



Amazon Web Services Data Engineering Immersion Day

Lake Formation Lab

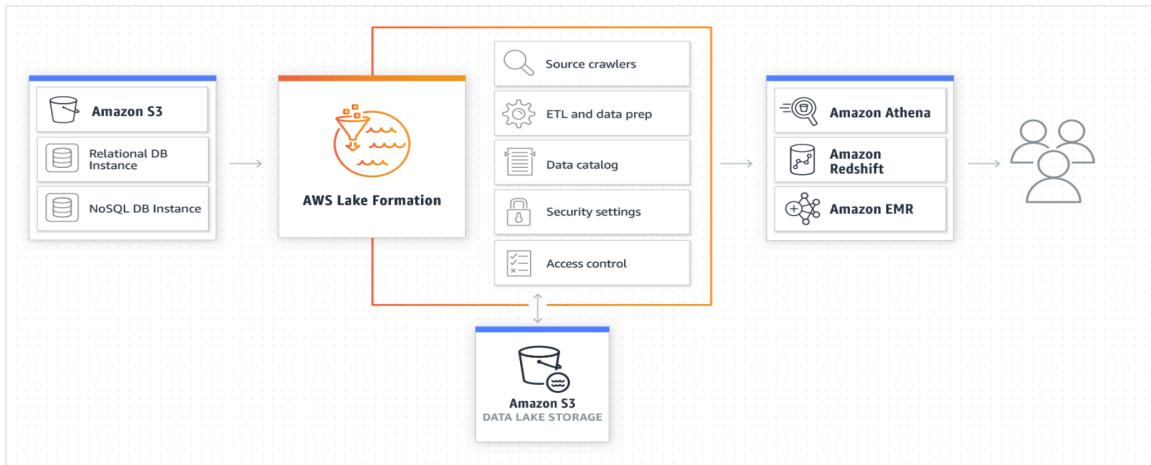
March 2020

Table of Contents

<i>Introduction</i>	3
Prerequisites:	3
Tasks Completed in this Lab:.....	3
Getting Started.....	4
<i>Setup Network Configuration</i>	4
<i>Create an IAM role to use with Lake Formation:</i>	8
<i>Create Glue JDBC connection for RDS</i>	13
<i>Lake Formation – Add Administrator and start workflows using Blueprints.</i>	15
<i>Explore the Underlying Components of a Blueprint</i>	20
<i>Explore workflow results in Athena</i>	21
<i>[Optional] Grant fine grain access controls to Data Lake user</i>	22
<i>[Optional] Verify data permissions using Athena</i>	26

Introduction

This lab will give you an understanding of the AWS Lake Formation – a service that makes it easy to set up a secure data lake in days, as well as Athena for querying the data you import into your data lake.



Prerequisites:

The DMS Lab is a prerequisite for this lab.

Tasks Completed in this Lab:

In this lab you will be completing the following tasks:

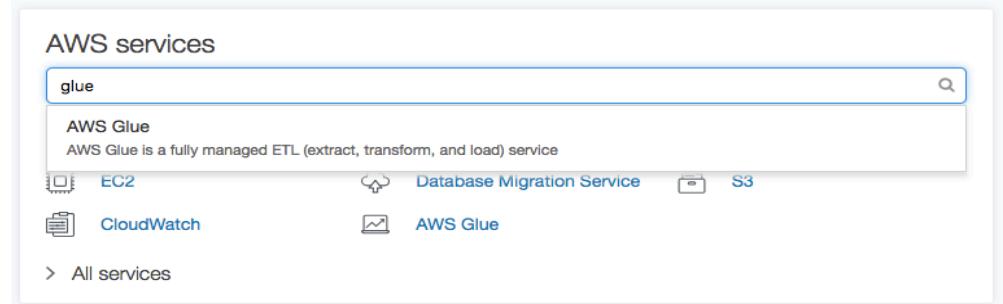
1. [Create a JDBC connection to RDS in AWS Glue](#)
2. [Lake Formation IAM Role](#)

NOTE: If your account is already setup with Glue Data catalog permissions then upgrade to Lake Formation Permission set by following -[Link](#)

3. [Lake Formation – Add Administrator and start workflows using Blueprints](#)
4. [Explore the components of a Glue WorkFlow created by lake formation](#)
5. [Explore workflow results in Athena](#)
6. [Grant fine grain access controls to Data Lake user](#)
7. [Verify data permissions using Athena](#)

Getting Started

Navigate to the AWS Glue service.



Setup Network Configuration

To begin with, we need to perform network configuration. We will open ports specific to RDS PostgreSQL DB, define inbound rules and create S3 endpoint to access data in S3 bucket (created as part of DMS Lab)

1. Navigate to **RDS dashboard** and click on the instance. Select “Connectivity and Security” and look for security group. Click on the **security group** and you will be navigated to VPC Dashboard.

Note: Make sure your RDS instance is **available**. If the Status says “Storage Full”, increase the DB storage capacity and apply changes immediately to the DB instance and proceed further.

A screenshot of the Amazon RDS instance details page for "dmslabinstance". The left sidebar shows navigation options like Dashboard, Databases (selected), Query Editor, etc. The main content area shows the "Summary" and "Connectivity & security" tabs. The "Connectivity & security" tab is active, displaying information about the endpoint, networking, and security groups. A red box highlights the "Security" section under the "Connectivity & security" tab, specifically the "VPC security groups" field which lists "dmslab-instructor-sgrdsLaunchWizard2-PTOG61QKHSX1 (sg-0a7f48756ac11406f) (active)".

2. Security group associated with RDS instance will be automatically selected. Select Inbound Rules and Click **Edit Rules**.

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under the 'Security' section, 'Security Groups' is selected. A search bar at the top right says 'Create security group'. Below it is a table with columns 'Name', 'Group ID', 'Group Name', and 'VPC ID'. One row is highlighted with a blue border: 'DMSLabRD...' (Group ID sg-0a7f48756ac11406f). At the bottom of the page, under 'Inbound Rules', there is a table with columns 'Type', 'Protocol', 'Port Range', and 'Source'. It lists two rules: 'All TCP' (Protocol TCP, Port Range 0 - 65535, Source sg-0a7f48756ac11406f) and 'PostgreSQL' (Protocol TCP, Port Range 5432, Source 0.0.0.0/0).

3. Create Inbound Rules as shown below:

- All TCP with self-reference of security group as its own source.
- PostgreSQL on 5432 from 0.0.0.0/0
- Click on **Save Rules**

[Security Groups > Edit inbound rules](#)

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Type	Protocol	Port Range	Source	Description
All TCP	TCP	0 - 65535	Custom sg-0a7f48756ac11406f	e.g. SSH for Admin Desktop
PostgreSQL	TCP	5432	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

* Required Cancel Save rules

4. Create S3 VPC endpoint. Navigate to **VPC Dashboard** and select **Endpoint** from left section. Click on “Create Endpoint”.

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under 'Virtual Private Cloud', the 'Endpoints' option is highlighted with an orange border. At the top right, there are two buttons: 'Launch VPC Wizard' (blue) and 'Launch EC2 Instances' (grey). Below these buttons is a note: 'Note: Your Instances will launch in the US East (N. Virginia) region.' The main area is titled 'Resources by Region' with a 'Refresh Resources' link. It lists the following resources and their counts in the N. Virginia region:

- VPCs: 4
- Subnets: 12
- Route Tables: 6
- Internet Gateways: 4
- Egress Only Internet Gateways: 0

At the bottom of the dashboard, there is a sub-section for 'Endpoints' with a 'Create Endpoint' button and a note: 'You do not have any Endpoints in this region. Click the Create Endpoint button to create your first Endpoint.'

5. In the next window, follow these steps:
 - a. Service Category: AWS Services
 - b. Service name: com.amazonaws.us-east-1.s3
 - c. VPC: same as VPC for RDS instance
 - d. Route table: in accordance to subnets
 - e. Policy: Full Access
 - f. Click on Create Endpoint

Create Endpoint

A VPC endpoint allows you to securely connect your VPC to another service. An interface endpoint is powered by [PrivateLink](#), and uses an elastic network interface (ENI) as an entry point for traffic destined to the service. A gateway endpoint serves as a target for a route in your route table for traffic destined for the service.

- Service category AWS services
 Find service by name
 Your AWS Marketplace services

Service Name com.amazonaws.us-east-1.s3 [i](#)

Service Name			Owner	Type
com.amazonaws.us-east-1.s3	amazon	Gateway		

VPC* [vpc-0b6d79195f89e9c39](#) [C](#) [i](#)

Configure route tables A rule with destination [pl-63a5400a \(com.amazonaws.us-east-1.s3\)](#) and a target with this endpoints' ID (e.g. vpce-12345678) will be added to the route tables you select below.

Subnets associated with selected route tables will be able to access this endpoint.

rtb-07cb8bc42e0314882 [C](#)

Route Table ID	Main	Associated With
<input checked="" type="checkbox"/> rtb-07cb8bc42e0314882	No	3 subnets
<input type="checkbox"/> rtb-025288763963b3bdc	Yes	0 subnets

⚠ Warning

When you use an endpoint, the source IP addresses from your instances in your affected subnets for accessing the AWS service in the same region will be private IP addresses, not public IP addresses. Existing connections from your affected subnets to the AWS service that use public IP addresses may be dropped. Ensure that you don't have critical tasks running when you create or modify an endpoint.

- Policy* Full Access - Allow access by any user or service within the VPC using credentials from any AWS accounts to any resources in this AWS service. All policies — IAM user policies, VPC endpoint policies, and AWS service-specific policies (e.g. Amazon S3 bucket policies, any S3 ACL policies) — must grant the necessary permissions for access to succeed.

- Custom

Use the [policy creation tool](#) to generate a policy, then paste the generated policy below.

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

Key (128 characters maximum)

Value (256 characters maximum)

This resource currently has no tags

Add Tag 50 remaining (Up to 50 tags maximum)

* Required

[Cancel](#) [Create endpoint](#)

Create Endpoint

✓ The following VPC Endpoint was created:

VPC Endpoint ID vpce-0ca75ab66a6e88d71

Close

Endpoint creation is successful.

Create an IAM role to use with Lake Formation:

Documentation Reference: [Link](#)

Create the IAM Role by following below steps:

1. On the IAM console, In the navigation pane, choose **Roles**, and then choose **Create role**.

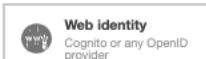
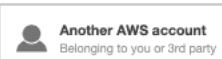
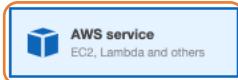
The screenshot shows the AWS IAM Roles page. On the left, there's a navigation pane with various options like Dashboard, Access management, and Roles. The 'Roles' option is highlighted with an orange box. The main content area has a heading 'What are IAM roles?' followed by a list of examples. Below that is a section titled 'Additional resources:' with links to IAM Roles FAQ, Documentation, and Tutorials. At the bottom of the page, there are two buttons: 'Create role' (which is highlighted with an orange box) and 'Delete role'. There's also a search bar at the bottom.

2. On the Create role page, choose **AWS service**, and choose **Glue**. Then choose **Next:Permissions**.

Create role

1 2 3 4

Select type of trusted entity



Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose a use case

Common use cases

EC2

Allows EC2 instances to call AWS services on your behalf.

Lambda

Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

API Gateway	CodeDeploy	EMR	KMS	RoboMaker
AWS Backup	CodeGuru	ElastiCache	Kinesis	S3
AWS Chatbot	CodeStar Notifications	Elastic Beanstalk	Lambda	SMS
AWS Support	Comprehend	Elastic Container Service	Lex	SNS
Amplify	Config	Elastic Transcoder	License Manager	SWF
AppStream 2.0	Connect	Elastic Load Balancing	Machine Learning	SageMaker
AppSync	DMS	Forecast	Macie	Security Hub
Application Auto Scaling	Data Lifecycle Manager	Global Accelerator	MediaConvert	Service Catalog
Application Discovery Service	Data Pipeline	Glue	Migration Hub	Step Functions
Batch	DataSync	Greengrass	OpsWorks	Storage Gateway
Chime	DeepLens	GuardDuty	Personalize	Textract
CloudFormation	DynamoDB	Health Organizational View	QLDB	Transfer
CloudHSM	EC2	IAM Access Analyzer	RAM	Trusted Advisor
CloudTrail	EC2 - Fleet	Inspector	RDS	VPC
CloudWatch Application Insights	EC2 Auto Scaling	IoT	Redshift	WorkLink
CloudWatch Events	EKS	IoT Things Graph	Rekognition	WorkMail
CodeBuild				

Select your use case

Glue

Allows Glue to call AWS services on your behalf.

* Required

Cancel

Next: Permissions

3. Search for and select the **AWSGlueServiceRole** managed policy. Optionally, provide tags.

Create role

1 2 3 4

Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy

Filter policies ▾ Showing 1 result

	Policy name ▾	Used as
<input checked="" type="checkbox"/>	AWSGlueServiceRole	Permissions policy (1)

4. Then complete the wizard, naming the role **LakeFormationWorkflowRole**, and choose **Create role**.

Create role

1 2 3 4

Review

Provide the required information below and review this role before you create it.

Role name* Use alphanumeric and '+,-,@-' characters. Maximum 64 characters.

Role description

Maximum 1000 characters. Use alphanumeric and '+,-,@-' characters.

Trusted entities AWS service: glue.amazonaws.com

Policies AWSGlueServiceRole

Permissions boundary Permissions boundary is not set

No tags were added.

* Required Cancel Previous Create role

5. Back on the **Roles page**, search for and choose **LakeFormationWorkflowRole**.
 6. On the role **Summary page**, under the **Permissions** tab, choose **Add inline policy**, and add the following policy in JSON editor.

Roles > LakeFormationWorkflowRole

Summary

[Delete role](#)

Role ARN	arn:aws:iam::678691952726:role/LakeFormationWorkflowRole
Role description	Allows Glue to call AWS services on your behalf. Edit
Instance Profile ARNs	View
Path	/
Creation time	2020-03-23 16:07 EDT
Last activity	Not accessed in the tracking period
Maximum CLI/API session duration	1 hour Edit

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

▼ Permissions policies (1 policy applied)

[Attach policies](#) [Add inline policy](#)

Policy name	Policy type	X
AWSGlueServiceRole	AWS managed policy	X

▶ Permissions boundary (not set)

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

7. Name the policy **DatalakeDataAccess** and click on **Create Policy**

Create policy

1 2

Review policy

Before you create this policy, provide the required information and review this policy.

Name* Maximum 128 characters. Use alphanumeric and '+,-,.,@,_' characters.

Summary			
<input type="text" value="Filter"/>			
Service ▾	Access level	Resource	Request condition
Allow (1 of 224 services) Show remaining 223			
IAM	Limited: Write	RoleName string like LakeFormationWorkflowRole	None

Required Cancel Previous **Create policy**

8. Add another inline policy that allows the role to pass itself by granting the **PassRole** permission. Name the policy **DatalakePassRole**.

Important: In the following policy, replace account-id with your AWS account number. Account ID can be found in Account Summary page.

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

9. Back to policy page, add two more AWS service level permissions:

AmazonS3FullAccess and **AWSGlueConsoleFullAccess**

10. On the **Summary** page, verify that there are **five policies** attached to the role.

Attach policies		Add inline policy
Policy name	Policy type	X
AmazonS3FullAccess	AWS managed policy	X
AWSGlueServiceRole	AWS managed policy	X
AWSGlueConsoleFullAccess	AWS managed policy	X
DatalakeDataAccess	Inline policy	X
DatalakePassRole	Inline policy	X

Create Glue JDBC connection for RDS

1. On the AWS Glue menu, select **Connections**.

The screenshot shows the AWS Glue Connections page. On the left, there's a sidebar with options like Data catalog, Databases, Tables, Connections (which is selected and highlighted in orange), Crawlers, Classifiers, and Settings. The main area has a heading "Connections" with a sub-instruction: "A connection contains the properties needed to connect to your data." Below this are buttons for "Add connection", "Test connection", and "Action". A table lists connections, but it currently shows "Showing: 0 - 0" and "You don't have any connections yet." There's a prominent blue "Add connection" button at the bottom of the table area.

2. Click **Add Connection**.

3. Enter the connection name. This name should be descriptive and easily recognized (e.g., "glue-rds-connection").
4. Choose **RDS** for connection type and **PostgreSQL** for Database Engine
5. Optionally, enter the description. This should also be descriptive and easily recognized and Click **Next**.

The screenshot shows the "Add connection" wizard. The title bar says "Add connection". On the left, there's a sidebar with three tabs: "Connection properties" (selected and highlighted in orange), "Connection access", and "Review all steps". The main form area is titled "Set up your connection's properties." It includes fields for "Connection name" (set to "rds-connection"), "Connection type" (set to "Amazon RDS"), and "Database engine" (set to "PostgreSQL"). There are checkboxes for "Require SSL connection" and "Description (optional)" (containing "Glue Connection to RDS Instance"). At the bottom right is a blue "Next" button.

6. Choose **dmslabinstance** as **Instance** and enter password –"master123" and Click **Next** and Click **Finish**.

Add connection

Connection properties
rds-connection
Type: PostgreSQL

Connection access

Review all steps

Set up access to your data store.

For more information, see [Working with Connections](#).

Instance: dmslabinstance

Database name: sportstickets

Username: master

Password:

[Back](#) [Next](#)

7. **glue-rds-connection** was created successfully. To test it, select the connection, and choose **Test connection**.

AWS Glue

Data catalog

Databases

Tables

Connections

Connections A connection contains the properties needed to connect to your data.

[Add connection](#) [Test connection](#) [Action ▾](#)

Name

glue-rds-connection

8. To Test connection, choose the IAM role created in the previous step and then click on **Test Connection**.

Test connection

Test connection from your VPC and subnet to data stores and Amazon S3.

IAM role [?](#)

LakeFormationWorkflowRole

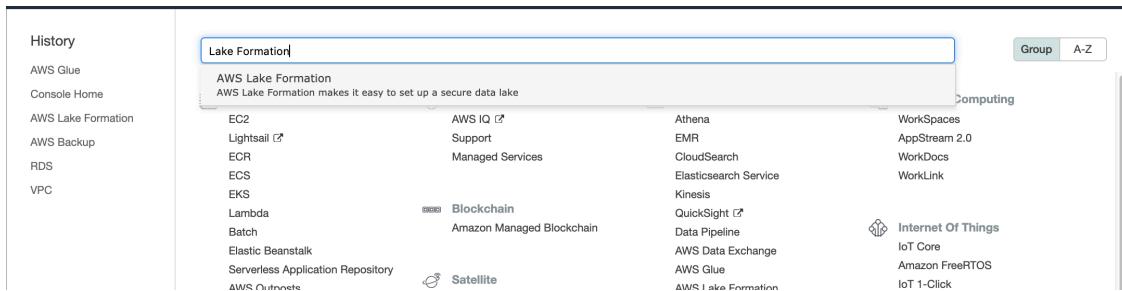
dmslab-student-GlueLabRole-Y0AJBNCP66ZI

LakeFormationWorkflowRole

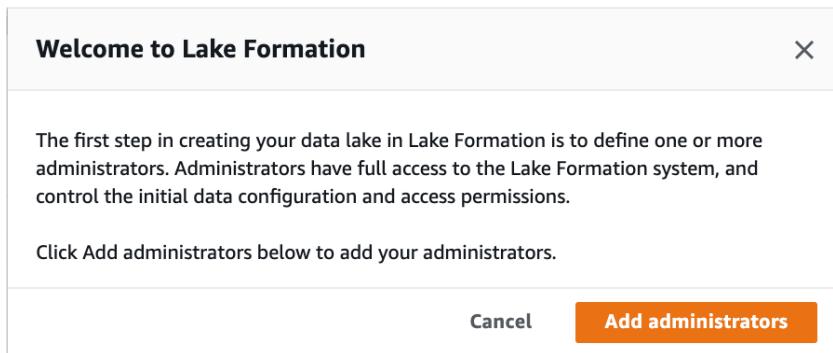
[Test connection](#)

Lake Formation – Add Administrator and start workflows using Blueprints.

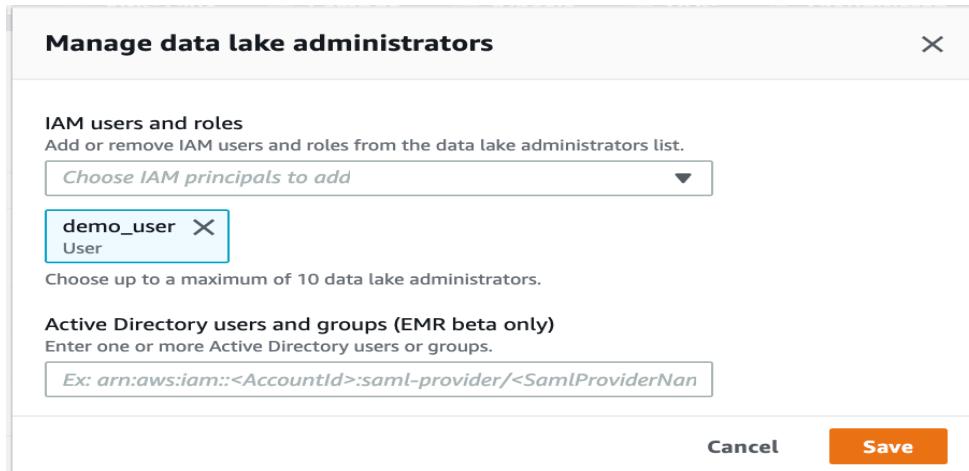
Navigate to the AWS Lake Formation service.



1. If you are logging into the lake formation console for the first time then you must add administrators first in order to do that follow Steps 2 and 3. Else skip to Step 4.
2. Click **Add administrators**



3. Add your <login user> as the Lake Formation Administrator and Click **Save**



4. Navigate to Databases on left pane. Select “ticketdata” and click on “Actions”, select “Grant” to grant permissions.

The screenshot shows the AWS Lake Formation interface. On the left, there's a sidebar with options like Dashboard, Data catalog, Tables, Settings, Register and ingest, Blueprints, Crawlers, and Jobs. Under Data catalog, 'Databases' is selected. The main area shows a table for 'Databases (0/1)'. One database, 'ticketdata', is listed with its name and an empty 'Amazon S3 path'. To the right of the table is a 'Actions' dropdown menu with options: Database, Delete, Edit, Permissions, Grant, Revoke, Verify permissions, and View permissions. The 'Grant' option is currently selected and highlighted with a blue border.

5. Under “IAM Users and Roles”, select Lake Formation role that you created – “LakeFormationWorkflowRole” and for user, select your username. Grant “super” permissions for Database permissions and Grantable permissions.

This screenshot shows the 'Grant permissions ticketdata' dialog. It has sections for 'IAM users and roles' (with 'LakeFormationWorkflowRole' selected), 'Active Directory users and groups (EMR beta only)', 'Database permissions' (with 'Super' checked), and 'Grantable permissions' (with 'Super' checked). At the bottom are 'Cancel' and 'Grant' buttons.

6. Add database Location by Edit the existing database **ticketdata**

This screenshot shows the 'Databases' page again. The 'ticketdata' database is selected. The 'Actions' menu is open, and the 'Edit' option is highlighted with a blue border.

7. Clear the checkbox **Use only IAM access control**. Changing the default security setting so that access to Data Catalog resources (databases and tables) is managed by Lake Formation permissions.

AWS Lake Formation > Databases > ticketdata > Edit database

Edit database

Database details

Name
ticketdata

Location - *optional*
Choose an Amazon S3 path for this database, which eliminates the need to grant data location permissions on catalog table paths that are this location's children
s3://mod-3fccddd609114925-dmslabs3bucket-4f4ndmet5tmw

Description - *optional*
Enter a description

Default permissions for newly created tables
This setting maintains existing AWS Glue data catalog behavior. You can still set individual permissions, 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 tables in this database

8. On the left pane navigate to **Blueprints** click **Use blueprints**.

AWS Lake Formation X

AWS Lake Formation > Blueprints

Blueprint overview
Blueprints enable data ingestion from common sources using automated workflows.

Database blueprints
Ingest data from MySQL, PostgreSQL, Oracle, and SQL server databases to your data lake, either as bulk load snapshot, or incrementally load new data over time.

Log file blueprints
Ingest data from popular log file formats from AWS CloudTrail, Classic Load Balancer, and Application Load Balancer logs

Workflows
Workflows are instances of ingestion blueprints in Lake Formation.

Actions

Filter workflows

Name	Created on	Last updated	Last run status
No available workflows			
<input type="button" value="Use blueprint"/>			

- For Blueprint Type, select **Database snapshot**
- Under Import Source

- i. For **Database Connection** choose the DB connection created in the glue. [Ex: "glue-rds-connection"]
- ii. For **Source Data Path** enter **sportstickets/dms_sample/player**

AWS Lake Formation > Blueprints > Use a blueprint

Use a blueprint

Blueprint type
Configure a blueprint to create a workflow.

- Database snapshot
Bulk load data to your data lake from MySQL, PostgreSQL, Oracle, and Microsoft SQL Server databases.
- Incremental database
Load new data to your data lake from MySQL, PostgreSQL, Oracle, and SQL Server databases.
- AWS CloudTrail
Bulk load data from AWS CloudTrail sources.
- Classic Load Balancer logs
Load data from Classic Load Balancer logs.
- Application Load Balancer logs
Load data from Application Load Balancer logs.

Import source
Configure the workflow source.

Database connection
Choose the connection to the data source. [Create a connection in AWS Glue](#)

glue-rds-connection

Source data path
Enter the path from which to ingest data. For JDBC databases with schema support, enter database/schema/table. Substitute the percent (%) wildcard for schema or table.

sportstickets/dms_sample/player

- c. Under Import Target
 - i. For **Target Database**, choose existing "ticketdata".
 - ii. For **Target storage location** browse and select the dmslab S3 bucket created in the previous lab.

Choose an Amazon S3 location in region us-east-1

< S3

- aws-athena-query-results-us-east-1-861525167008 >
- aws-glue-scripts-861525167008-us-east-1 >
- aws-glue-temporary-861525167008-us-east-1 >
- cf-templates-1am9ivtpy9915-us-east-1 >
- kinesis-pre-lab-processeds3bucket-1rjdj6en5pjxa >
- kinesis-pre-lab-raws3bucket-r8zx4qoouthk >
- lf-data-lake-861525167008 >
- lf-workshop-861525167008 >
- mod-3fccddd609114925-dmslabs3bucket-4f4ndmet5trw >
- mod-3fccddd609114925-s3bucketworkgroupa-1m6lh4qussvia >
- mod-3fccddd609114925-s3bucketworkgroupb-10lkurw7b6mu >

Cancel Select

- iii. Add a folder at the end of the bucket url path.
NOTE: The value is similar to the following string
“dmslab-student-dmslabs3bucket-q29vwhf8n41w/**lakeformation**”
- iv. For Data Format choose Parquet

Import target
Configure the target of the workflow.

Target database
Choose a database in the Data Catalog. [Create database](#)

Target storage location
Choose a data lake location or other Amazon S3 path.

Data format
Choose the output data format.

- d. For Import Frequency, Select Run On Demand
- e. For Import Options;
 - i. Give a Workflow Name **RDS-S3-Glue-Workflow**
 - ii. For the IAM role choose the **LakeFormationWorkflowRole** created previously
 - iii. For Table prefix type **lakeformation**

Import options
Configure the workflow.

Workflow name

Names may contain letters (A-Z), numbers (0-9), hyphens (-), or underscores (_), and must be less than 256 characters long.

IAM role

Table prefix
The table prefix that is used for catalog tables that are created.

Table prefixes may contain lower case letters (a-z), numbers (0-9), hyphens (-), or underscores (_).

Maximum capacity - optional
Sets the number of data processing units (DPUs) that can be allocated when this job runs. A DPU is a relative measure of processing power that consists of 4 vCPUs of compute capacity and 16 GB of memory.

Concurrency - optional
Sets the maximum number of concurrent runs that are allowed for this job. An error is returned when this threshold is reached. The default is 5.

9. Leave other options as default, Choose **Create**, and wait for the console to report that the workflow was successfully created.
10. Once the blueprint gets created, click on **Start it Now?** There may be a delay of 5-10s delay in the blueprint showing up. You may have to **hit refresh**. Select the blueprint and choose **Start** in **Actions drop down**
11. Once the workflow starts executing, you will see the status changes from running → discovering → Completed

The screenshot shows the 'Workflows (0/1)' section of the AWS Lake Formation console. It displays a single workflow entry:

Name	Created on	Last updated	Last run status
RDS-S3-Glue-Workflow	Mon, Mar 23, 2020, 10:50 PM UTC	Mon, Mar 23, 2020, 10:50 PM UTC	Discovering

Explore the Underlying Components of a Blueprint

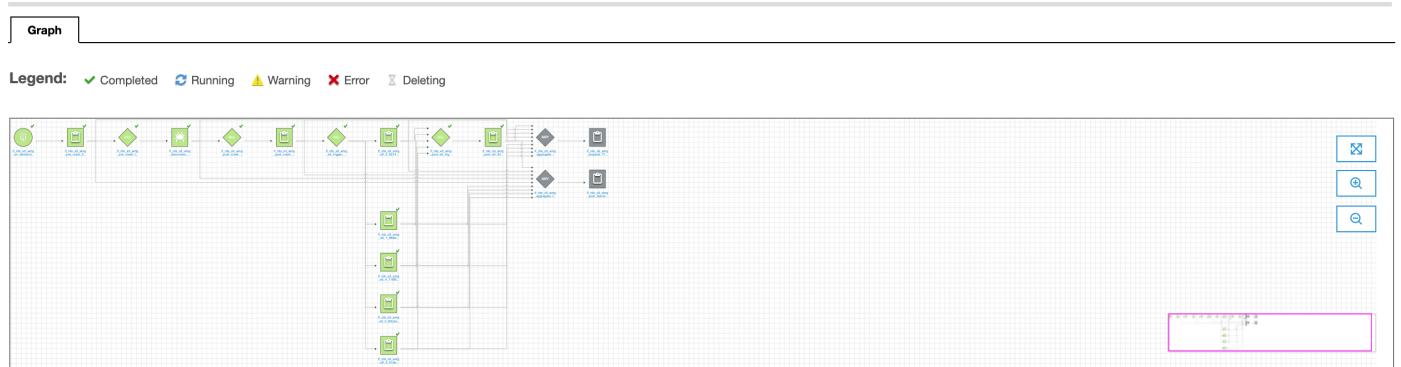
The Lake Formation blueprint creates a Glue Workflow under the hood which contains Glue ETL jobs – both python shell and pyspark; Glue crawlers and triggers. It will take somewhere between 15-20 mins to finish execution. In the meantime, let us drill down to see what it creates for us;

1. On the **Lake Formation console**, in the navigation pane, choose **Blueprints**
2. In the **Workflow section**, click on the **Workflow name**. This will direct you to the Workflow run page. Click on the **Run Id**.

The screenshot shows the 'Workflow runs (1)' section of the AWS Lake Formation console. It displays a single run entry:

Name	Started on	Run ID
lf_rds_s3_amg	Mon, Dec 16, 2019, 6:07 AM UTC	wr_bcf4efeb081293bb613e63c2a586ee37f9ccbdd1ca97dd953d2851 4a0951908f [2]

3. On the Glue Console page, click on **Get Started** and head to **ETL Workflows** in the navigation pane.
4. Here you can see the graphical representation of the Glue workflow built by Lake Formation blueprint. Highlighting and clicking on individual components will display the details of those components (name, description, job run id, start time, execution time)
5. To understand what all Glue Jobs got created as a part of this workflow, in the navigation pane, click on **Jobs**.
12. Every job comes with history, details, script and metrics tab. Review each of these tabs for any of the python shell or pyspark jobs.



Explore workflow results in Athena

1. Navigate to the Glue console

This screenshot shows the AWS Services console with the search bar set to "glue". The "AWS Glue" service is highlighted, showing its description as "AWS Glue is a fully managed ETL (extract, transform, and load) service". Below the search bar, there are links for EC2, Database Migration Service, S3, CloudWatch, and AWS Glue. A link to "All services" is also visible.

2. Navigate to **Databases** on the left panel and select **ticketdata**

3. Click on “Tables in ticketdata” and this table will be pre fixed by “lakeformation_”

Example: lakeformation__sportstickets_dms_sample_player

This screenshot shows the AWS Glue Data catalog. The left sidebar has "Tables" selected under "Databases". The main pane displays a table of tables in the "ticketdata" database. The table has columns for "Name" and "Database". There are four entries: "_lakeformation__sportstickets_dms_sample_player", "_temp_lakeformation__sportstickets_dms_sample_player", and "lakeformation__sportstickets_dms_sample_player" (which is highlighted in blue). All three entries belong to the "ticketdata" database.

4. And Click Action -> View Data

This screenshot shows the AWS Glue Data catalog with the "Tables" section for the "ticketdata" database. The left sidebar has "Tables" selected under "Databases". The main pane shows the same table of tables. The "Action" dropdown menu is open over the row for "lakeformation__sportstickets_dms_sample_player". The "View data" option is highlighted in blue, indicating it is the selected action.

5. This will now take you to **Athena** console, where you can preview the table contents, as show below;

The screenshot shows the AWS Athena Query Editor interface. On the left, there's a sidebar with 'Data source' set to 'AwsDataCatalog' and 'Database' set to 'ticketdata'. Below these are sections for 'Tables (22)' and 'Views (2)'. In the main area, a query is running: 'SELECT * FROM "ticketdata"."lakeformation_sportstickets_dms_sample_player" limit 10;'. A context menu is open over the table name in the query, with options like 'Preview table', 'Show properties', 'Delete table', and 'Generate Create Table DDL'. Below the query editor, the results section displays the table structure and 10 rows of data.

	full_name	sport_team_id	last_name	id	first_name
1	Loewen	131.0	Adam Loewen	1.0	Adam
2	Pollock	131.0	A.J. Pollock	11.0	A.J.
3	Sanabia	131.0	Alex Sanabia	21.0	Alex
4	Chaffin	131.0	Andrew Chaffin	31.0	Andrew
5	Marte	131.0	Andy Marte	41.0	Andy
6	Bradley	131.0	Archie Bradley	51.0	Archie
7	Francisco	131.0	Ben Francisco	61.0	Ben
8	Shipley	131.0	Braden Shipley	71.0	Braden
9	Hagens	131.0	Bradin Hagens	81.0	Bradin
10	Drury	131.0	Brandon Drury	91.0	Brandon

Congratulation!!! You have completed lake formation lab. To explore more fine grain data lake security feature, continue to next section.

[Optional] Grant fine grain access controls to Data Lake user

Before we start the querying the data, let us create an IAM User **datalake_user** and grant column level access on the table created by the Lake formation workflow above, to **datalake_user**.

1. Login as admin user to your account. Navigate to **IAM Console** and click on **Add User**.

The screenshot shows the AWS IAM 'Add user' page. At the top, there are 'Services' and 'Resource Groups' navigation items. The main area has a heading 'Identity and Access Management (IAM)'. On the left, there's a sidebar with 'Dashboard', 'Access management', 'Groups', and 'Users' (which is highlighted). In the main content area, there's a search bar with 'Find users by username or' and a dropdown for 'User name' containing 'demo_user'. A large blue 'Add user' button is prominently displayed.

2. Create a user named **datalake_user** and give it a password: **master123**.

Add user

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type* **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password* Autogenerated password Custom password

.....
 Show password

Require password reset User must create a new password at next sign-in
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

* Required [Cancel](#) [Next: Permissions](#)

3. Next click on **Permissions**

4. Choose **Attach existing policies directly** and search for **AthenaFullAccess**

Add user

Set permissions

[Add user to group](#) [Copy permissions from existing user](#) [Attach existing policies directly](#)

[Create policy](#)

Filter policies		Showing 2 results	
	Policy name	Type	Used as
<input checked="" type="checkbox"/>	AmazonAthenaFullAccess	AWS managed	Permissions policy (3)
<input type="checkbox"/>	AWSQuicksightAthenaAccess	AWS managed	Permissions policy (2)

5. Keep navigating to the next steps until reached the end. Review the details and click on “Create User”.

6. Click on the **datalake_user** user, and **add inline policy**

[Add user](#) [Delete user](#)

Showing 4 results						
	User name	Groups	Access key age	Password age	Last activity	MFA
<input checked="" type="checkbox"/>	datalake_user	None	None	Today	None	Not enabled
<input type="checkbox"/>	EC2Console	None	None	None	Not enabled	

User ARN: arn:aws:iam::861525167008:user/datalake_user

Path: /

Creation time: 2020-04-09 17:27 UTC+1000

Permissions **Groups** **Tags** **Security credentials** **Access Advisor**

▼ Permissions policies (1 policy applied)

Add permissions **Add inline policy**

Policy name	Policy type
AmazonAthenaFullAccess	AWS managed policy

Attached directly

▶ Permissions boundary (not set)

Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor

JSON

[Import managed policy](#)

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "s3:Put*",
8         "s3:Get*",
9         "s3>List"
10      ],
11      "Resource": ["arn:aws:s3:::mod-3fccddd609114925-dms1abs3bucket-4f4ndmet5tmw/athena-results/*"]
12    }
13  ]
14 }
```

Character count: 193 of 2,048.

The current character count includes character for all inline policies in the user: datalake_user.

Cancel

Review policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Put*",
        "s3:Get*",
        "s3>List"
      ],
      "Resource": [
        "arn:aws:s3:::<your_dmslabs3bucket_unique_name>/athena-results/*"
      ]
    }
  ]
}
```

7. Give a name **athena_3access** to the policy, then **Create Policy**
8. Navigate to the **Lake Formation console**, in the navigation pane, under **Permissions**, choose **Data permissions**.

The screenshot shows the AWS Lake Formation interface. In the left sidebar, under the 'Permissions' section, 'Data permissions' is highlighted with an orange box. The main content area shows a table titled 'Data permissions (80)' with a note: 'Choose a database or table for which to review, grant or revoke user permissions.' A search bar and a pagination area (1-4) are at the bottom of the table. On the right side of the table, there are three buttons: 'Grant' (highlighted with an orange box), 'Revoke', and 'Cancel'.

9. Choose **Grant**, and in the **Grant permissions** dialog box, do the following:
 - a. For **IAM user and roles**, choose **datalake_user**.
 - b. For **Database**, choose **ticketdata**
 - c. The **Table** list populates.
 - d. For **Table**, choose the table shown.
 - e. For **Columns**, select **Include Columns** and choose **id, first_name**
 - f. For **Table permissions**, choose **Select**.

10. Choose **Grant**.

The screenshot shows the 'Grant permissions' dialog box for the database 'gov_sporttickets_dms_sample_player'. The dialog has several sections:

- IAM users and roles:** Shows 'data_lake_user' selected in a dropdown.
- Active Directory users and groups (EMR beta only):** An empty input field.
- Column - optional:** Shows 'Include columns' selected.
- Include columns:** Shows 'id double' and 'first_name string' selected.
- Table permissions:** Shows 'Select' checked.
- Super:** A note explaining Super permission.
- Grantable permissions:** An empty section.
- Super:** A note explaining Super permission.
- At the bottom are 'Cancel' and 'Grant' buttons, with 'Grant' highlighted with an orange box.

[Optional] Verify data permissions using Athena

Using Athena, let us now explore the data set as the **datalake_user**.

1. Sign out and Sign back in to your AWS account with the same **account ID**, but with a different username - **datalake_user**

The screenshot shows two side-by-side browser windows. On the left is the AWS Lake Formation console under the 'Permissions' section for a database named 'If-db'. A 'Sign Out' button is highlighted with a red box. On the right is the AWS 'Sign in as IAM user' page, where the 'Account ID' field contains '861525167008' and the 'IAM user name' field contains 'datalake_user'. The 'Sign In' button is at the bottom.

2. Navigate to **Amazon Athena** console.
3. Before run your first Athena query, you need to **set up a query result location** in Amazon S3. On the right side of the Athena console, click on **Settings** and type in the S3 bucket location. The entry should look something like: `s3://<your_dmslabs3bucket_unique_name>/athena-results/`

The screenshot shows the 'Query Editor' tab of the Athena console. A message at the top says, 'Before you run your first query, you need to set up a query result location in Amazon S3. Learn more'. Below it, the 'Settings' dialog is open. The 'Query result location' field is filled with 's3://xxxx-dmslabs3bucket-xxxx/athena-results/'. The 'Workgroup' is set to 'primary'. At the bottom right are 'Cancel' and 'Save' buttons.

4. Next, ensure database **ticketdata** database is selected.
5. Now run a **Select** query on the table within the **ticketdata** database. There should be only one table in there, which was created by Lake Formation workflow.
6. You will see that the **datalake_user** can **only see the columns id, first_name** in the select query result. The **datalake_user** cannot see **last_name, sports_team_id, full_name** columns in the table. This is because, **datalake_admin** gave **datalake_user** permissions to only select from table for the **id** and **first_name** columns.

The screenshot shows the AWS Athena Query Editor interface. On the left, the sidebar displays the Data source (awsdatacatalog), Database (ticketdata), and Tables (0). The main area shows a query editor with three tabs: New query 1, New query 2, and New query 3 (selected). The query text is:

```
1 | SELECT * FROM "ticketdata"."lakeformation_sportstickets_dms_sample_player" limit 10;
```

Below the query, the status bar indicates: (Run time: 2.06 seconds, Data scanned: 43.69 KB). The results section displays a table with the following data:

	id	first_name
1	1.0	Adam
2	11.0	A.J.
3	21.0	Alex
4	31.0	Andrew
5	41.0	Andy
6	51.0	Archie

An orange oval highlights the column headers and the first few rows of the results table.