data shared via cross-account

When you use Amazon EMR to access data shared with you from another account, some Spark libraries will attempt to call `Glue:GetUserDefinedFunctions` API operation. Since versions 1 and 2 of the AWS RAM managed permissions does not support this action, you receive the following error message:

```
"ERROR: User: arn:aws:sts::012345678901:assumed-role/my-spark-role/i-06ab8c2b59299508a is not authorized to perform: glue:GetUserDefinedFunctions on resource: arn:exampleCatalogResource because no resource-based policy allows the glue:GetUserDefinedFunctions action"
```

To resolve this error, the data lake administrator who created the resource share must update the AWS RAM managed permissions attached to the resource share. Version 3 of the AWS RAM managed permissions allows principals to perform the `glue:GetUserDefinedFunctions` action.

If you create a new resource share, Lake Formation applies the latest version of the AWS RAM managed permission by default, and no action is required by you. To enable cross-account data access for existing resource shares, you need to update the AWS RAM managed permissions to version 3.

You can view the AWS RAM permissions assigned to resources shared with you in AWS RAM. The following permissions are included in version 3:

```
Databases
AWSRAMPermissionGlueDatabaseReadWriteForCatalog
AWSRAMPermissionGlueDatabaseReadWrite

Tables
AWSRAMPermissionGlueTableReadWriteForCatalog
AWSRAMPermissionGlueTableReadWriteForDatabase

AllTables
AWSRAMPermissionGlueAllTablesReadWriteForCatalog
AWSRAMPermissionGlueAllTablesReadWriteForDatabase
```

To update AWS RAM managed permissions version of existing resource shares

You (data lake administrator) can either [update AWS RAM managed permissions to a newer version](#) by following instructions in the *AWS RAM User Guide* or you can revoke all existing permissions for the resource type and regrant them. If you revoke permissions, AWS RAM deletes the AWS RAM resource share associated with the resource type. When you regrant permissions, AWS RAM creates new resource shares attaching the latest version of AWS RAM managed permissions.

Troubleshooting blueprints and workflows

Use the information here to help you diagnose and fix blueprint and workflow issues.

Topics

- [My blueprint failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>" \(p. 381\)](#)
- [My workflow failed with "User: <user-ARN> is not authorized to perform: iam:PassRole on resource: <role-ARN>" \(p. 381\)](#)
- [A crawler in my workflow failed with "Resource does not exist or requester is not authorized to access requested permissions" \(p. 381\)](#)
- [A crawler in my workflow failed with "An error occurred \(AccessDeniedException\) when calling the CreateTable operation..." \(p. 382\)](#)

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 \(p. 371\)](#).

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" \(p. 11\)](#).

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" \(p. 225\)](#).

Known issues for AWS Lake Formation

Review these known issues for AWS Lake Formation.

Topics

- [Limitation on filtering of table metadata \(p. 383\)](#)
- [Issue with renaming an excluded column \(p. 384\)](#)
- [Issue with deleting columns in CSV tables \(p. 384\)](#)
- [Table partitions must be added under a common path \(p. 384\)](#)
- [Issue with creating a database during workflow creation \(p. 384\)](#)
- [Issue with deleting and then re-creating a user \(p. 384\)](#)
- [GetTables and SearchTables APIs do not update the value for the IsRegisteredWithLakeFormation parameter \(p. 385\)](#)
- [Data Catalog API operations do not update the value for the IsRegisteredWithLakeFormation parameter \(p. 385\)](#)
- [Lake Formation operations do not support AWS Glue Schema Registry \(p. 385\)](#)

Limitation on filtering of table metadata

AWS Lake Formation column-level permissions can be used to restrict access to specific columns in a table. When a user retrieves metadata about the table using the console or an API like `glue:GetTable`, the column list in the table object contains only the fields to which they have access. It is important to understand the limitations of this metadata filtering.

Although Lake Formation makes available metadata about column permissions to integrated services, the actual filtering of columns in query responses is the responsibility of the integrated service. Lake Formation clients that support column-level filtering, including Amazon Athena, Amazon Redshift Spectrum, and Amazon EMR filter the data based on the column permissions registered with Lake Formation. Users won't be able to read any data to which they should not have access. Currently, AWS Glue ETL doesn't support column filtering.

Note

EMR clusters are not completely managed by AWS. Therefore, it's the responsibility of EMR administrators to properly secure the clusters to avoid unauthorized access to data.

Certain applications or formats might store additional metadata, including column names and types, in the `Parameters` map as table properties. These properties are returned unmodified and are accessible by any user with `SELECT` permission on any column.

For example, the [Avro SerDe](#) stores a JSON representation of the table schema in a table property named `avro.schema.literal`, which is available to all users with access to the table. We recommend that you avoid storing sensitive information in table properties and be aware that users can learn the complete schema of Avro format tables. This limitation is specific to the metadata about a table.

AWS Lake Formation removes any table property beginning with `spark.sql.sources.schema` when responding to a `glue:GetTable` or similar request if the caller does not have `SELECT` permissions on all columns in the table. This prevents users from gaining access to additional metadata about

tables created with Apache Spark. When run on Amazon EMR, Apache Spark applications still can read these tables, but certain optimizations might not be applied, and case-sensitive column names are not supported. If the user has access to all columns in the table, Lake Formation returns the table unmodified with all table properties.

Issue with renaming an excluded column

If you use column-level permissions to exclude a column and then rename the column, the column is no longer excluded from queries, such as `SELECT *`.

Issue with deleting columns in CSV tables

If you create a Data Catalog table with the CSV format and then delete a column from the schema, queries could return erroneous data, and column-level permissions might not be adhered to.

Workaround: Create a new table instead.

Table partitions must be added under a common path

Lake Formation expects all partitions of a table to be under a common path that is set in the table's location field. When you use the crawler to add partitions to a catalog, this works seamlessly. But if you add partitions manually, and these partitions are not under the location set in the parent table, data access does not work.

Issue with creating a database during workflow creation

When creating a workflow from a blueprint using the Lake Formation console, you can create the target database if it doesn't exist. When you do so, the 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.

GetTables and SearchTables APIs do not update the value for the IsRegisteredWithLakeFormation parameter

There is a known limitation that Data Catalog API operations such as GetTables and SearchTables do not update the value for the IsRegisteredWithLakeFormation parameter, and return the default, which is false. It is recommended to use the GetTable API to view the correct value for the IsRegisteredWithLakeFormation parameter.

Data Catalog API operations do not update the value for the IsRegisteredWithLakeFormation parameter

There is a known limitation that Data Catalog API operations such as GetTables and SearchTables do not update the value for the IsRegisteredWithLakeFormation parameter, and return the default, which is false. It is recommended to use the GetTable API to view the correct value for the IsRegisteredWithLakeFormation parameter.

Lake Formation operations do not support AWS Glue Schema Registry

Lake Formation operations do not support AWS Glue tables that contain a SchemaReference in the StorageDescriptor to be utilized in the [Schema Registry](#).

Document history for AWS Lake Formation

The following table describes important changes to the documentation for AWS Lake Formation.

| Change | Description | Date |
|--|---|-------------------|
| Support for cross-account data sharing directly with principals (p. 386) | Added information about sharing data directly with IAM principals in another account. For more information see Cross-account data sharing in AWS Lake Formation . | November 10, 2022 |
| Support for AWS RAM enabled data sharing using TBAC (p. 386) | Added information about The LF-TBAC method of granting Data Catalog permissions use AWS Resource Access Manager for cross-account grants . | November 10, 2022 |
| Added a section on working with other services (p. 386) | Added information on how AWS services such as Athena, AWS Glue, Redshift Spectrum, and Amazon EMR can use Lake Formation to securely access data in Amazon S3 locations registered with Lake Formation. For more information see Working with other AWS services (p. 311) | November 10, 2022 |
| ??? (p. 386) | Added information on troubleshooting an error when using Amazon EMR to access cross-account data. For more information, see Error when using Amazon EMR to access data shared via cross-account (p. 380) . | November 7, 2022 |
| Updates to cross-account resource share (p. 386) | Added a description for how cross-account resource shares work in Lake Formation. Documented the change to the AWSLakeFormationCrossAccountManager policy. | May 6, 2022 |
| New tutorials (p. 386) | Added new tutorials for creating governed tables, securing data lakes, and sharing data lakes. For more details, see Get started section. | April 20, 2022 |

| | | |
|--|---|-------------------|
| New Lake Formation landing page (p. 386) | Updated the Lake Formation landing page to include links for tutorials that provide step-by-step instructions on how to build a data lake, ingest data, share, and secure data lakes using Lake Formation. | April 20, 2022 |
| Support for credential vending (p. 386) | Added information about credential vending, which supports Lake Formation to allow third-party services to integrate with Lake Formation by using credential vending API operations. For more information, see How credential vending works in Lake Formation . | February 28, 2022 |
| Support for governed tables and advanced data filtering (p. 386) | Added information about governed tables, which support ACID transactions, automatic data compaction, and time-travel queries. Added information about creating data filters to support for column-level security, row-level security, and cell-level security. For more information, see Governed Tables in Lake Formation and Data Filtering and Cell-Level Security in Lake Formation . | November 30, 2021 |
| Support for VPC interface endpoints (p. 386) | Added information about creating a virtual private cloud (VPC) interface endpoint for Lake Formation, so that communication between your VPC and Lake Formation is conducted entirely and securely within the AWS network. For more information, see Using Lake Formation with VPC Endpoints . | October 11, 2021 |
| Support for VPC endpoint policies (p. 386) | Added information about support for Virtual Private Cloud (VPC) endpoint policies in Lake Formation. For more information, see Using Lake Formation with VPC Endpoints . | October 11, 2021 |

| | | |
|--|--|--------------------|
| Support for tag-based access control (p. 386) | Lake Formation tag-based access control provides a new, more scalable way to manage access to Data Catalog resources and underlying data by using LF-tags. For more information, see Lake Formation Tag-Based Access Control . | May 7, 2021 |
| New opt-in requirement for data filtering on Amazon EMR. (p. 386) | Added information about the requirement to opt in to allow Amazon EMR to filter data that is managed by Lake Formation. For more information, see Allow Data Filtering on Amazon EMR . | October 9, 2020 |
| Support for granting full cross-account permissions on Data Catalog databases (p. 386) | Added information about granting full Lake Formation permissions on Data Catalog databases across AWS accounts, including CREATE_TABLE. For more information, see Sharing Data Catalog Databases . | October 1, 2020 |
| Support for Amazon Athena users authenticating through SAML. (p. 386) | 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 AWS Service Integrations with Lake Formation . | September 30, 2020 |
| Support for cross-account access with an encrypted Data Catalog (p. 386) | Added information about granting cross-account permissions when the Data Catalog is encrypted. For more information, see Cross-Account Access Prerequisites . | July 30, 2020 |
| Support for cross-account access to the data lake (p. 386) | 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 Cross-Account Access . | July 7, 2020 |

| | | |
|---|---|--------------------|
| Integration with Amazon QuickSight (p. 386) | 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 Granting Data Catalog Permissions . | June 29, 2020 |
| Updates to setting up and Getting Started chapters (p. 386) | Reorganized and improved the Setting Up and Getting Started chapters. Updated the recommended AWS Identity and Access Management (IAM) permissions for the data lake administrator. | February 27, 2020 |
| Support for AWS Key Management Service (p. 386) | 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 keys. For more information, see Adding an Amazon S3 location to your data lake (p. 107) . | February 27, 2020 |
| Updates to blueprints and data lake administrator IAM policies (p. 386) | Clarified input parameters for incremental database blueprints. Updated the IAM policies required for a data lake administrator. | December 20, 2019 |
| Security chapter rewrite and upgrade chapter revisions (p. 386) | Improved the security and upgrading chapters. | October 29, 2019 |
| Super permission replaces All permission (p. 386) | Updated the Security and Upgrading chapters to reflect the replacement of the permission All with Super. | October 10, 2019 |
| Additions, corrections, and clarifications (p. 386) | Made additions, corrections, and clarifications based on feedback. Revised the security chapter. Updated the Security and Upgrading chapters to reflect the replacement of the group Everyone with IAMAllowedPrincipals. | September 11, 2019 |

New guide (p. 386)

This is the initial release of the
*AWS Lake Formation Developer
Guide.*

August 8, 2019

AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.