Event ends in 5 hours 46 minutes.

Setup Athena Connectors and Catalogs

Athena uses data source connectors that run on AWS Lambda to run federated queries. A data source connector is a piece of code that can translate between your target data source and Athena. Prebuilt Athena data source connectors exist for data sources like Amazon CloudWatch Logs, Amazon DynamoDB, Amazon DocumentDB, and Amazon RDS, and JDBC-compliant relational data sources such MySQL, and PostgreSQL under the Apache 2.0 license.

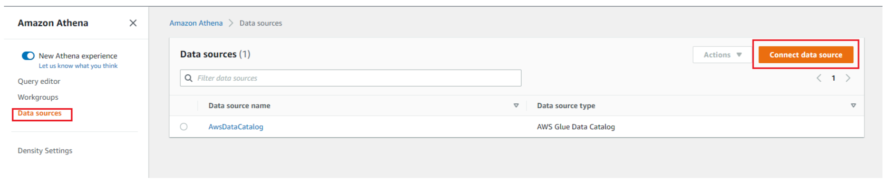
In this section, we will first deploy a connector for DynamoDB data source and then a connector for Aurora PostgreSQL data source.

Preparing to create federated queries is a two-part process: deploying a Lambda function data source connector, and connecting the Lambda function to a data source. In the first part, you give the Lambda function a name that you can later choose in the Athena console. In the second part, you give the connector a name that you can reference in your SQL queries.

1.1 Deploy Amazon DynamoDB Data Source Connector

This connector enables Amazon Athena to communicate with DynamoDB, making the *trips* tables accessible via SQL. For more information about Amazon DynamoDB Connector usage, parameters and limitations, refer to documentation.

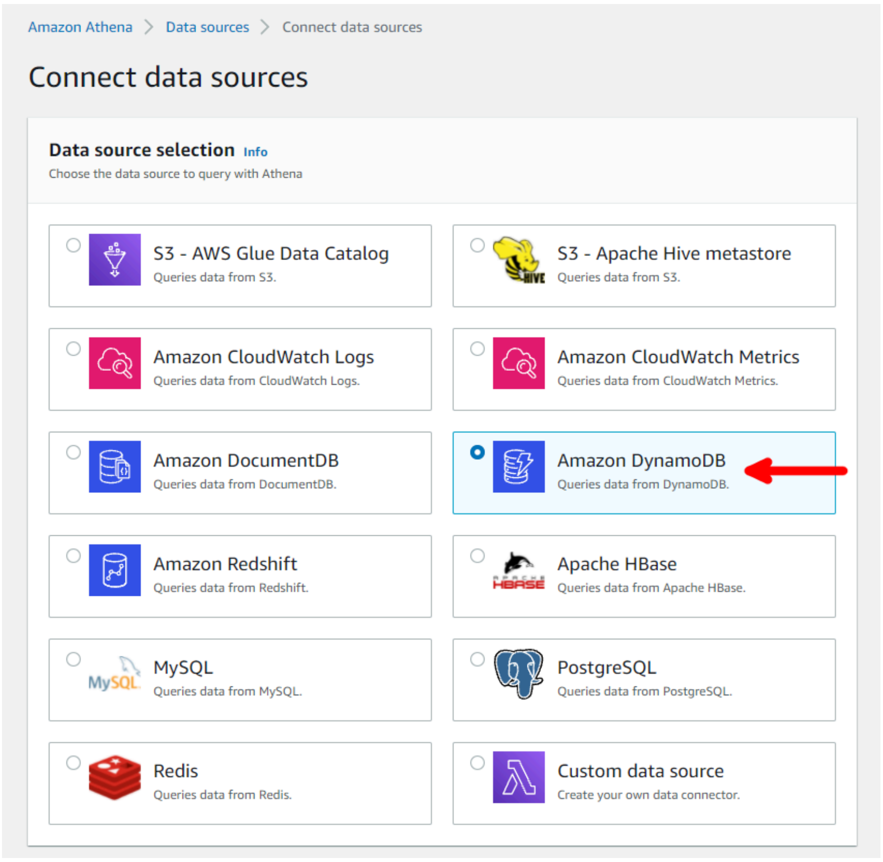
1. Choose Data sources in the Amazon Athena console navigation bar and click **Connect data source**.



Data sources (1)

	Data source name	Data source type
○	AwsDataCatalog	AWS Glue Data Catalog

2. Under **Data source selection** section, choose **Amazon DynamoDB**.

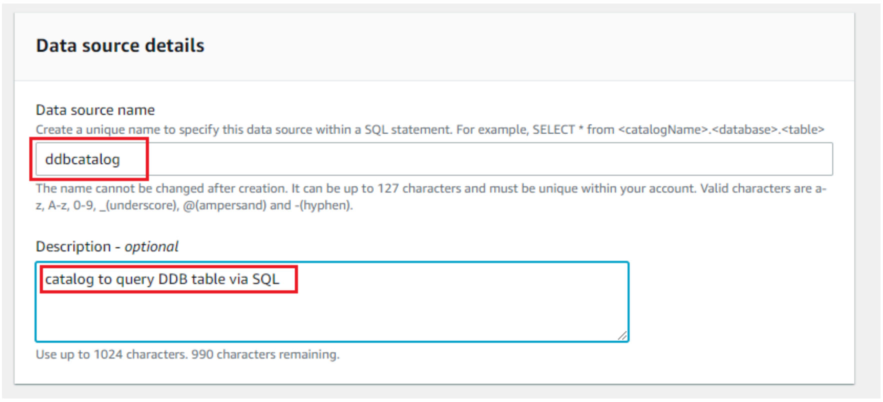

Amazon Athena > Data sources > Connect data sources

Connect data sources

Data source selection Info
Choose the data source to query with Athena

- S3 - AWS Glue Data Catalog — Queries data from S3.
- S3 - Apache Hive metastore — Queries data from S3.
- Amazon CloudWatch Logs — Queries data from CloudWatch Logs.
- Amazon CloudWatch Metrics — Queries data from CloudWatch Metrics.
- Amazon DocumentDB — Queries data from DocumentDB.
- [✓ Amazon DynamoDB — Queries data from DynamoDB.]
- Amazon Redshift — Queries data from Redshift.
- Apache HBase — Queries data from Apache HBase.
- MySQL — Queries data from MySQL.
- PostgreSQL — Queries data from PostgreSQL.
- Redis — Queries data from Redis.
- Custom data source — Create your own data connector.

3. For **Data source name** specify *ddbcatalog* and for **Description - optional** specify *catalog to query DDB table via SQL*.

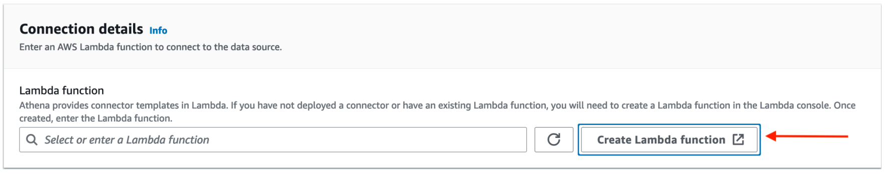

Data source details

Data source name
Create a unique name to specify this data source within a SQL statement. For example, SELECT * from <catalogName>.<database>.<table>
ddbcatalog
The name cannot be changed after creation. It can be up to 127 characters and must be unique within your account. Valid characters are a-z, A-z, 0-9, _(underscore), @(ampersand) and -(hyphen).

Description - *optional*
catalog to query DDB table via SQL
Use up to 1024 characters. 990 characters remaining.

4. Under **Lambda function** section, choose **Create a new Lambda function** and click **Create a new Lambda function in AWS Lambda**. A new browser window/tab will open.

> Leave the current browser window/tab open. After creating the Lambda function in the new browser window/tab, we will come back to the current browser window/tab to finish the setup.


Connection details Info
Enter an AWS Lambda function to connect to the data source.

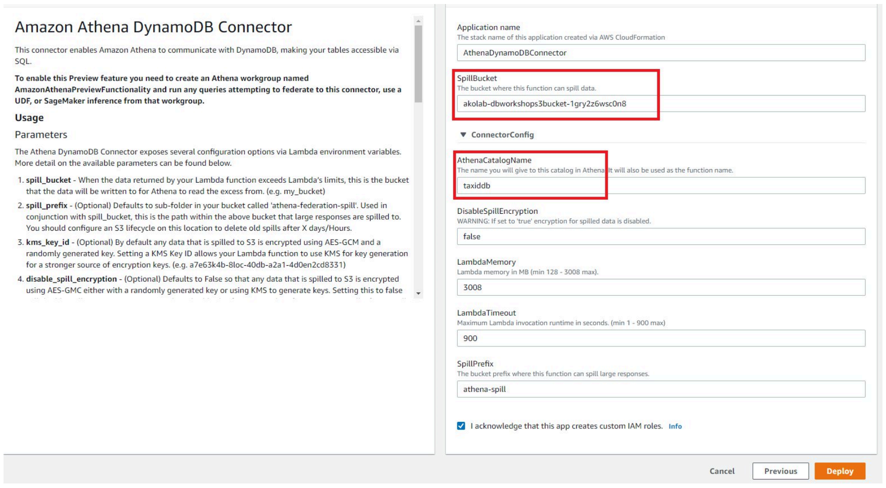
Lambda function
Athena provides connector templates in Lambda. If you have not deployed a connector or have an existing Lambda function, you will need to create a Lambda function in the Lambda console. Once created, enter the Lambda function.

5. You will be be taken to AWS Lambda console where the connector will be deployed as a SAM Application. Provide values for the following parameters and leave the rest to the default.

   SpillBucket: Specify the S3 bucket name that was created as part of the CloudFormation Stack (Look for **S3bucketName** in the parent CloudFormation stack Outputs section) e.g. *cfn-dbworkshops3bucket-1xihfupnzuugu*

AthenaCatalogName: taxiddb

Select the option **I acknowledge that this app creates custom IAM Roles** and click **Deploy**.


Amazon Athena DynamoDB Connector

This connector enables Amazon Athena to communicate with DynamoDB, making your tables accessible via SQL.

To enable this Preview feature you need to create an Athena workgroup named AmazonAthenaPreviewFunctionality and run any queries attempting to federate to this connector, use a UDF, or SageMaker inference from that workgroup.

Usage

Parameters

The Athena DynamoDB Connector exposes several configuration options via Lambda environment variables. More detail on the available parameters can be found below.

1. **spill_bucket** - When the data returned by your Lambda function exceeds Lambda's limits, this is the bucket that the data will be written to for Athena to read the excess from. (e.g. my_bucket)
2. **spill_prefix** - (Optional) Defaults to sub-folder in your bucket called 'athena-federation-spill'. Used in conjunction with spill_bucket, this is the path within the above bucket that large responses are spilled to. You should configure an S3 lifecycle on this location to delete old spills after X days/Hours.
3. **kms_key_id** - (Optional) By default any data that is spilled to S3 is encrypted using AES-GCM and a randomly generated key. Setting a KMS Key ID allows your Lambda function to use KMS for key generation for a stronger source of encryption keys. (e.g. a7e63k4b-8loc-40db-a2a1-4d0en2cd8331)
4. **disable_spill_encryption** - (Optional) Defaults to False so that any data that is spilled to S3 is encrypted using AES-GMC either with a randomly generated key or using KMS to generate keys. Setting this to false

Application name
The stack name of this application created via AWS CloudFormation
AthenaDynamoDBConnector

SpillBucket
The bucket where this function can spill data.
akolab-dbworkshops3bucket-1gry2z6wsc0n8

▼ ConnectorConfig

AthenaCatalogName
The name you will give to this catalog in Athena. It will also be used as the function name.
taxiddb

DisableSpillEncryption
WARNING: If set to 'true' encryption for spilled data is disabled.
false

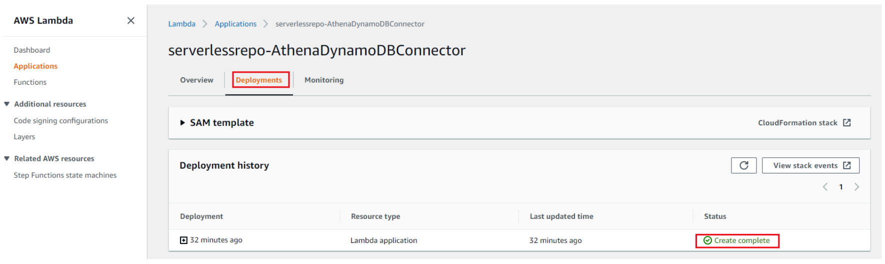
LambdaMemory
Lambda memory in MB (min 128 - 3008 max).
3008

LambdaTimeout
Maximum Lambda invocation runtime in seconds. (min 1 - 900 max)
900

SpillPrefix
The bucket prefix where this function can spill large responses.
athena-spill

☑ I acknowledge that this app creates custom IAM roles. Info

6. A new browser window/tab will open showing the SAM deployment. You can monitor the progress by choosing the **Deployments** tab.

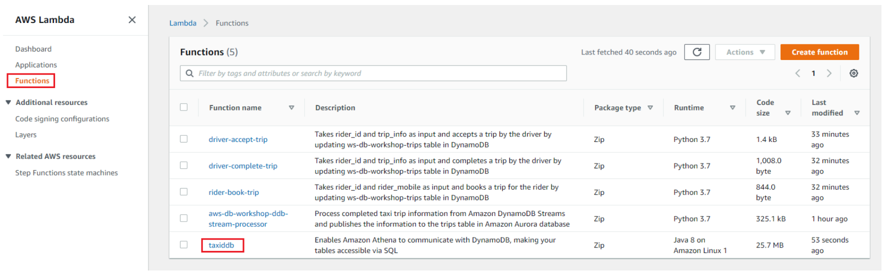

AWS Lambda
Dashboard
Applications
Functions
▼ Additional resources
Code signing configurations
Layers
▼ Related AWS resources
Step Functions state machines
Lambda > Applications > serverlessrepo-AthenaDynamoDBConnector

serverlessrepo-AthenaDynamoDBConnector

Deployment history

Deployment	Resource type	Last updated time	Status
32 minutes ago	Lambda application	32 minutes ago	Create complete

7. After successful deployment, you should see the **taxiddb** Lambda function deployed in your AWS account when you click Functions on the AWS Lambda console navigation bar.


AWS Lambda
Dashboard
Applications
Functions
▼ Additional resources
Code signing configurations
Layers
▼ Related AWS resources
Step Functions state machines
Lambda > Functions
Last fetched 40 seconds ago
Actions
Create function
Filter by tags and attributes or search by keyword

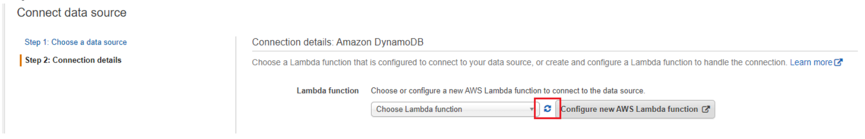
Functions (5)

Function name	Description	Package type	Runtime	Code size	Last modified
driver-accept-trip	Takes rider_id and trip_info as input and accepts a trip by the driver by updating ws-db-workshop-trips table in DynamoDB	Zip	Python 3.7	1.4 kB	33 minutes ago
driver-complete-trip	Takes rider_id and trip_info as input and completes a trip by the driver by updating ws-db-workshop-trips table in DynamoDB	Zip	Python 3.7	1,008.0 byte	32 minutes ago
rider-book-trip	Takes rider_id and rider_mobile as input and books a trip for the rider by updating ws-db-workshop-trips table in DynamoDB	Zip	Python 3.7	844.0 byte	32 minutes ago
aws-db-workshop-ddb-stream-processor	Process completed taxi trip information from Amazon DynamoDB Streams and publishes the information to the trips table in Amazon Aurora database	Zip	Python 3.7	325.1 kB	1 hour ago
taxiddb	Enables Amazon Athena to communicate with DynamoDB, making your tables accessible via SQL	Zip	Java 8 on Amazon Linux 1	25.7 MB	53 seconds ago

1.2 Finish setting up connection to DynamoDB Data Source

In this step, we will finish setting up connection to DynamoDB Data Source that we started in the previous step.

1. Go back to the previous Athena **Connect data sources** window in your browser and under the **Lambda functions** section, click the refresh button next to the **Lambda function** input.


Connect data source

Step 1: Choose a data source
Step 2: Connection details

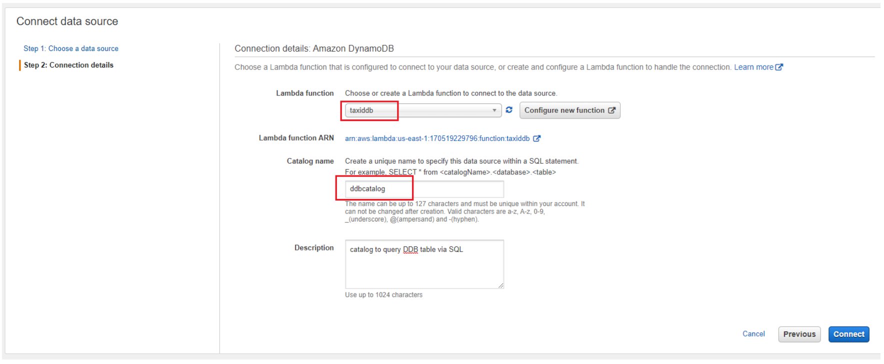
Connection details: Amazon DynamoDB

Choose a Lambda function that is configured to connect to your data source, or create and configure a Lambda function to handle the connection. Learn more

Lambda function: Choose or configure a new AWS Lambda function to connect to the data source.

Choose Lambda function
Configure new AWS Lambda function

2. Choose the Lambda function named **taxiddb** in the dropdown.



Connect data source

Step 1: Choose a data source
Step 2: Connection details

Connection details: Amazon DynamoDB

Choose a Lambda function that is configured to connect to your data source, or create and configure a Lambda function to handle the connection. Learn more

Lambda function: Choose or create a Lambda function to connect to the data source.
taxiddb
Configure new function

Lambda function ARN: arn:aws:lambda:us-east-1:170519229796:function:taxiddb

Catalog name: Create a unique name to specify this data source within a SQL statement. For example, SELECT * from <catalogName>.<database>.<table>
ddbcatalog
The name can be up to 127 characters and must be unique within your account. It can not be changed after creation. Valid characters are a-z, A-Z, 0-9, _(underscore), @(ampersand) and -(hyphen).

Description: catalog to query DDB table via SQL
Use up to 1024 characters

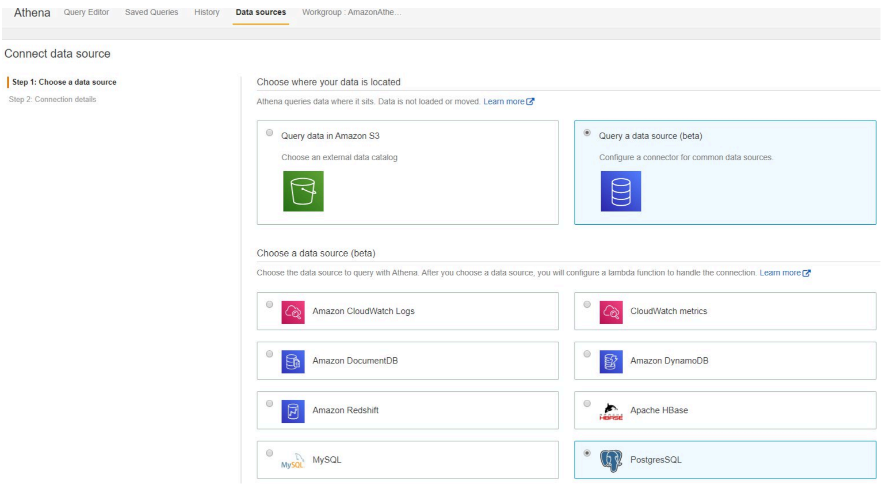
3. Click **Connect data source**.

Now, we need to repeat the same process to deploy connector for Aurora PostgreSQL data source.

2.1 Deploy Aurora PostgreSQL Data Source Connector

This connector enables Amazon Athena to access your Amazon RDS and Amazon Aurora databases using JDBC driver. For more information about the Amazon Athena Lambda Jdbc Connector usage, parameters and limitations, refer to documentation.

1. Choose Data sources in Amazon Athena console navigation bar and click **Connect data source**.
2. Under **Data source selection** section, choose **PostgreSQL**.



Connect data source

Step 1: Choose a data source
Step 2: Connection details

Choose where your data is located

Athena queries data where it sits. Data is not loaded or moved. Learn more

Query data in Amazon S3 — Choose an external data catalog

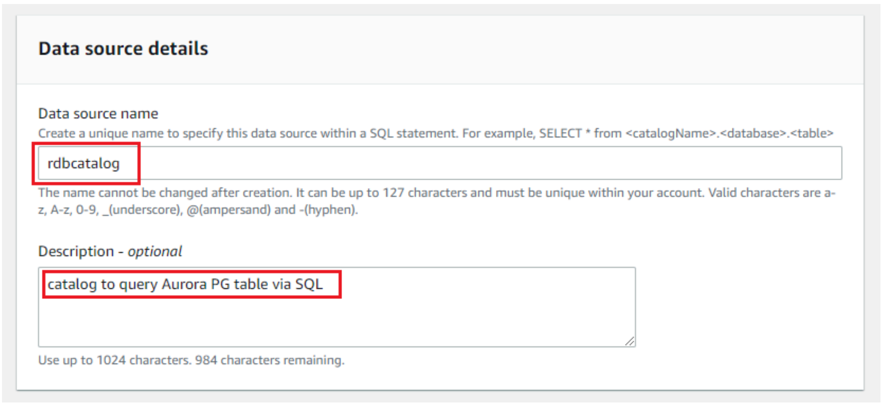
[✓ Query a data source (beta)] — Configure a connector for common data sources.

Choose a data source (beta)

Choose the data source to query with Athena. After you choose a data source, you will configure a lambda function to handle the connection. Learn more

Amazon CloudWatch Logs
CloudWatch metrics
Amazon DocumentDB
Amazon DynamoDB
Amazon Redshift
Apache HBase
MySQL
[✓ PostgresSQL]

3. For **Data source name** specify *rdbcatalog* and for **Description - optional** specify *catalog to query Aurora PG table via SQL*.



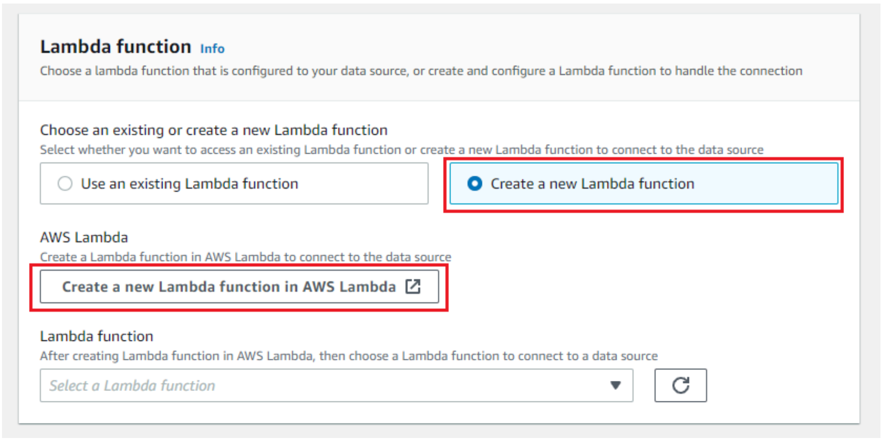
Data source details

Data source name
Create a unique name to specify this data source within a SQL statement. For example, SELECT * from <catalogName>.<database>.<table>
rdbcatalog
The name cannot be changed after creation. It can be up to 127 characters and must be unique within your account. Valid characters are a-z, A-Z, 0-9, _(underscore), @(ampersand) and -(hyphen).

Description - *optional*
catalog to query Aurora PG table via SQL
Use up to 1024 characters. 984 characters remaining.

4. Under **Lambda function** section, choose **Create a new Lambda function** and click **Create a new Lambda function in AWS Lambda**. A new browser window/tab will open.

Leave the current browser window/tab open. After creating the Lambda function in the new browser window/tab, we will come back to the current browser window/tab to finish the setup.


Lambda function Info

Choose a lambda function that is configured to your data source, or create and configure a Lambda function to handle the connection

Choose an existing or create a new Lambda function

Select whether you want to access an existing Lambda function or create a new Lambda function to connect to the data source

Use an existing Lambda function [✓ Create a new Lambda function]

AWS Lambda

Create a Lambda function in AWS Lambda to connect to the data source

Lambda function

After creating Lambda function in AWS Lambda, then choose a Lambda function to connect to a data source

5. You will be be taken to AWS Lambda console where the connector will be deployed as a SAM Application. Provide values for the following parameters and leave the rest to the default.

   SecretNamePrefix : dbadmin

   SpillBucket: Specify the S3 bucket name that was created as part of the CloudFormation Stack (Look for **S3bucketName** in the parent CloudFormation stack Outputs section) e.g. *cfn-dbworkshops3bucket-1xihfupnzuugu*

   DefaultConnectionString : postgres:// `<AuroraJDBCConnectionString from the output of parent CloudFormation stack>` ?user=auradmin&password=<output of command - echo $PGPASSWORD>

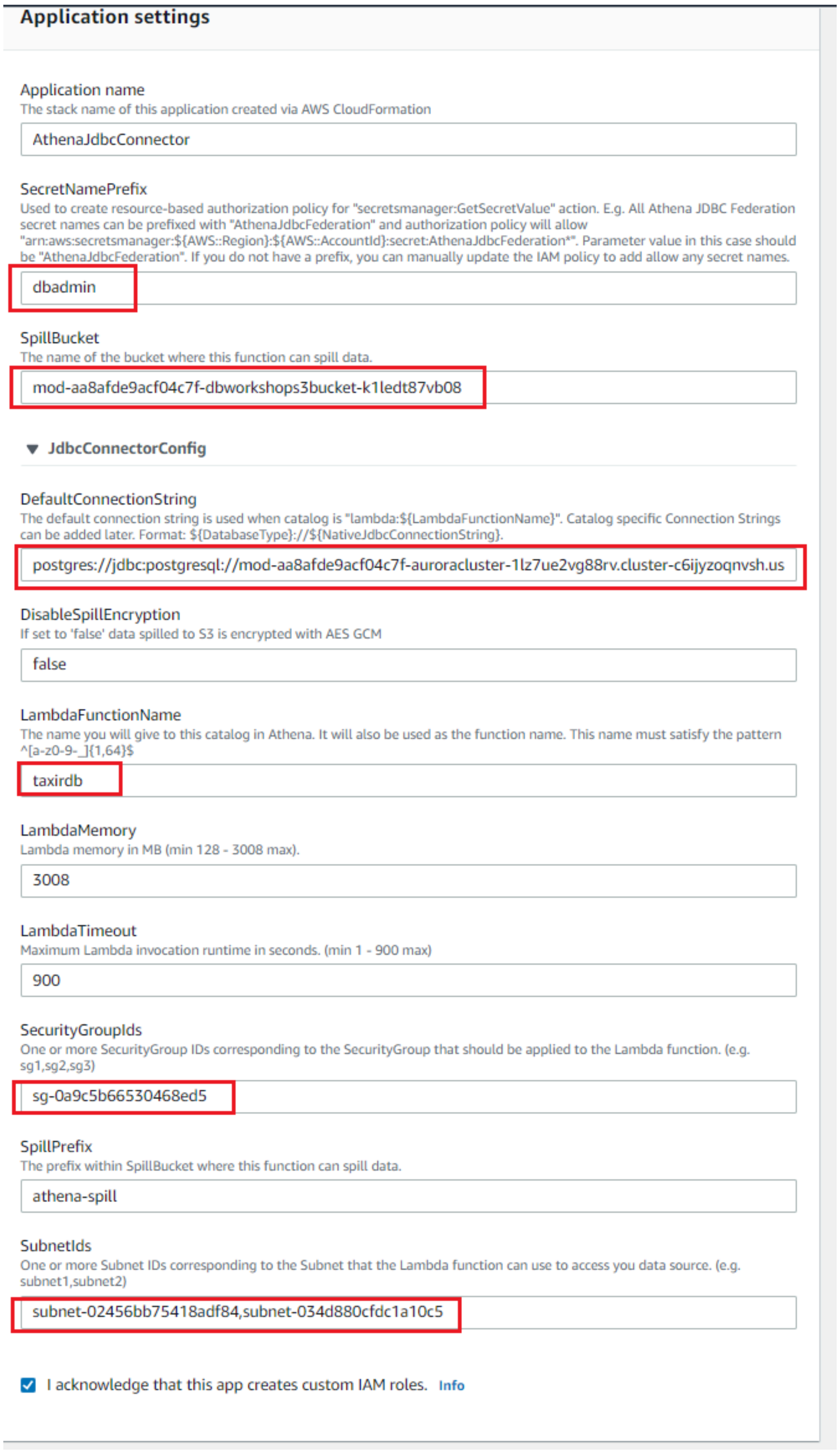
   e.g. *postgres://jdbc:postgresql://cfn-aa8afde9acf04c7f-auroracluster-bisaobgtnnm0.cluster-cpy3mfafjonu.us-west-2.rds.amazonaws.com:5432/taxidb?user=auradmin&password=auradmin123*

   LambdaFunctionName : taxirdb

   SecurityGroupIds : specify the value for output key **LambdaSecurityGroupId** from the outputs of CloudFormation stack

   SubnetIds : specify the values for output keys **LambdaSubnet1**,**LambdaSubnet2** (separated by commas) from the output of CloudFormation stack

   Select the option **I acknowledge that this app creates custom IAM Roles** and click **Deploy**.


Application settings

Application name
The stack name of this application created via AWS CloudFormation
AthenaJdbcConnector

SecretNamePrefix
Used to create resource-based authorization policy for "secretsmanager:GetSecretValue" action. E.g. All Athena JDBC Federation secret names can be prefixed with "AthenaJdbcFederation" and authorization policy will allow "arn:aws:secretsmanager:${AWS::Region}:${AWS::AccountId}:secret:AthenaJdbcFederation*". Parameter value in this case should be "AthenaJdbcFederation". If you do not have a prefix, you can manually update the IAM policy to add allow any secret names.
dbadmin

SpillBucket
The name of the bucket where this function can spill data.
mod-aa8afde9acf04c7f-dbworkshops3bucket-k1ledt87vb08

▼ JdbcConnectorConfig

DefaultConnectionString
The default connection string is used when catalog is "lambda:${LambdaFunctionName}". Catalog specific Connection Strings can be added later. Format: ${DatabaseType}://${NativeJdbcConnectionString}.
postgres://jdbc:postgresql://mod-aa8afde9acf04c7f-auroracluster-1lz7ue2vg88rv.cluster-c6ijyzoqnvsh.us

DisableSpillEncryption
If set to 'false' data spilled to S3 is encrypted with AES GCM
false

LambdaFunctionName
The name you will give to this catalog in Athena. It will also be used as the function name. This name must satisfy the pattern ^[a-z0-9-_]{1,64}$
taxirdb

LambdaMemory
Lambda memory in MB (min 128 - 3008 max).
3008

LambdaTimeout
Maximum Lambda invocation runtime in seconds. (min 1 - 900 max)
900

SecurityGroupIds
One or more SecurityGroup IDs corresponding to the SecurityGroup that should be applied to the Lambda function. (e.g. sg1,sg2,sg3)
sg-0a9c5b66530468ed5

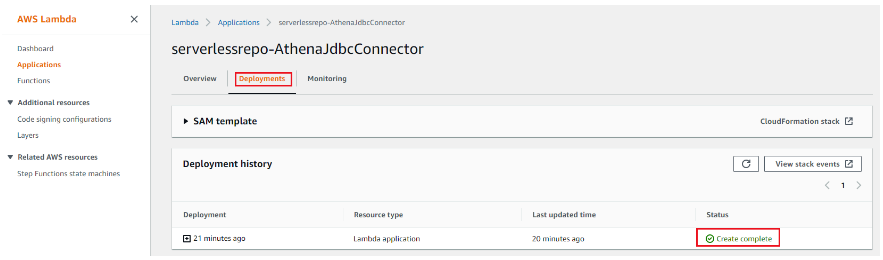
SpillPrefix
The prefix within SpillBucket where this function can spill data.
athena-spill

SubnetIds
One or more Subnet IDs corresponding to the Subnet that the Lambda function can use to access you data source. (e.g. subnet1,subnet2)
subnet-02456bb75418adf84,subnet-034d880cfdc1a10c5

☑ I acknowledge that this app creates custom IAM roles. Info

The JDBC connector can connect to database using credentials stored in AWS Secrets manager or directly by specifying an userid and password. For this lab, we will specify the userid and password directly in the connection string. We have still provided a dummy value as a secretname prefix as this parameter is mandatory.

6. A new browser window/tab will open showing the SAM deployment. You can monitor the progress by choosing the **Deployments** tab.

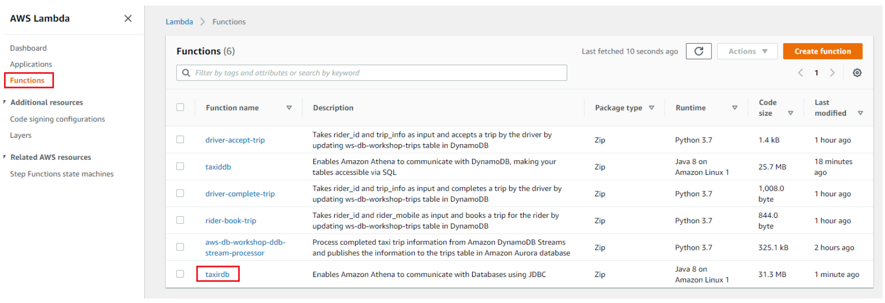


serverlessrepo-AthenaJdbcConnector

Deployment history

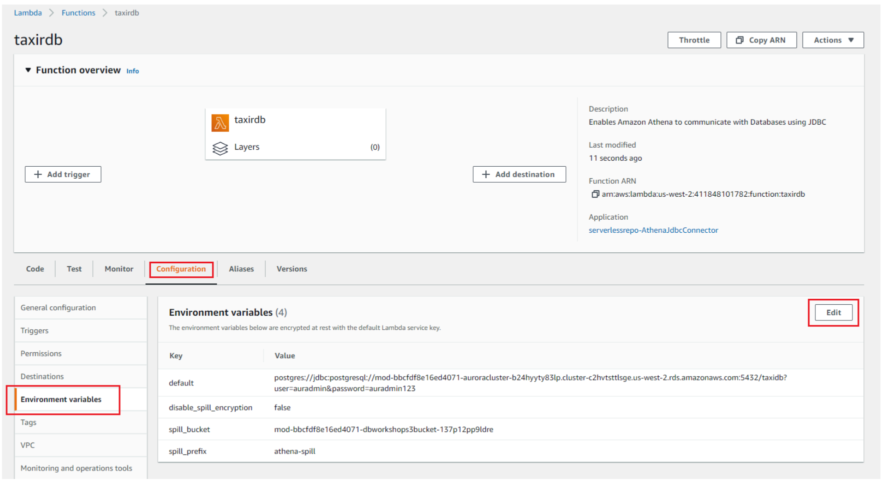
Deployment	Resource type	Last updated time	Status
21 minutes ago	Lambda application	20 minutes ago	Create complete

7. After successful deployment, you should see the **taxirdb** Lambda function deployed in your AWS account when you click Functions on the AWS Lambda console navigation bar.



Function name	Description	Package type	Runtime	Code size	Last modified
driver-accept-trip	Takes rider_id and trip_info as input and accepts a trip by the driver by updating ws-db-workshop-trips table in DynamoDB	Zip	Python 3.7	1.4 kB	1 hour ago
taxiddb	Enables Amazon Athena to communicate with DynamoDB, making your tables accessible via SQL	Zip	Java 8 on Amazon Linux 1	25.7 MB	18 minutes ago
driver-complete-trip	Takes rider_id and trip_info as input and completes a trip by the driver by updating ws-db-workshop-trips table in DynamoDB	Zip	Python 3.7	1,008.0 byte	1 hour ago
rider-book-trip	Takes rider_id and rider_mobile as input and books a trip for the rider by updating ws-db-workshop-trips table in DynamoDB	Zip	Python 3.7	844.0 byte	1 hour ago
aws-db-workshop-ddb-stream-processor	Process completed taxi trip information from Amazon DynamoDB Streams and publishes the information to the trips table in Amazon Aurora database	Zip	Python 3.7	325.1 kB	2 hours ago
taxirdb	Enables Amazon Athena to communicate with Databases using JDBC	Zip	Java 8 on Amazon Linux 1	31.3 MB	1 minute ago

8. Click **taxirdb** Lambda function name and choose the **Configuration** tab. Choose **Environment variables** in the left navigation bar and click **Edit** to edit the Lambda Environment variables.



Description
Enables Amazon Athena to communicate with Databases using JDBC

Last modified
11 seconds ago

Function ARN
arn:aws:lambda:us-west-2:411848101782:function:taxirdb

Application
serverlessrepo-AthenaJdbcConnector

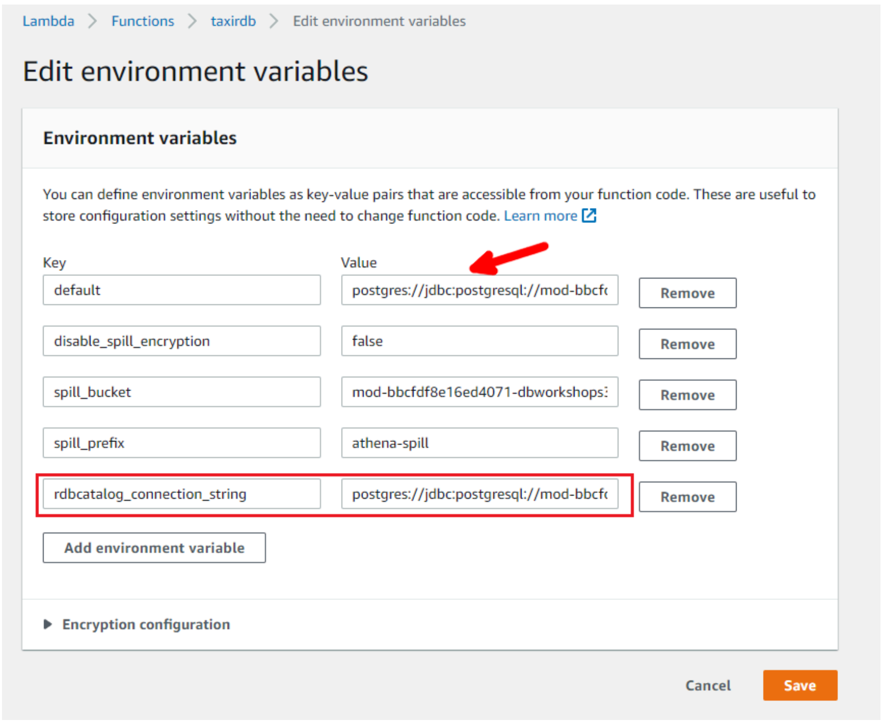
Environment variables (4)

The environment variables below are encrypted at rest with the default Lambda service key.

Key	Value
default	postgres://jdbc:postgresql://mod-bbcfdf8e16ed4071-auroracluster-b24hyyty83lp.cluster-c2hvtsttlsge.us-west-2.rds.amazonaws.com:5432/taxidb?user=auradmin&password=auradmin123
disable_spill_encryption	false
spill_bucket	mod-bbcfdf8e16ed4071-dbworkshops3bucket-137p12pp9ldre
spill_prefix	athena-spill

9. Choose **Add environment variable**. Add the following key and value. This is required for Athena to connect to multiple database instances of any type using a single Lambda function. Refer Athena JDBC Multiplexing handler parameters for more details.

Key	Value
rdbcatalog_connection_string	Copy and paste the same Value for the Key named *default*



Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more

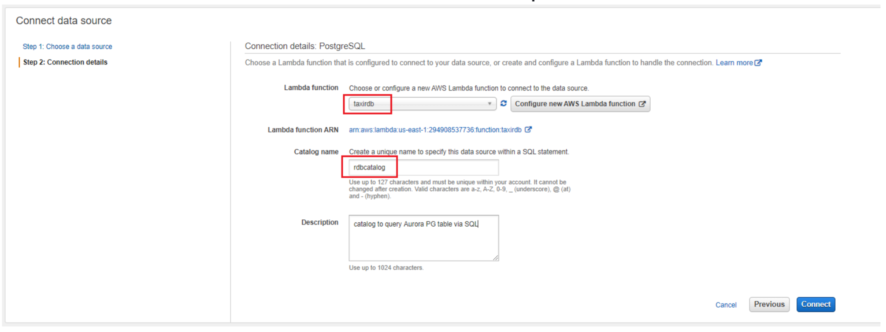
Key	Value
default	postgres://jdbc:postgresql://mod-bbcf
disable_spill_encryption	false
spill_bucket	mod-bbcfdf8e16ed4071-dbworkshops3
spill_prefix	athena-spill
rdbcatalog_connection_string	postgres://jdbc:postgresql://mod-bbcf

10. Click **Save**.

2.2 Finish setting up connection to Aurora PostgreSQL Data Source

In this step, we will finish setting up connection to Aurora PostgreSQL Data Source that we started in the previous step.

1. Go back to the previous Athena **Connect data sources** window in your browser and under the **Lambda functions** section, click the refresh button next to the **Lambda function** input.
2. Choose the Lambda function named **taxirdb** in the dropdown.



Connect data source

Step 1: Choose a data source
Step 2: Connection details

Connection details: PostgreSQL
Choose a Lambda function that is configured to connect to your data source, or create and configure a Lambda function to handle the connection. Learn more

Lambda function: Choose or configure a new AWS Lambda function to connect to the data source.
taxirdb
Configure new AWS Lambda function

Lambda function ARN: arn:aws:lambda:us-east-1:294908537736:function:taxirdb

Catalog name: Create a unique name to specify this data source within a SQL statement.
rdbcatalog
Use up to 127 characters and must be unique within your account. It cannot be changed after creation. Valid characters are a-z, A-Z, 0-9, _ (underscore), @ (at) and - (hyphen).

Description: catalog to query Aurora PG table via SQL
Use up to 1024 characters.

3. Click **Connect data source**.

Now, we are ready to query both DynamoDB and Aurora PostgreSQL using Athena federated query.

Previous Next