Amazon Web Services
Data Engineering Immersion Day

Lake Formation Lab

March 2020

Table of Contents

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:

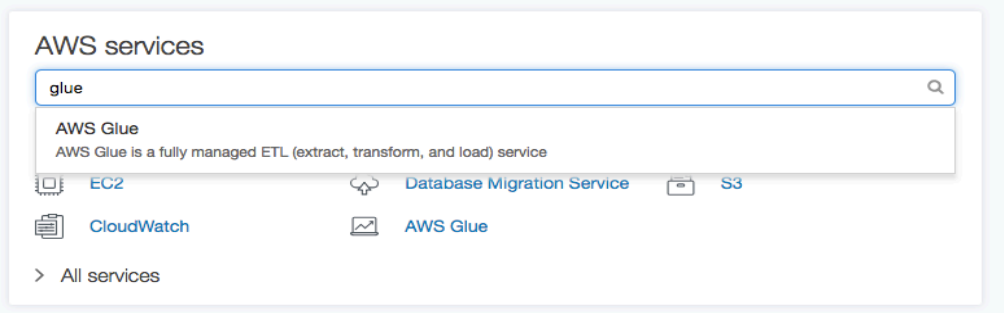
1. Create a JDBC connection to RDS in AWS Glue
2. Lake Formation IAM Role

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.

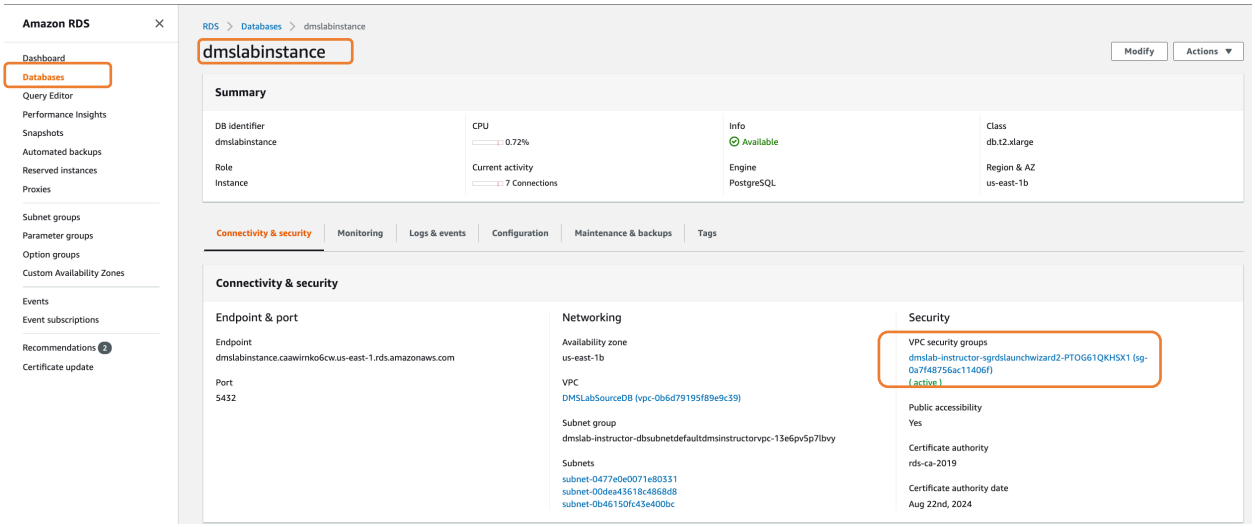


Setup Network Configuration

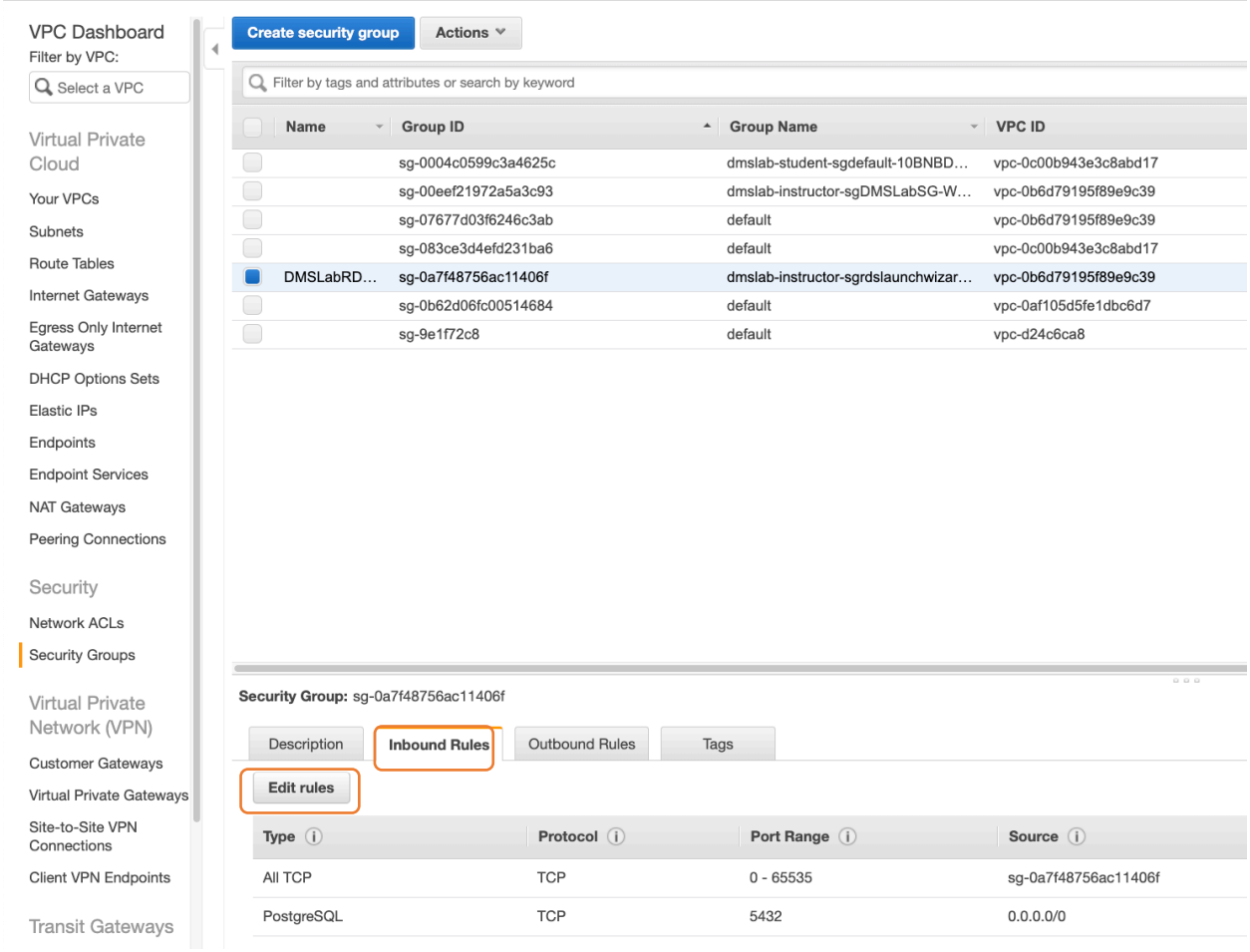
To begin with, we need to perform network configuration. We will open ports specific to RDS PostgreSQL DB, define inbound rules and create S3 endpoint to access data in S3 bucket (created as part of DMS Lab)

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** and you will be navigated to VPC Dashboard.

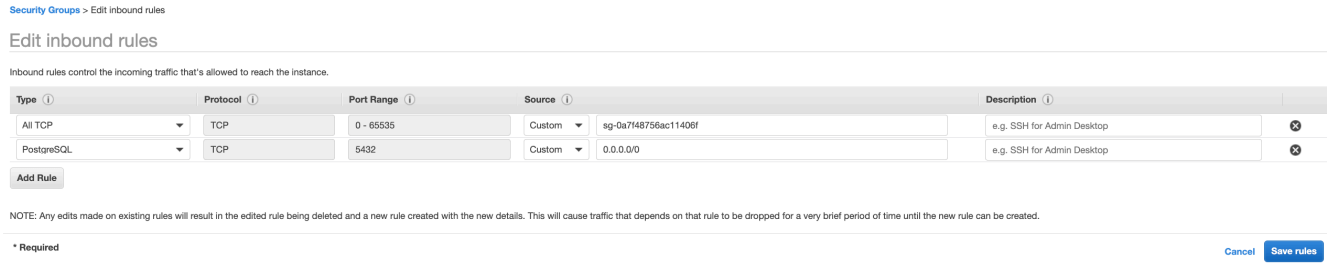
   Note: Make sure your RDS instance is **available**. If the Status says “Storage Full”, increase the DB storage capacity and apply changes immediately to the DB instance and proceed further.



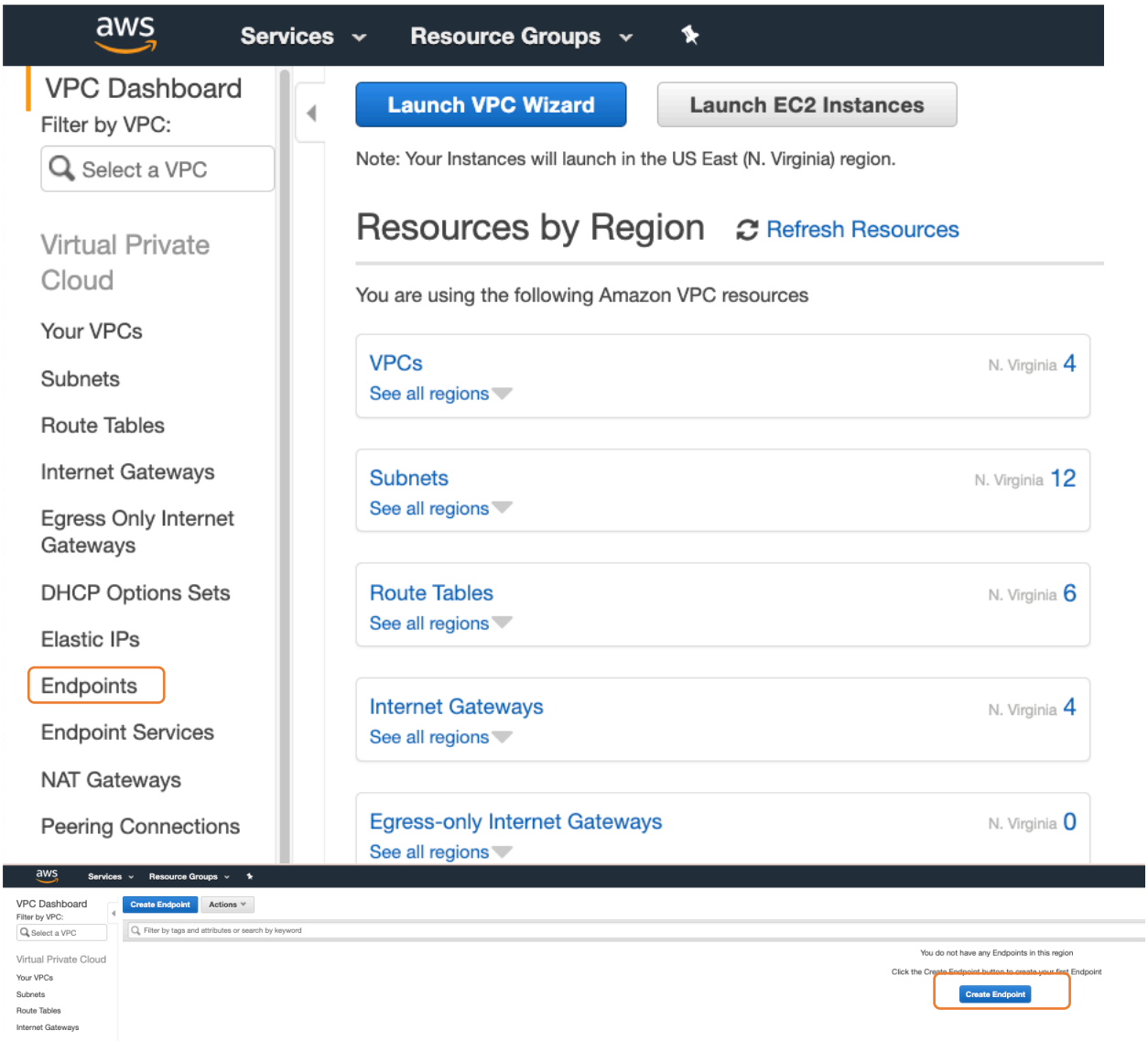
2. Security group associated with RDS instance will be automatically selected. Select Inbound Rules and Click **Edit Rules.**



3. Create Inbound Rules as shown below:
    a. All TCP with self-reference of security group as its own source.
    b. PostgreSQL on 5432 from 0.0.0.0/0
    c. Click on **Save Rules**



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

search : s3 | Add filter | 1 to 1 of 1

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

VPC* vpc-0b6d79195f89e9c39

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

	Route Table ID	Main	Associated With
[x]	rtb-07cb8bc42e0314882	No	3 subnets
[ ]	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": "*"
    }
  ]
}
```

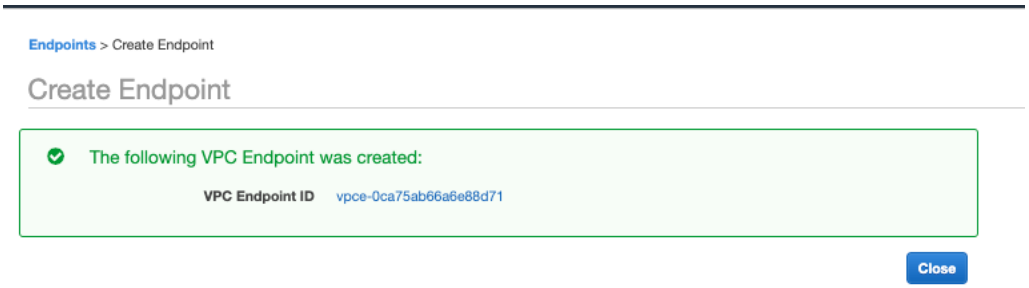
Key (128 characters maximum) **Value** (256 characters maximum)

This resource currently has no tags

Add Tag 50 remaining (Up to 50 tags maximum)

* Required

Cancel Create endpoint



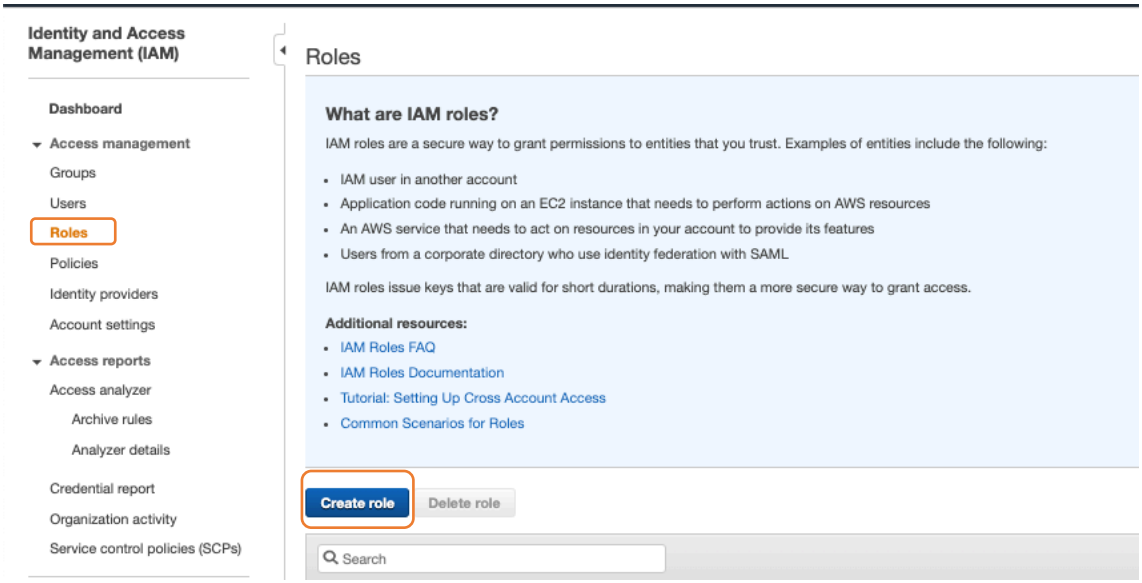
Endpoint creation is successful.

Create an IAM role to use with Lake Formation:

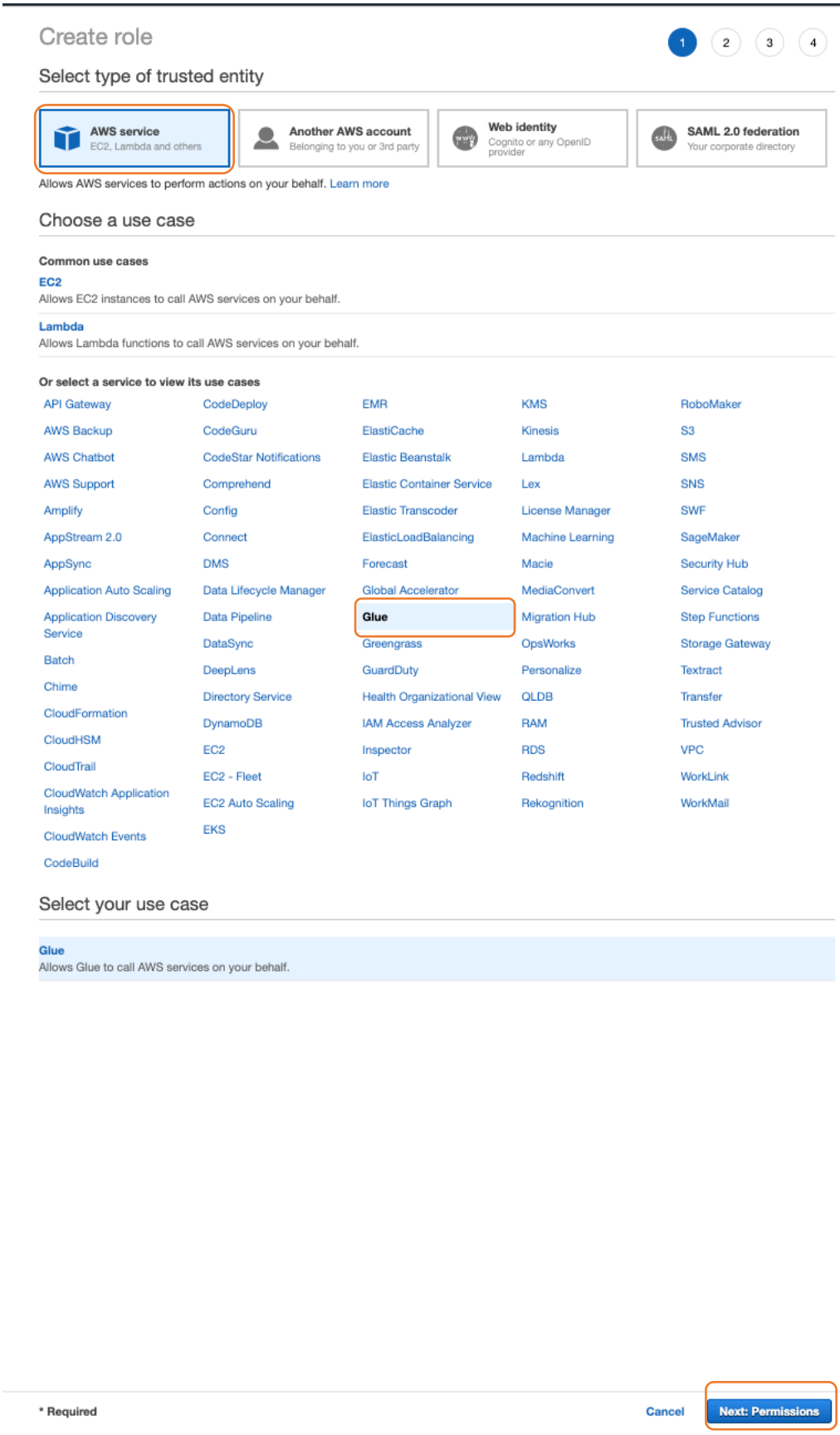
Documentation Reference: Link

Create the IAM Role by following below steps:

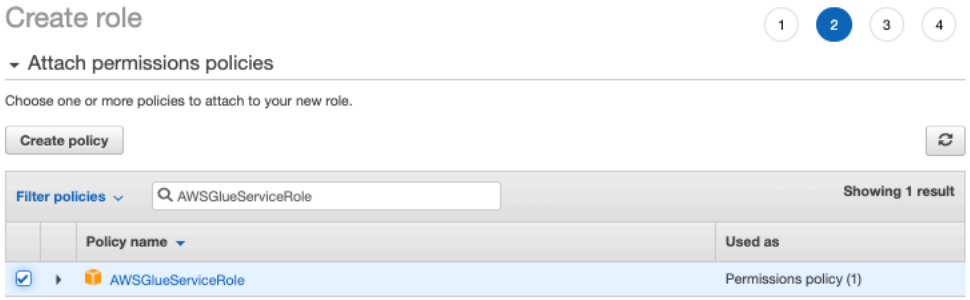
1. On the IAM console, In the navigation pane, choose **Roles**, and then choose **Create role**.



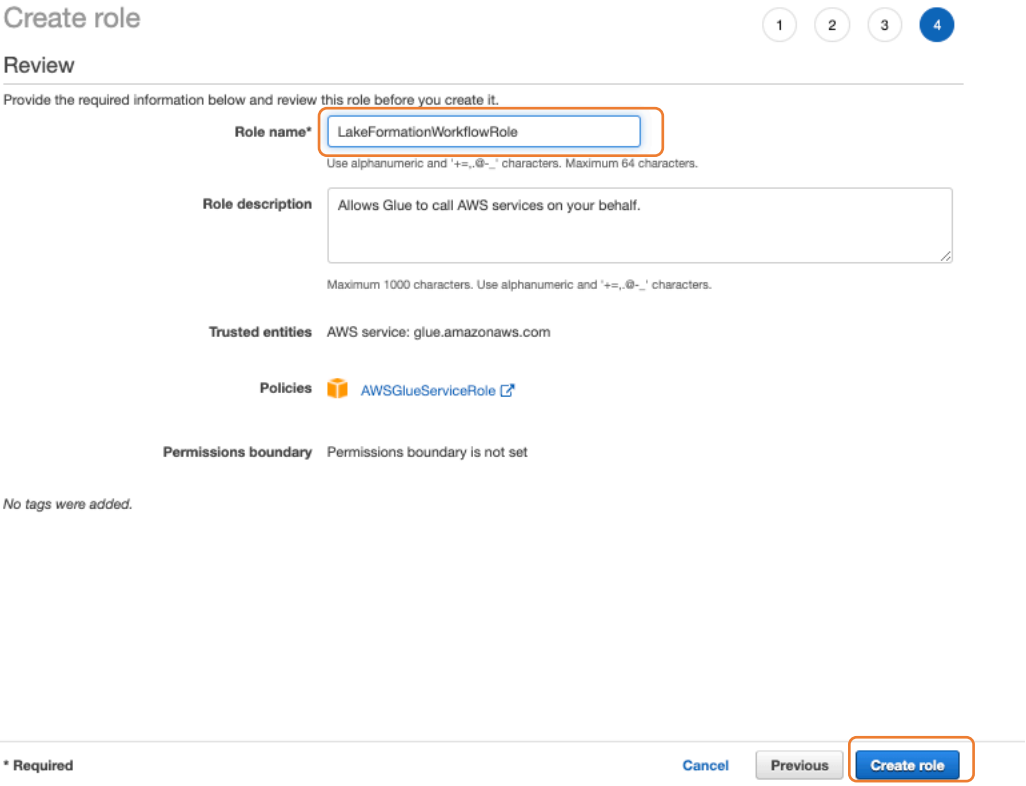
2. On the Create role page, choose **AWS service**, and choose **Glue**. Then choose **Next:Permissions**.


Create role
1
2
3
4
Select type of trusted entity
AWS service
EC2, Lambda and others
Another AWS account
Belonging to you or 3rd party
Web identity
Cognito or any OpenID provider
SAML 2.0 federation
Your corporate directory
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
AWS Backup
AWS Chatbot
AWS Support
Amplify
AppStream 2.0
AppSync
Application Auto Scaling
Application Discovery Service
Batch
Chime
CloudFormation
CloudHSM
CloudTrail
CloudWatch Application Insights
CloudWatch Events
CodeBuild
CodeDeploy
CodeGuru
CodeStar Notifications
Comprehend
Config
Connect
DMS
Data Lifecycle Manager
Data Pipeline
DataSync
DeepLens
Directory Service
DynamoDB
EC2
EC2 - Fleet
EC2 Auto Scaling
EKS
EMR
ElastiCache
Elastic Beanstalk
Elastic Container Service
Elastic Transcoder
ElasticLoadBalancing
Forecast
Global Accelerator
Glue
Greengrass
GuardDuty
Health Organizational View
IAM Access Analyzer
Inspector
IoT
IoT Things Graph
KMS
Kinesis
Lambda
Lex
License Manager
Machine Learning
Macie
MediaConvert
Migration Hub
OpsWorks
Personalize
QLDB
RAM
RDS
Redshift
Rekognition
RoboMaker
S3
SMS
SNS
SWF
SageMaker
Security Hub
Service Catalog
Step Functions
Storage Gateway
Textract
Transfer
Trusted Advisor
VPC
WorkLink
WorkMail
Select your use case
Glue
Allows Glue to call AWS services on your behalf.
* Required
Cancel
Next: Permissions

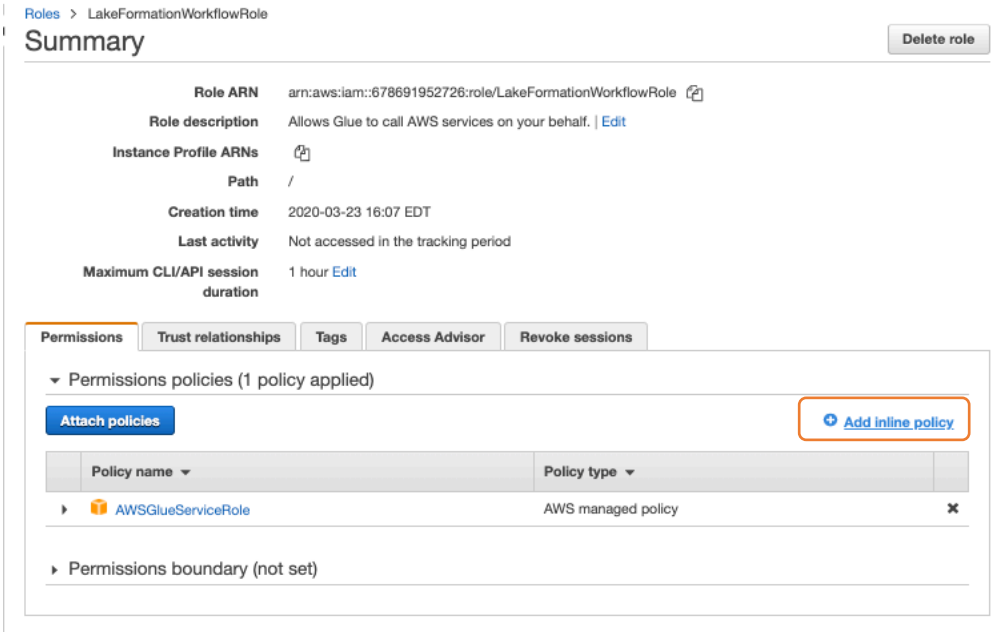
3. Search for and select the **AWSGlueServiceRole** managed policy. Optionally, provide tags.



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

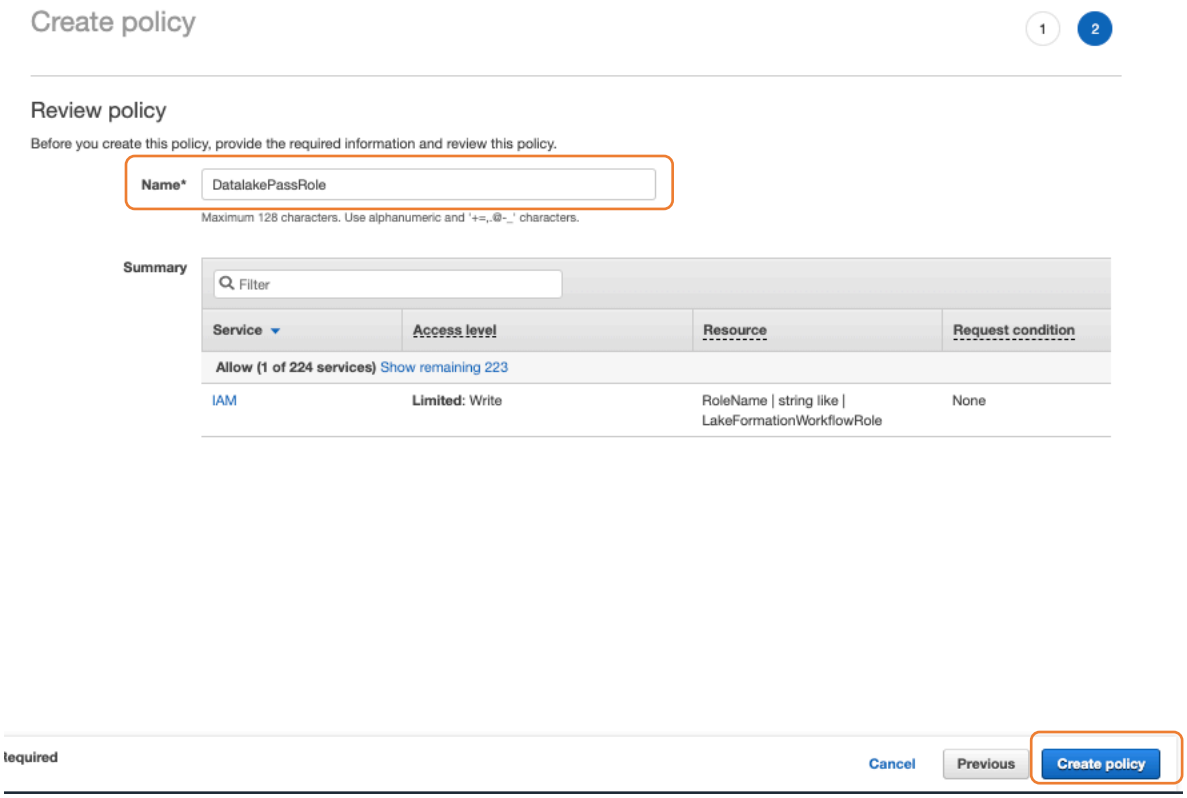


5. Back on the **Roles page**, search for and choose **LakeFormationWorkflowRole**.
6. On the role **Summary page**, under the **Permissions** tab, choose **Add inline policy**, and add the following policy in JSON editor.




```
{
   "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**



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

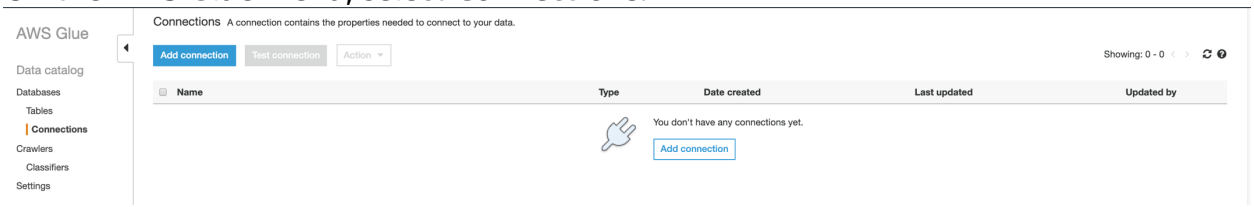
Important: In the following policy, replace account-id with your AWS account number. Account ID can be found in Account Summary page.

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

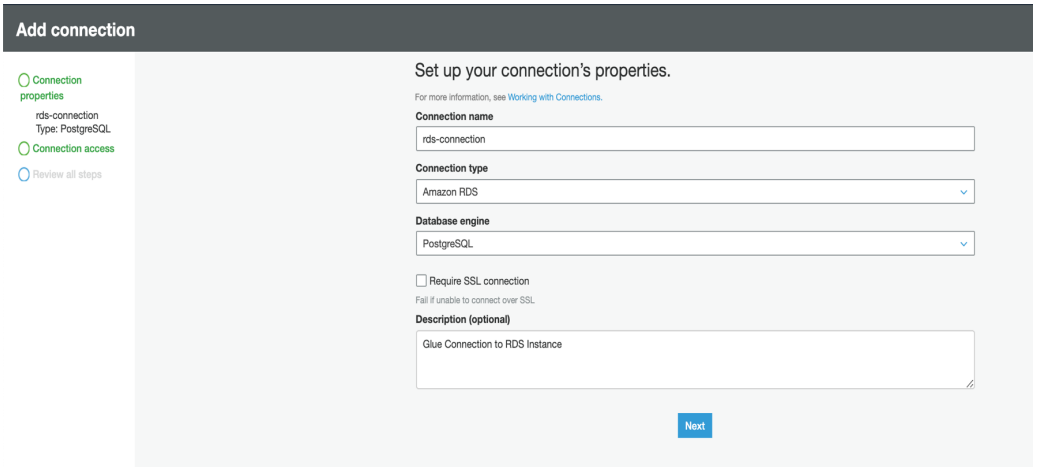
9. Back to policy page, add two more AWS service level permissions: **AmazonS3FullAccess** and **AWSGlueConsoleFullAccess**
10. On the **Summary** page, verify that there are **five policies** attached to the role.

Create Glue JDBC connection for RDS

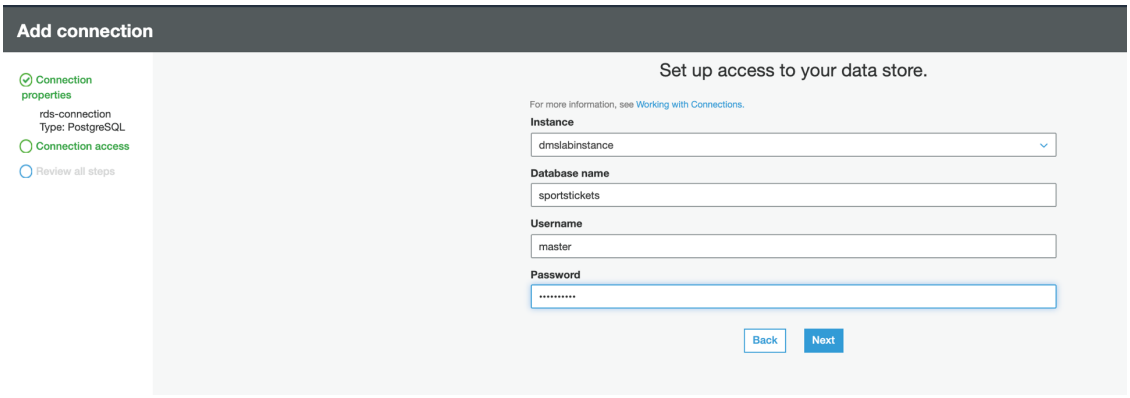
1. On the AWS Glue menu, select **Connections**.



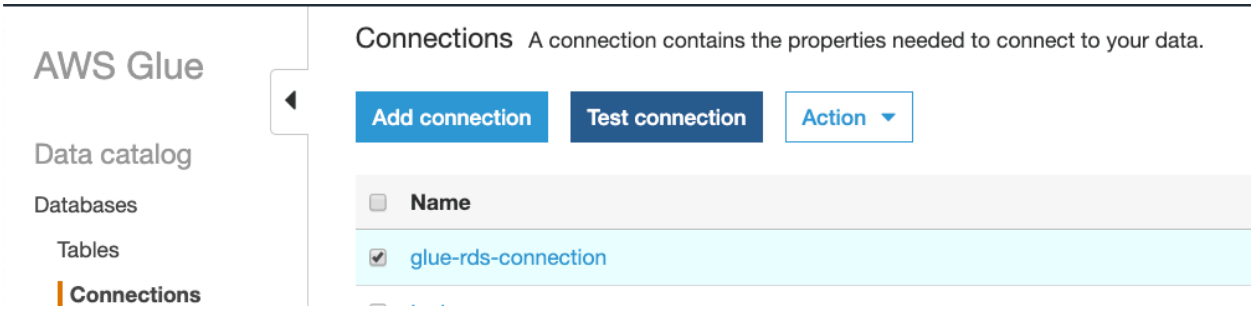
2. Click **Add Connection**.
3. Enter the connection name. This name should be descriptive and easily recognized (e.g ," glue-rds-connection").
4. Choose **RDS** for connection type and **PostgreSQL** for Database Engine
5. Optionally, enter the description. This should also be descriptive and easily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



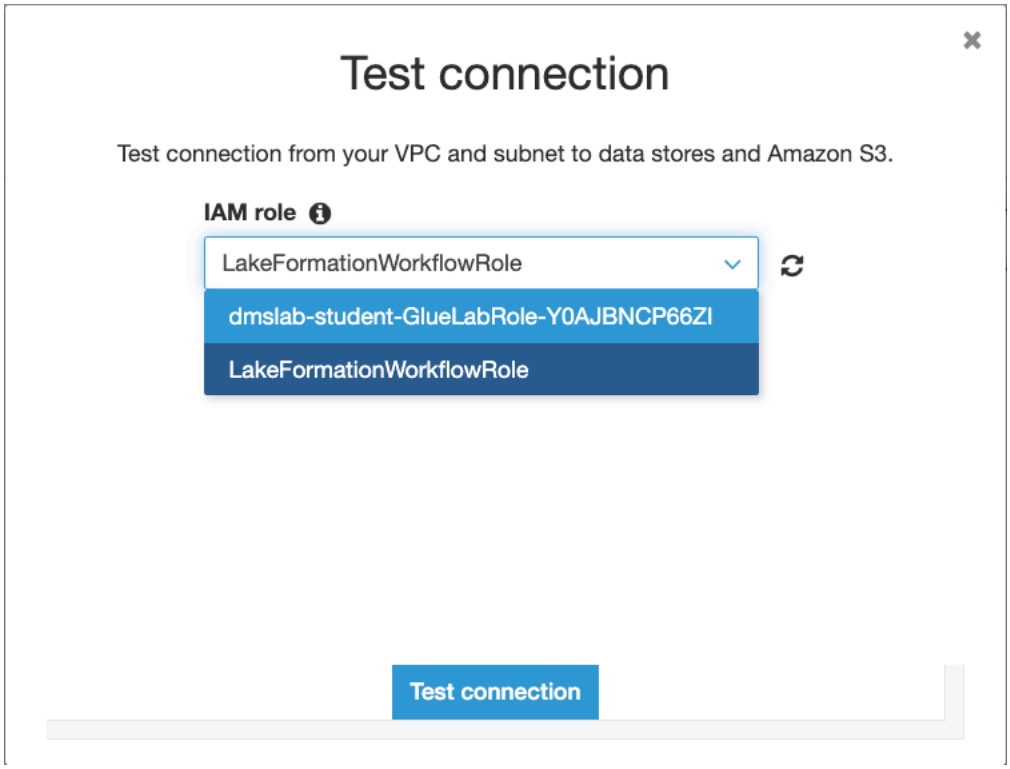
6. Choose **dmslabinstance** as **Instance** and enter password –"master123" **and** Click **Next and** Click **Finish.**



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

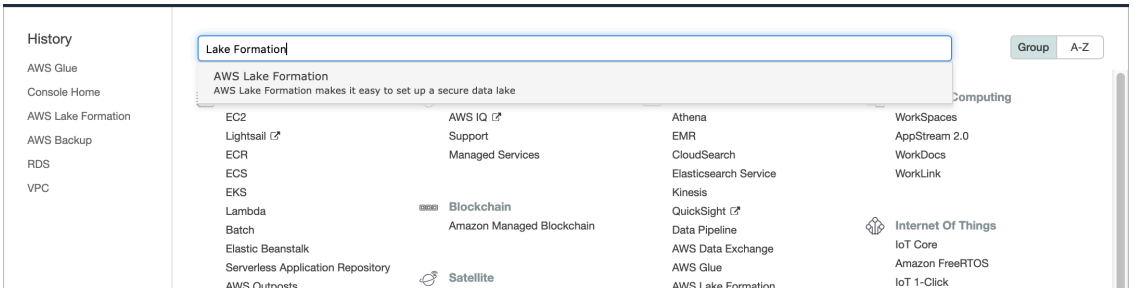


8. To Test connection, choose the IAM role created in the previous step and then click on **Test Connection.**

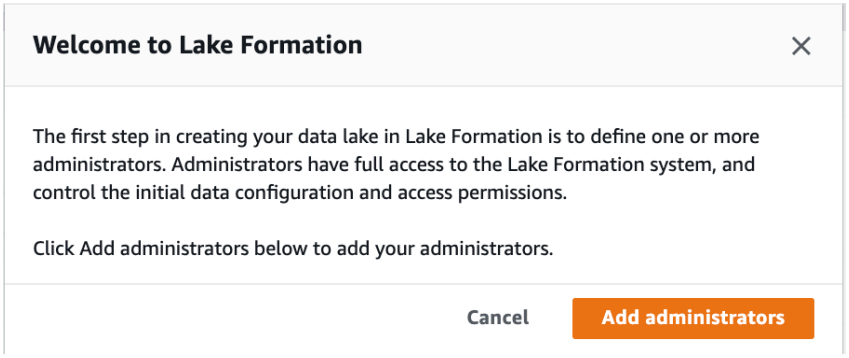


Lake Formation – Add Administrator and start workflows using Blueprints.

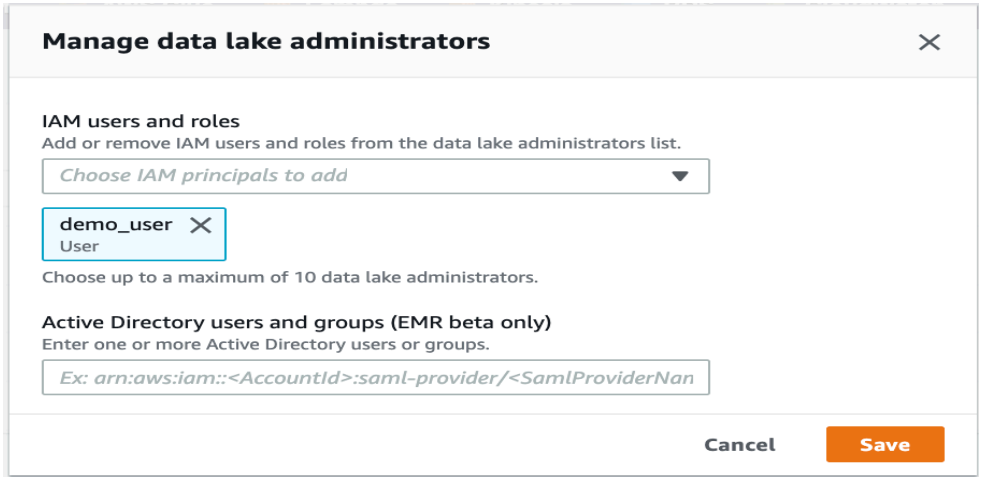
Navigate to the AWS Lake Formation service.



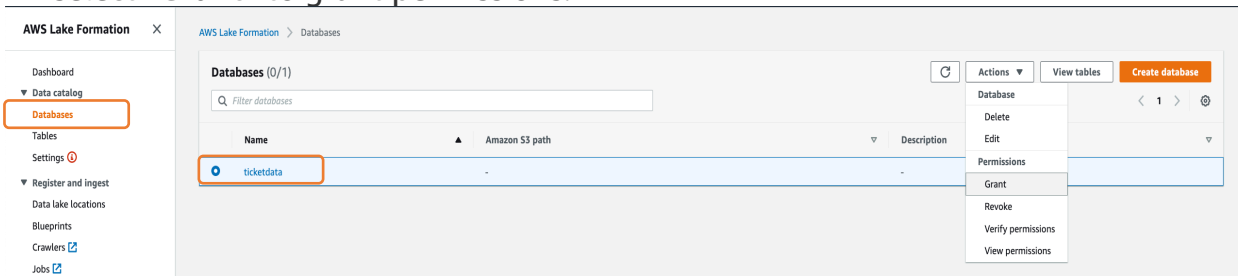
1. If you are logging into the lake formation console for the first time then you must add administrators first in order to do that follow Steps 2 and 3. Else skip to Step 4.
2. Click **Add administrators**



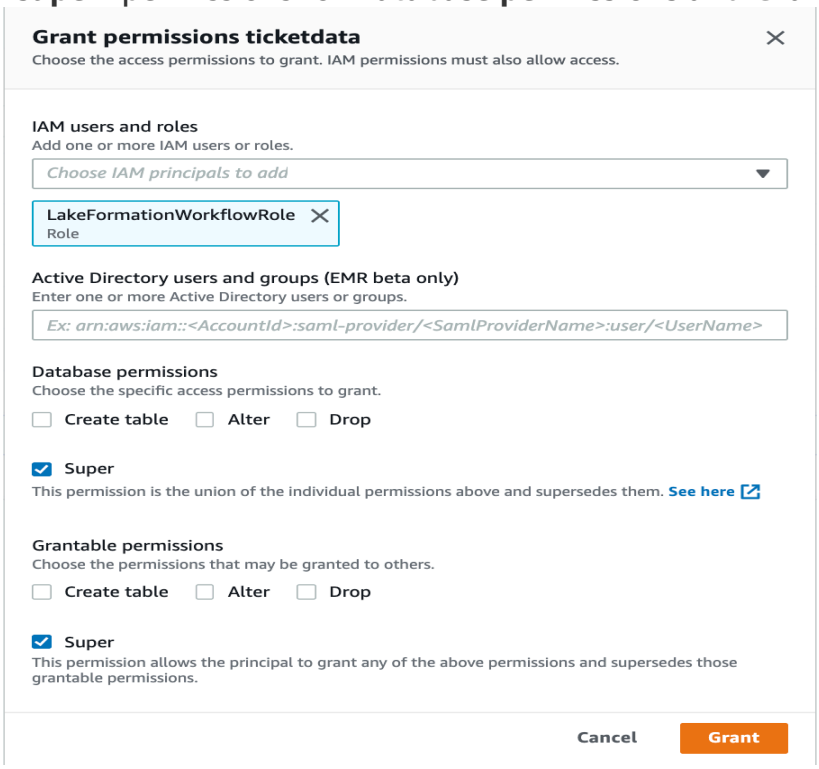
3. Add your <**login user**> as the Lake Formation Administrator and Click **Save**



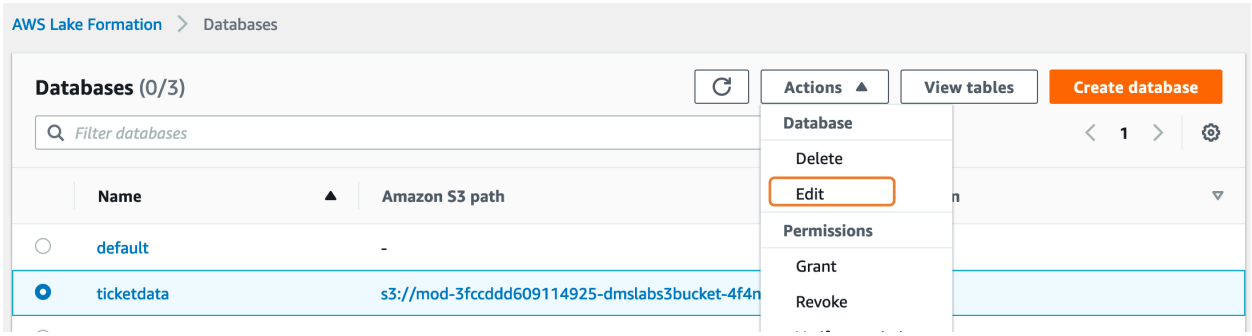
4. Navigate to **Databases** on left pane. Select "**ticketdata**" and click on "**Actions**", select "**Grant**" to grant permissions.



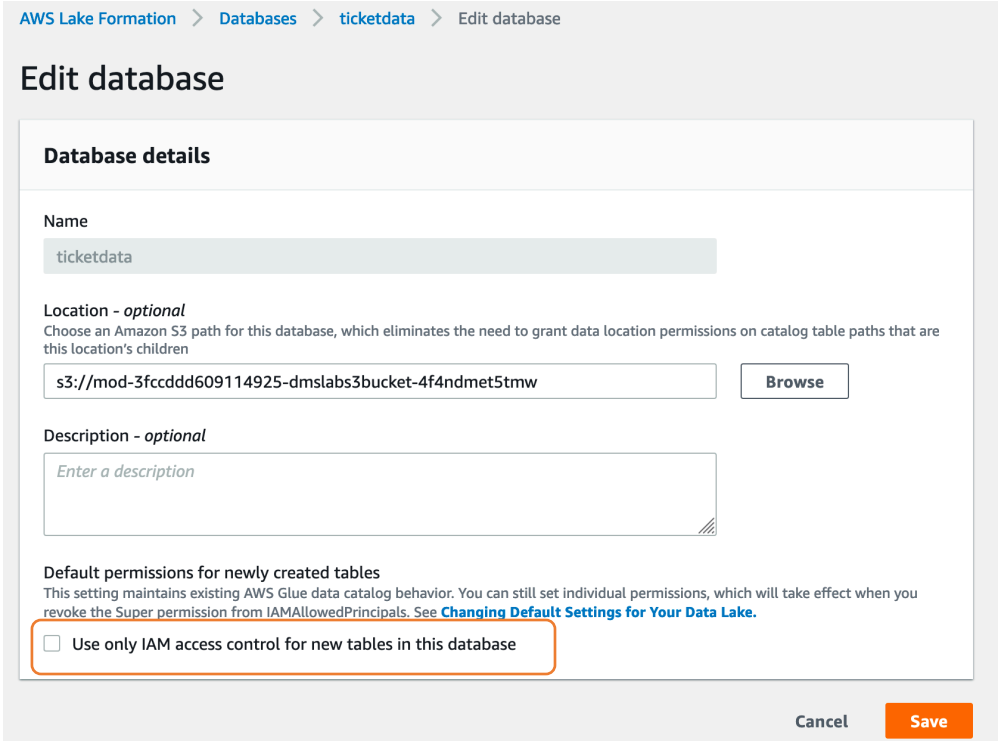
5. Under "**IAM Users and Roles**", select Lake Formation role that you created – "**LakeFormationWorkflowRole**" and for user, select your **username**. Grant "**super**" permissions for **Database permissions and Grantable permissions.**



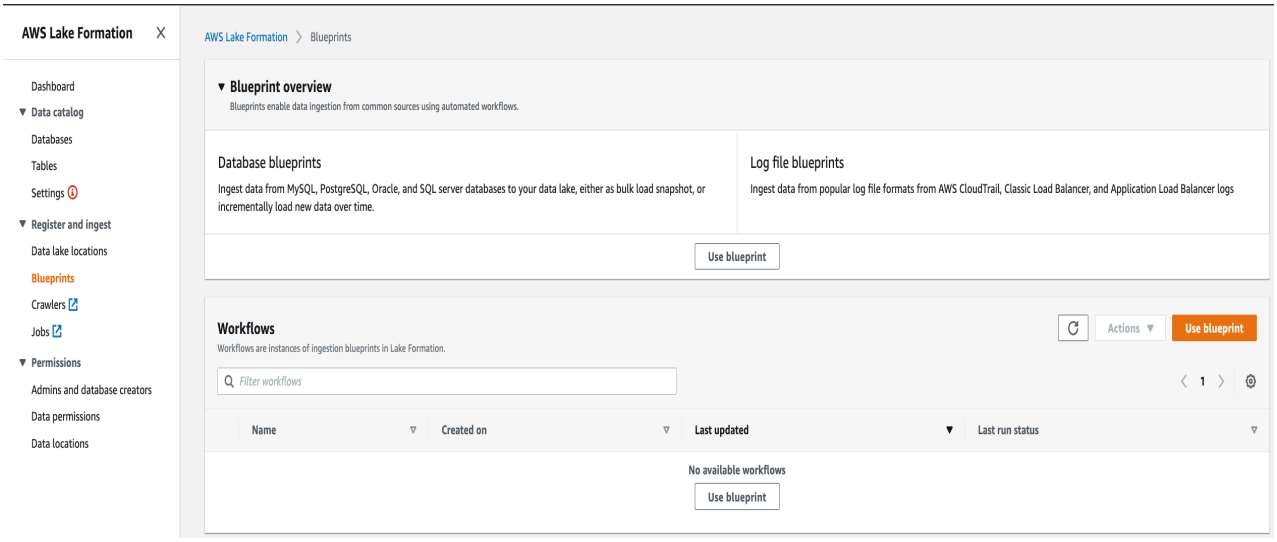
6. Add database Location by **Edit** the existing database **ticketdata**



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.

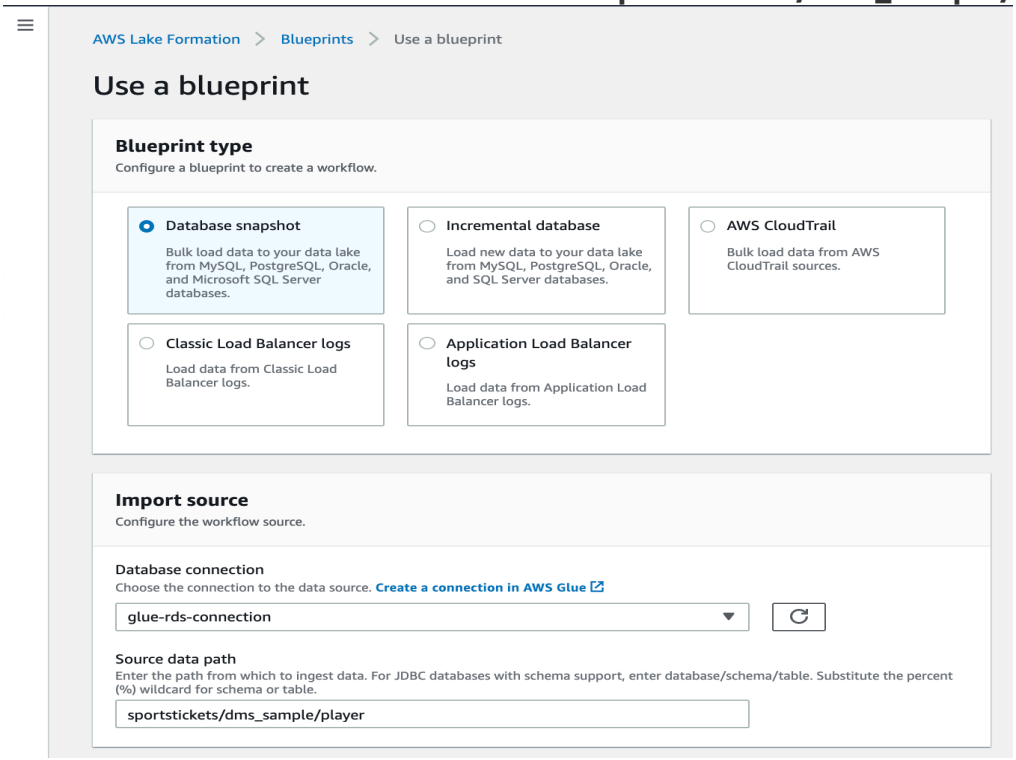


8. On the left pane navigate to **Blueprints** click **Use blueprints**.

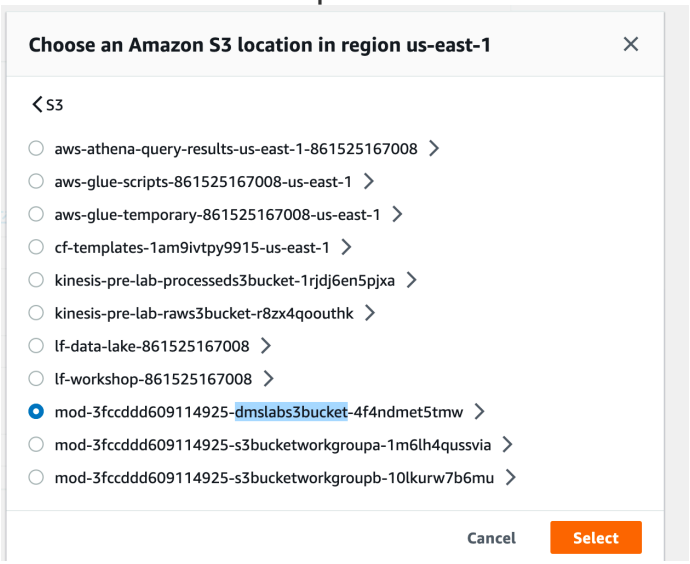


a. For **Blueprint Type**, select **Database snapshot**
b. Under Import Source

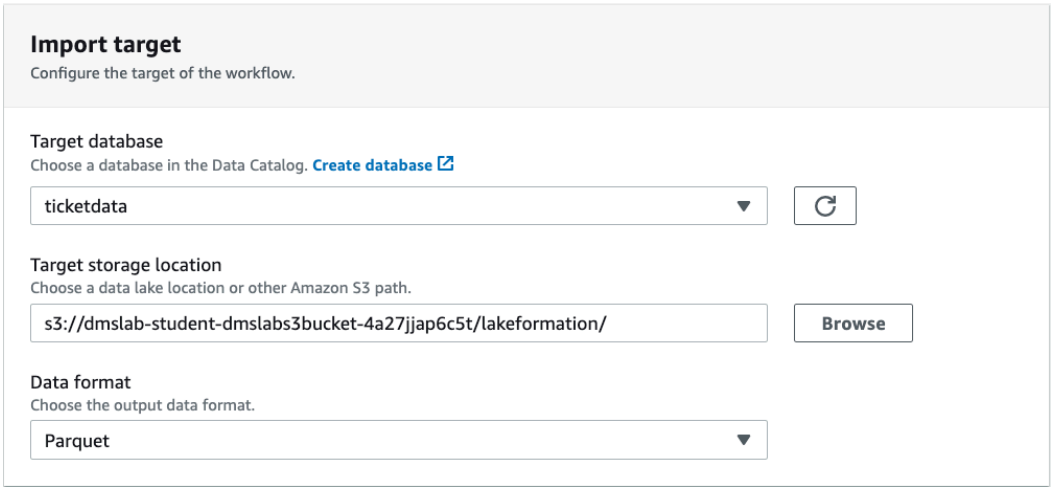
i. For **Database Connection** choose the DB connection created in the glue. [Ex: “glue-rds-connection”]
ii. For **Source Data Path** enter **sportstickets/dms_sample/player**



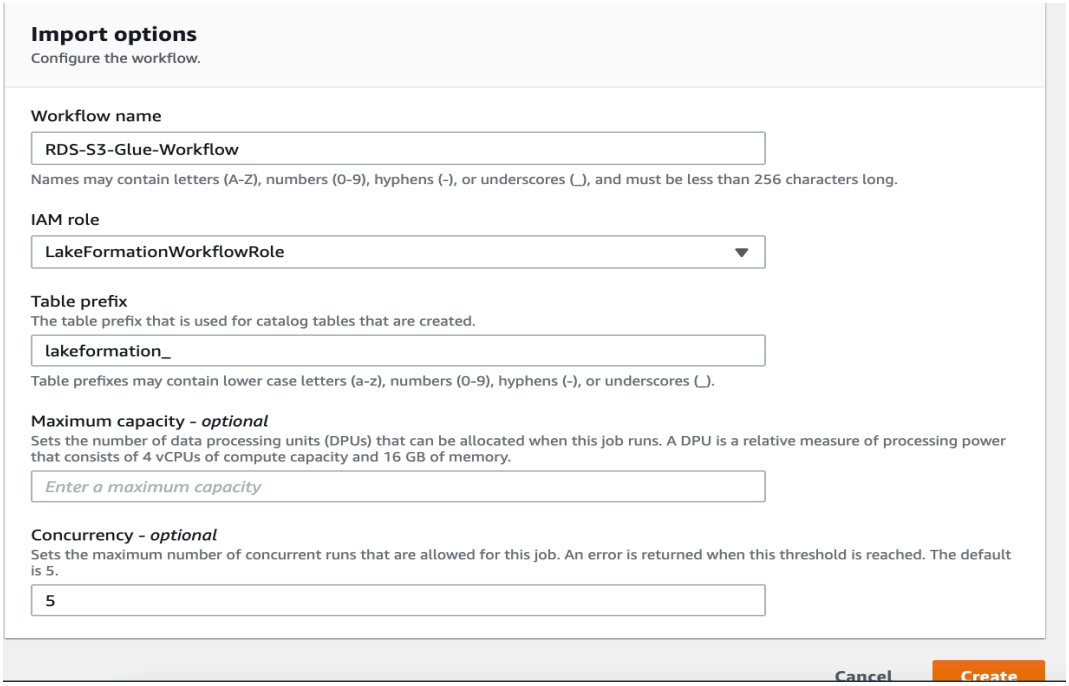
c. Under Import Target
   i. For **Target Database**, choose existing “**ticketdata**”.
   ii. For **Target storage location** browse and select the dmslab S3 bucket created in the previous lab.



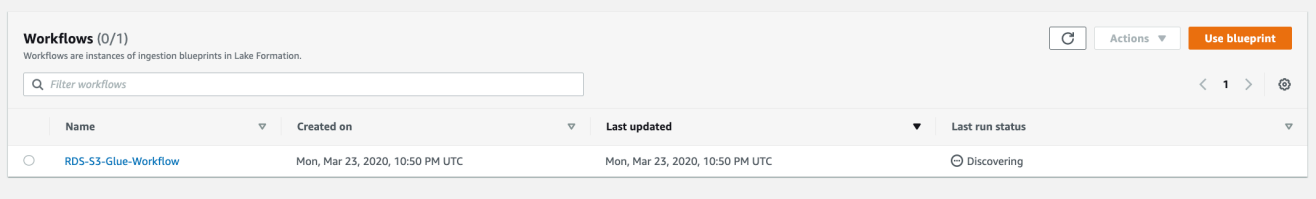
iii. Add a folder at the end of the bucket url path. NOTE: The value is similar to the following string "dmslab-student-dmslabs3bucket-q29vwhf8n41w/**lakeformation**"
iv. For **Data Format** choose **Parquet**



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**



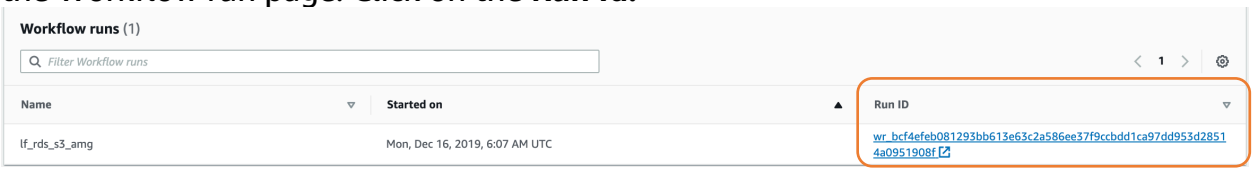
9. Leave other options as default, Choose **Create**, and wait for the console to report that the workflow was successfully created.
10. Once the blueprint gets created, click on **Start it Now?** There may be a delay of 5-10s delay in the blueprint showing up. You may have to **hit refresh**. Select 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



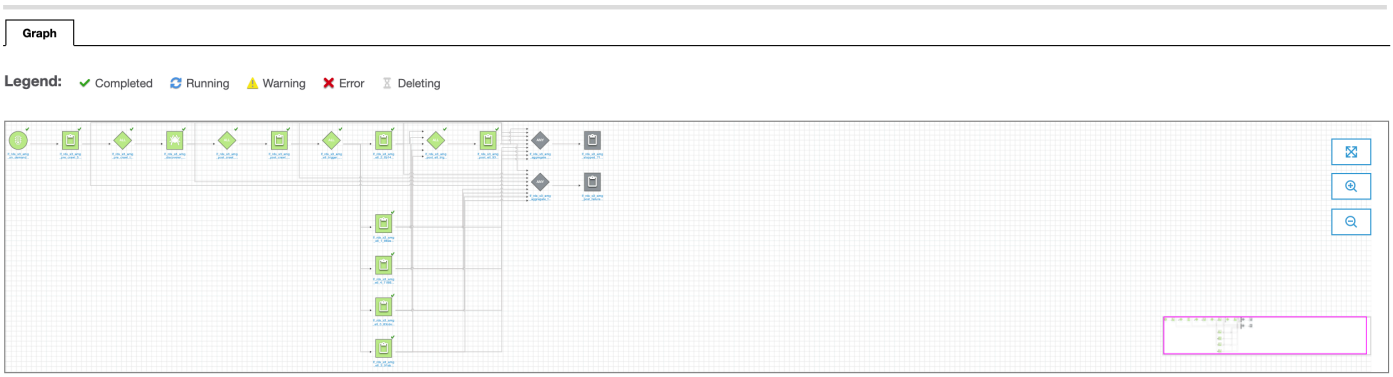
Explore the Underlying Components of a Blueprint

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.**

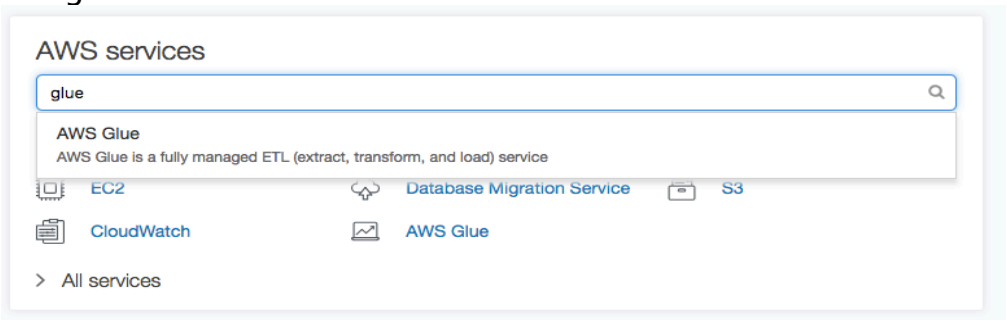


3. On the Glue Console page, click on **Get Started** and head to ETL **Workflows** in the navigation pane.
4. Here you can see the graphical representation of the Glue workflow built by Lake Formation blueprint. Highlighting and clicking on individual components will display the details of those components (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.

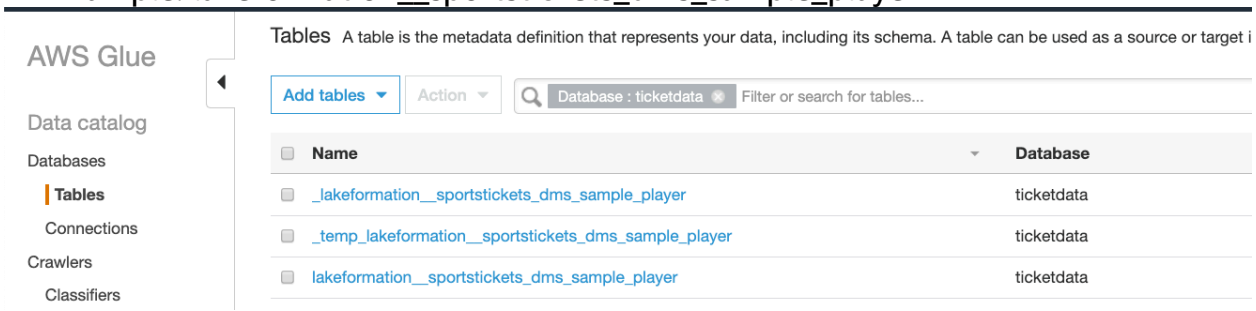


Explore workflow results in Athena

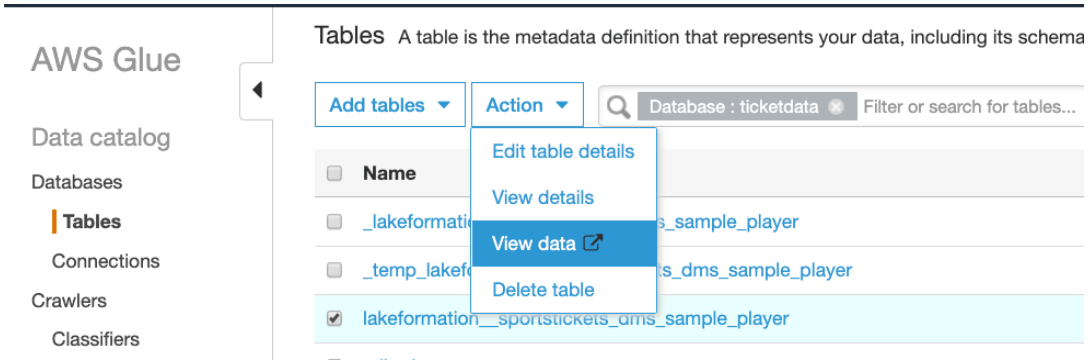
1. Navigate to the Glue console



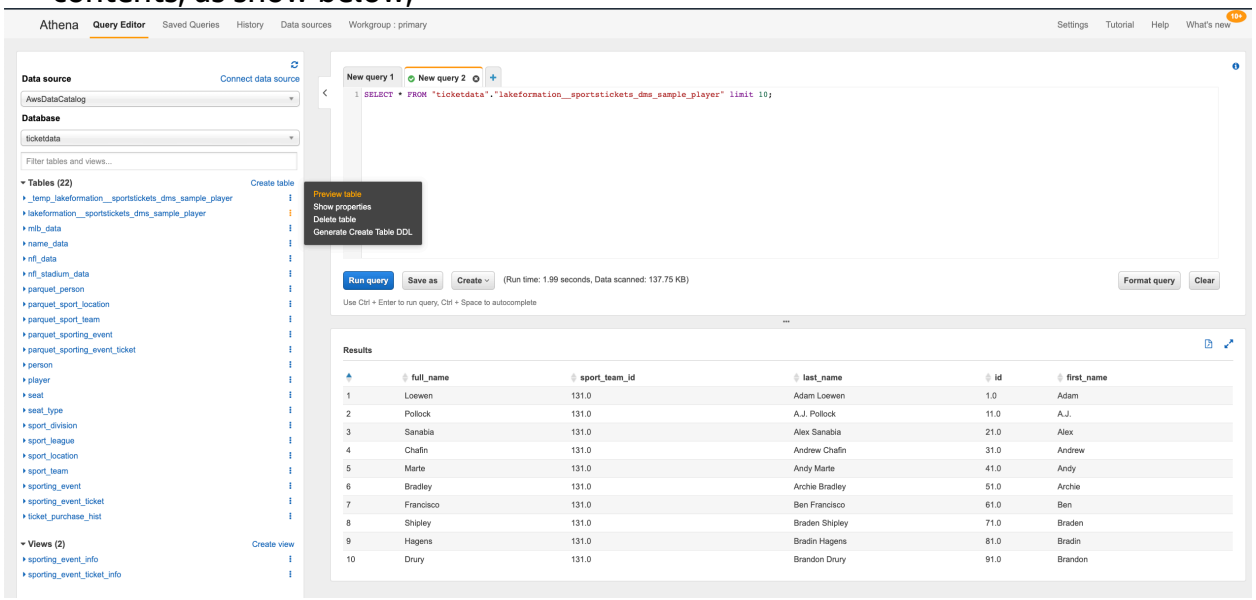
2. Navigate to **Databases** on the left panel and select **ticketdata**
3. Click on "Tables in ticketdata" and this table will be pre fixed by "**lakeformation_**"
   Example: lakeformation__sportstickets_dms_sample_player



4. And Click **Action** -> **View Data**



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

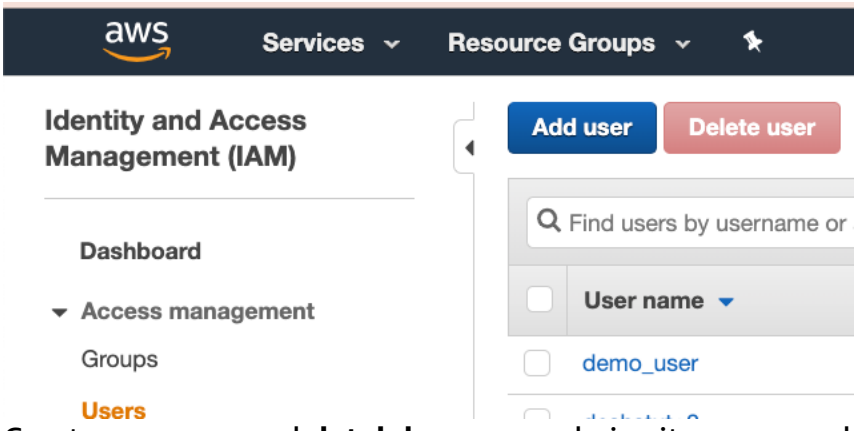


Congratulation!!! You have completed lake formation lab. To explore more fine grain data lake security feature, continue to next section.

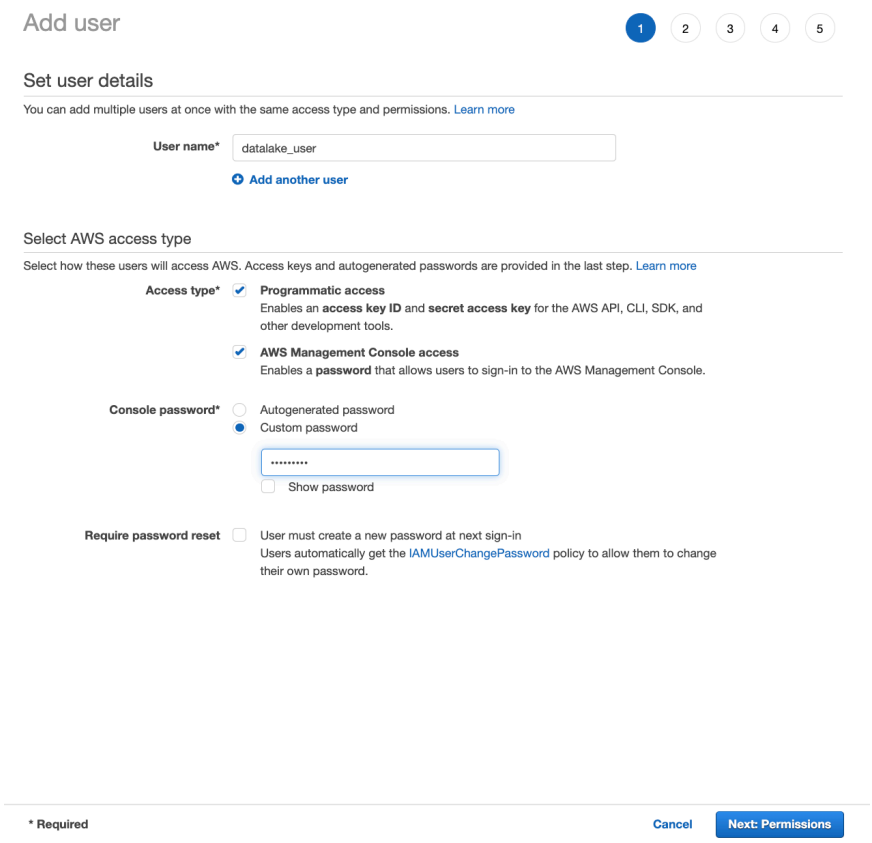
[Optional] Grant fine grain access controls to Data Lake user

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.**

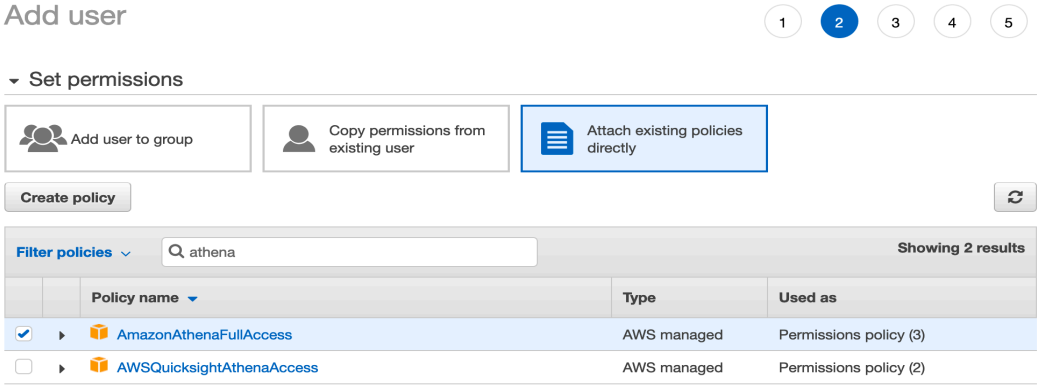
1. Login as admin user to your account. Navigate to **IAM Console** and click on **Add User**.



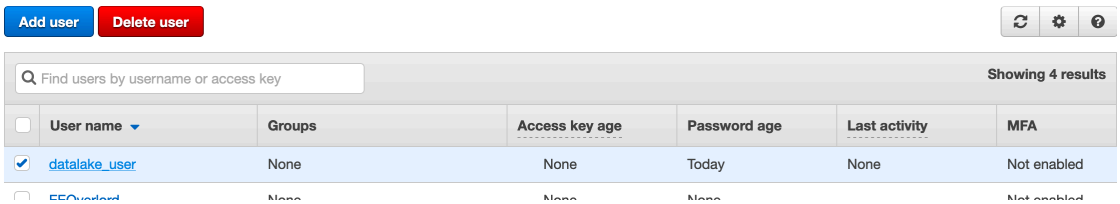
2. Create a user named **datalake_user** and give it a password: **master123**.

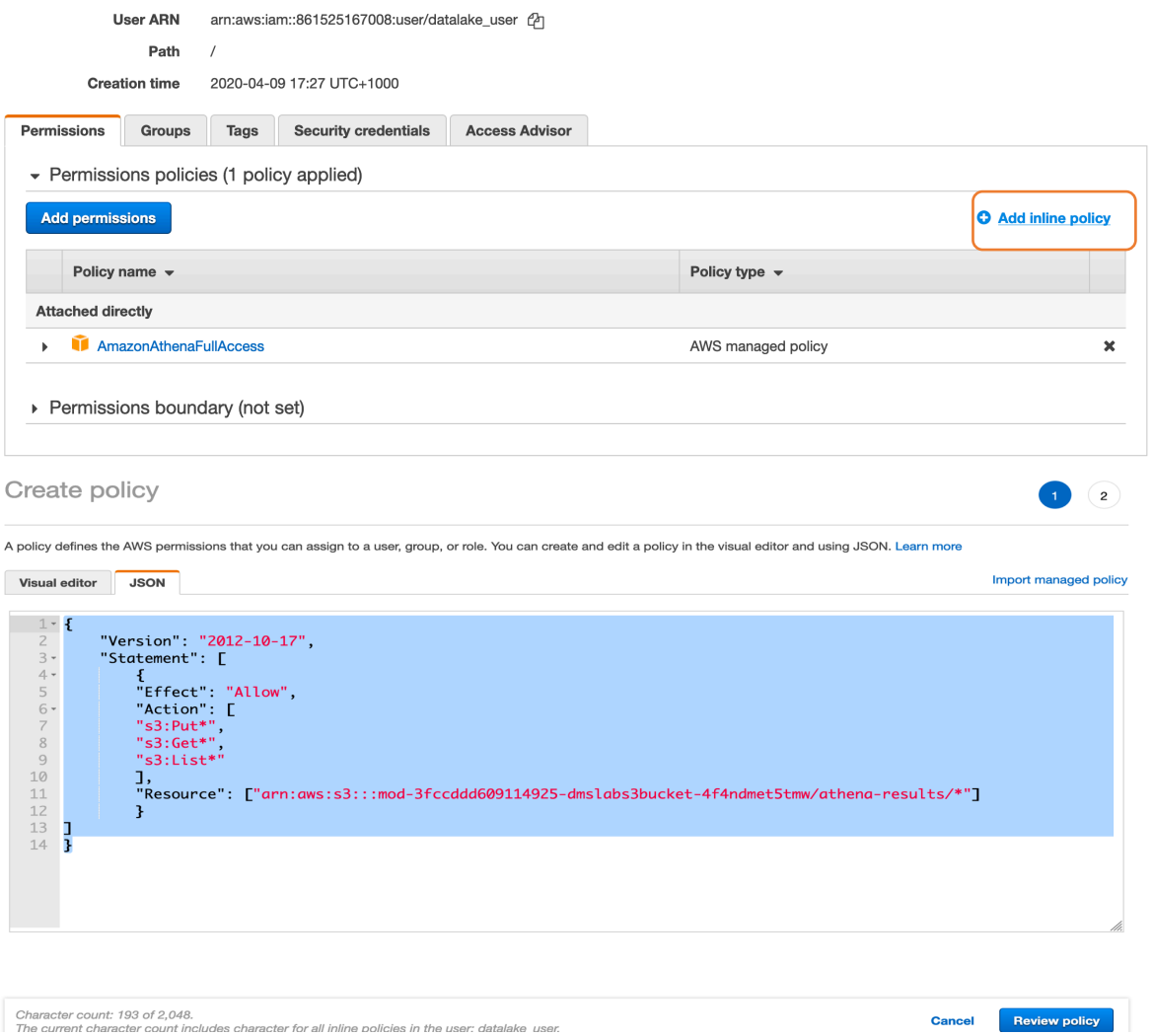


3. Next click on **Permissions**
4. Choose **Attach existing policies directly** and search for **AthenaFullAccess**



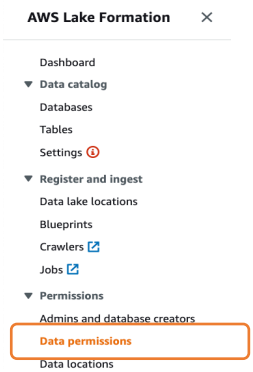
5. Keep navigating to the next steps until reached the end. Review the details and click on "**Create User**".
6. Click on the **datalake_user** user, and **add inline policy**

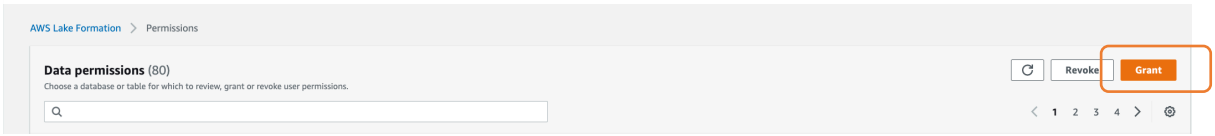




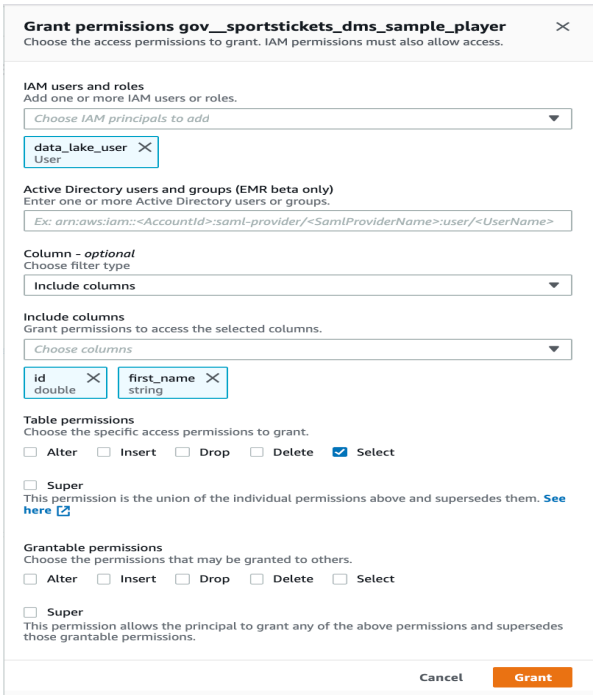

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    "Effect": "Allow",
    "Action": [
    "s3:Put*",
    "s3:Get*",
    "s3:List*"
    ],
    "Resource": [
      "arn:aws:s3:::<your_dmslabs3bucket_unique_name>/athena-results/*"
      ]
    }
  ]
}
```

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**.





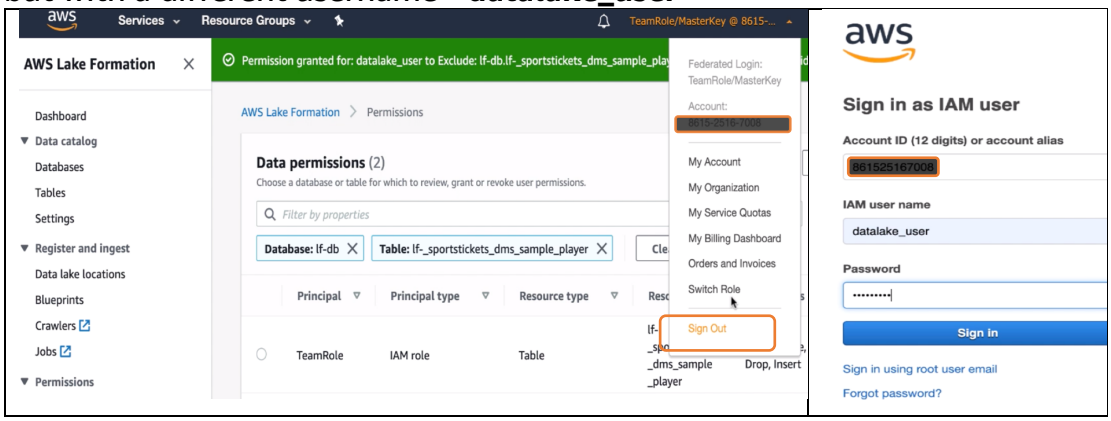
9. Choose **Grant**, and in the **Grant permissions** dialog box, do the following:
    a. For **IAM user and roles**, choose **datalake_user.**
    b. For **Database**, choose **ticketdata**
    c. The **Table** list populates.
    d. For **Table**, choose the table shown.
    e. For Columns, select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



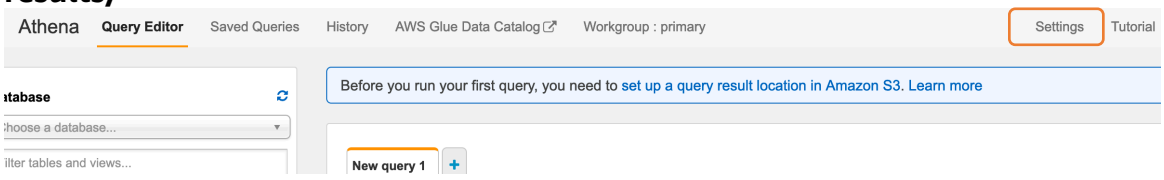
[Optional] Verify data permissions using Athena

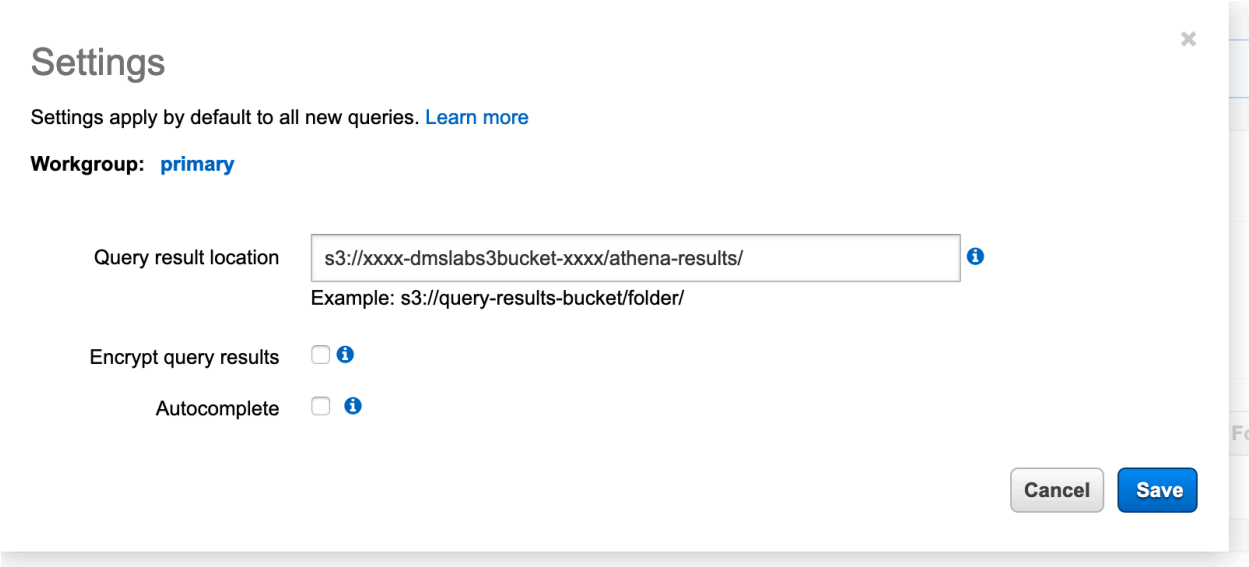
Using Athena, let us now explore the data set as the **datalake_user**.

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



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/**"





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.