

Lambda Glue Architecture for Batch Processing Documentation

Table of Contents

| | |
|---|----|
| 1. Use case..... | 3 |
| 2. Project Description..... | 3 |
| 3. Assumption..... | 3 |
| 4. Architecture | 3 |
| 5. Complete Flow..... | 4 |
| 6. Lambda Glue Batch Build Steps..... | 5 |
| 7. Create S3 bucket | 5 |
| 8. Lambda configuration in Dynamo DB..... | 9 |
| 8.1. configuration table..... | 9 |
| 8.2. jobs table..... | 11 |
| 9. Create RDS Postgres database..... | 12 |
| 10. Create Roles..... | 15 |
| 10.1. Glue Role | 15 |
| 10.2. Lambda Role..... | 18 |
| 11. Setup Glue Job..... | 20 |
| 11.1. Create Classifier..... | 20 |
| 11.2. Create Crawler | 21 |
| 11.3. Create Table | 25 |
| 11.4. Create Connection | 29 |
| 11.5. Create Job | 31 |
| 12. Setup Cloud Watch Event Rule | 33 |
| 13. Build Lambdas..... | 35 |
| 14. Deploying Lambda..... | 37 |
| 14.1 sales-glue-trigger | 37 |
| 14.2 sales-glue-processed-trigger..... | 40 |
| 14.3 sales-inspect-ready-trigger | 42 |
| 15. Testing & Verification..... | 44 |
| 15.1 Setting up pgAdmin Tool..... | 44 |
| 15.2 Testing Scenario | 46 |
| 15.3 Expected Behaviour vs Actual Behaviour | 46 |
| 15.4 Testing In-Progress | 47 |
| 16. Performance Stats | 51 |
| 17. Troubleshooting & Learnings..... | 52 |

1. Use case

To migrate the data from on prem servers to AWS.

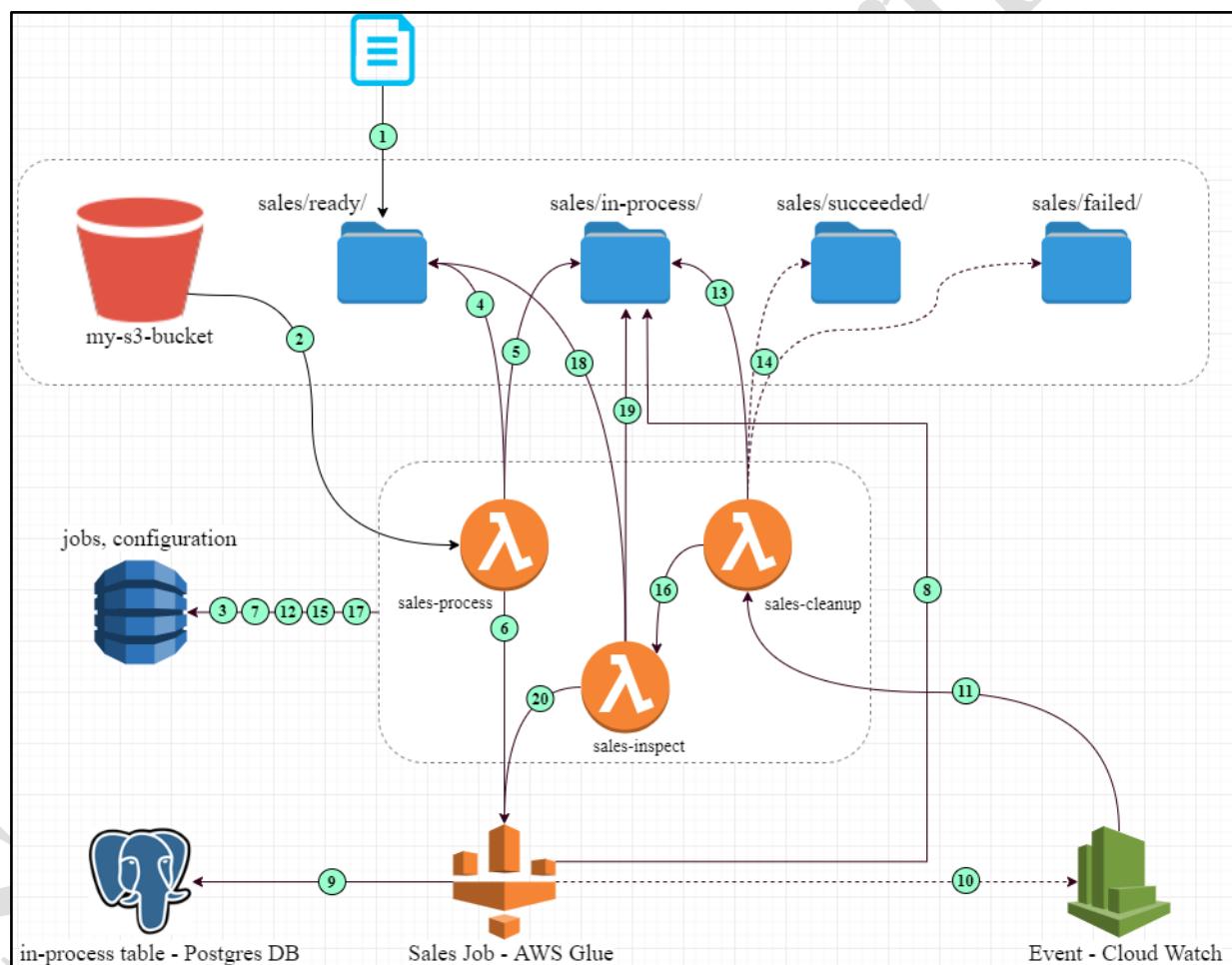
2. Project Description

Lambda Glue Architecture for batch processing is about copying the data from S3 bucket which is in CSV format to AWS RDS Postgres database. As the title self explains that this solution uses AWS Lambda and AWS Glue to persist data to database.

3. Assumption

- Data shall be extracted as a CSV file and is provided for processing.
- No validations implemented on the CSV file as extract is from source system.

4. Architecture



5. Complete Flow

- 1) The legacy system uploads the file to the S3 bucket.
- 2) S3 bucket triggers the notification to the sales-process lambda.

sales-process lambda will -

- 3) Use the configuration stored in dynamo DB which helps to process files.
- 4) Reads the files in ready directory.
- 5) When no files are found in in-process directory, all the files from ready directory are moved to in-process directory.
- 6) Triggers glue-job.
- 7) In-turn receives job run id, which is persisted to dynamo DB along with files picked up for processing & state of glue job.

Glue sales-job will -

- 8) Crawls the in-process directory for files.
- 9) Data in the CSV files of the in-process directory are persisted to the Postgres database.
- 10) While processing the data logs are written to the AWS cloud watch.
- 11) Cloud watch rule generates an event when glue state changes which in-turn triggers sales-cleanup lambda.

sales-cleanup lambda will -

- 12) Use the configuration stored in dynamo DB which helps to cleanup processed files.
- 13) Based on glue state change, if succeeded moves the processed files to succeeded directory.
- 14) Based on glue state change, if failed moves the processed files to failed directory.
- 15) Updates the state of job in dynamo db.
- 16) Triggers a call to sales-inspect lambda (lambda invoking lambda)

sales-inspect lambda will -

- 17) Use the configuration stored in dynamo DB which helps to process any newly added files in ready directory.
- 18) Reads the files in ready directory.
- 19) When no files are found in in-process directory, all the files from ready directory are moved to in-process directory.
- 20) Triggers glue-job & 8,9,10.... steps repeat as long as there are files found in ready directory.

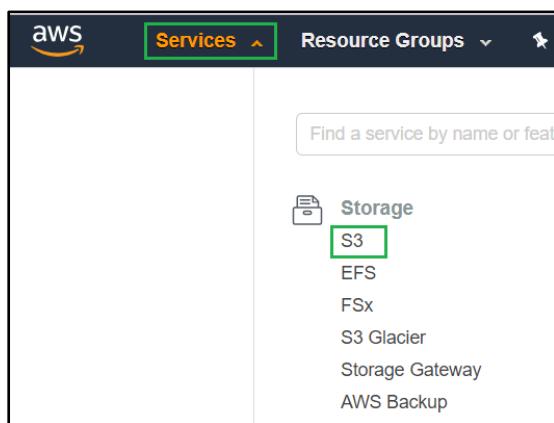
6. Lambda Glue Batch Build Steps

To run the application, follow below steps.

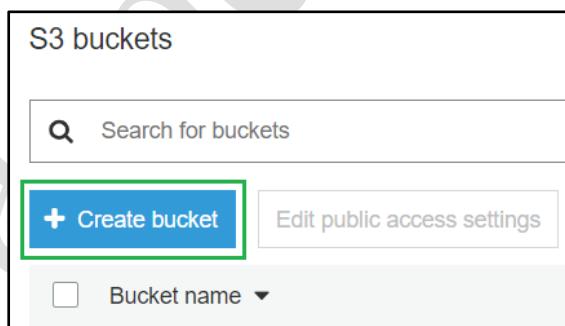
- 1) [Create S3 bucket](#)
- 2) [Lambda configuration in Dynamo DB](#)
- 3) [Create RDS Postgres database](#)
- 4) [Create Roles](#)
- 5) [Setup Glue Job](#)
- 6) [Setup Cloud Watch Event Rule](#)
- 7) [Deploying Lambda's](#)
- 8) [Testing](#)
- 9) [Performance Statistics](#)
- 10) [Troubleshooting & Learnings](#)

7. Create S3 bucket

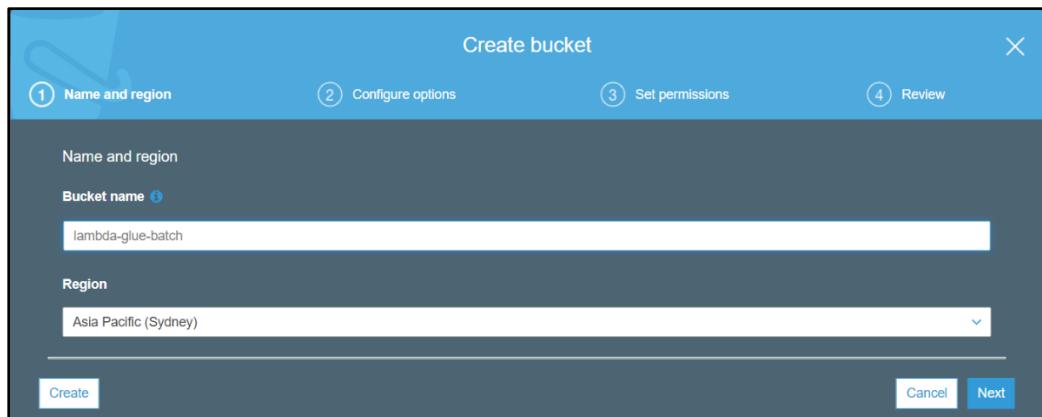
- 1) Click on Services → Storage → S3



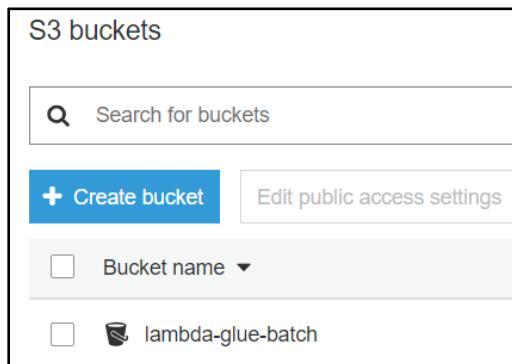
- 2) Click on create bucket



- 3) Provide a bucket name



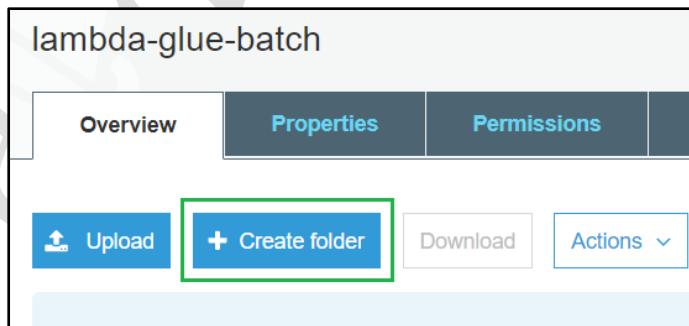
- 4) Bucket gets created as shown below.



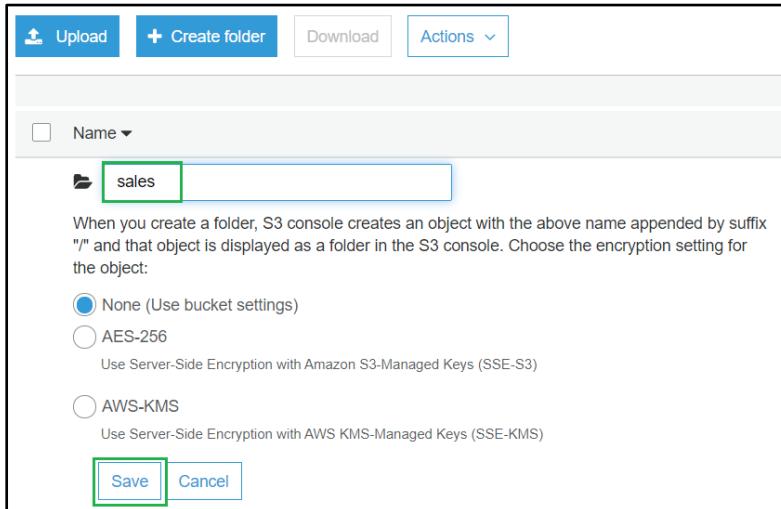
- 5) Inside the bucket create the below directory structure

- sales/ready/
- sales/in-process/
- sales/succeeded/
- sales/failed/

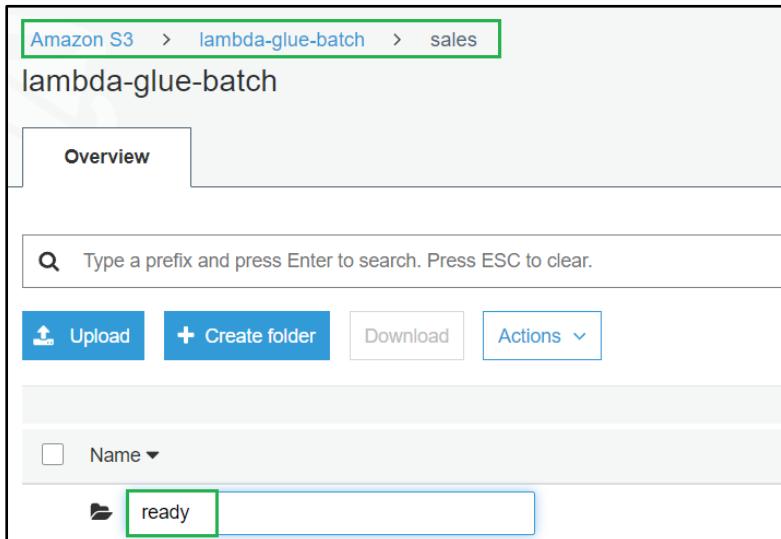
- 6) To create the above structure, click on create folder



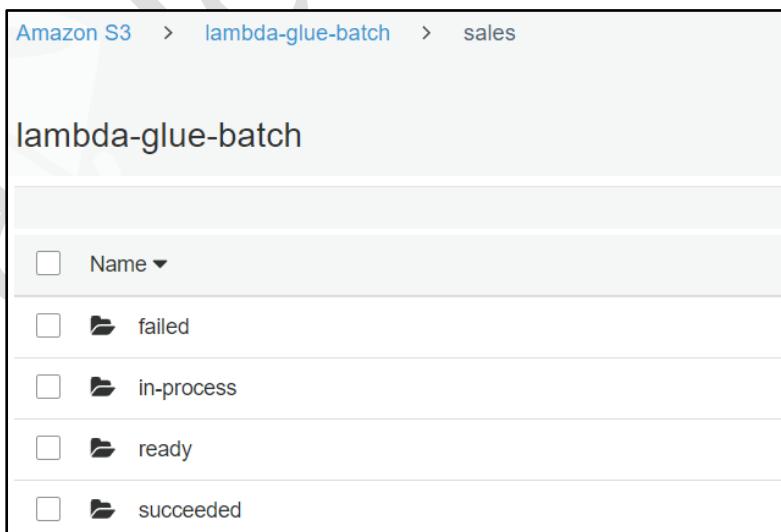
- 7) Provide “sales” as directory path



- 8) Create “**ready**” directory inside sales directory



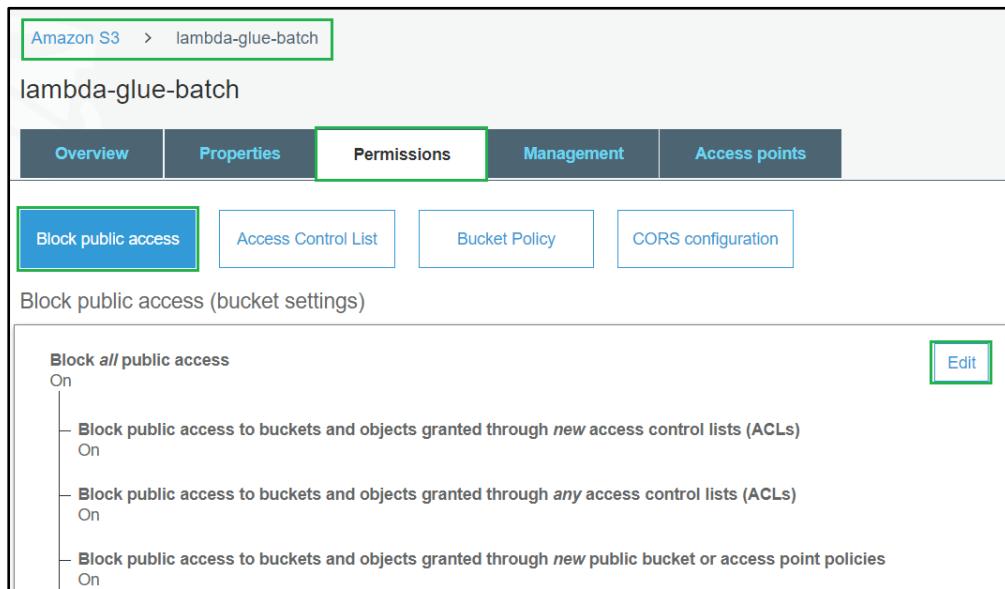
- 9) Similarly, repeat step 7 and 8 to create in-process, succeeded, failed directories.



- 10) To allow glue to crawl the bucket, provide necessary access from permissions tab as shown below.

Note: This setting is for demo purpose. Do not use this setting in production.

Go to S3→lambda-glue-batch→Permissions→Block public access→Edit



Amazon S3 > lambda-glue-batch

lambda-glue-batch

Overview Properties **Permissions** Management Access points

Block public access Access Control List Bucket Policy CORS configuration

Block public access (bucket settings)

Block all public access

On

- Block public access to buckets and objects granted through *new* access control lists (ACLs)
- On
- Block public access to buckets and objects granted through *any* access control lists (ACLs)
- On
- Block public access to buckets and objects granted through *new* public bucket or access point policies
- On

Edit

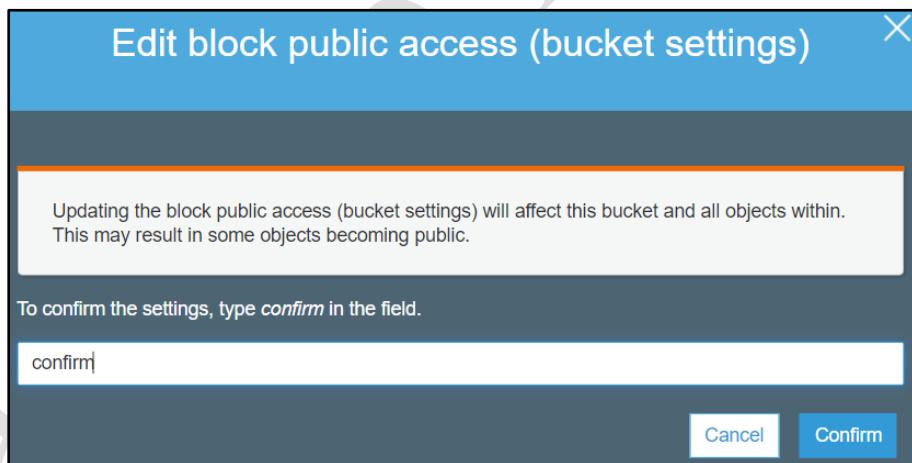
11) Uncheck Block all public access and click save



Block all public access
Turning this setting on is the same

Block all public access
Turning this setting on is the same

12) Confirm the unblock



Edit block public access (bucket settings) X

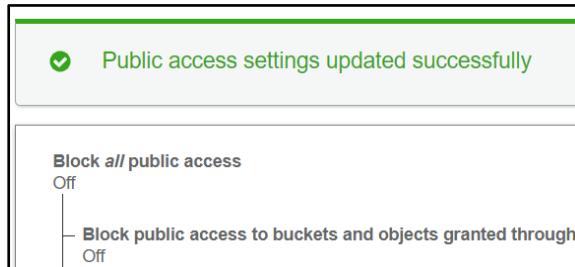
Updating the block public access (bucket settings) will affect this bucket and all objects within. This may result in some objects becoming public.

To confirm the settings, type *confirm* in the field.

confirm

Cancel Confirm

13) Settings will be updated



Public access settings updated successfully

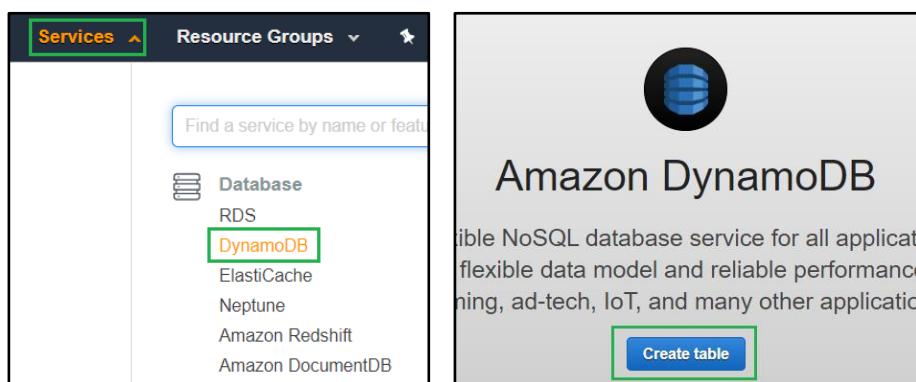
Block all public access

Off

- Block public access to buckets and objects granted through Off

8. Lambda configuration in Dynamo DB

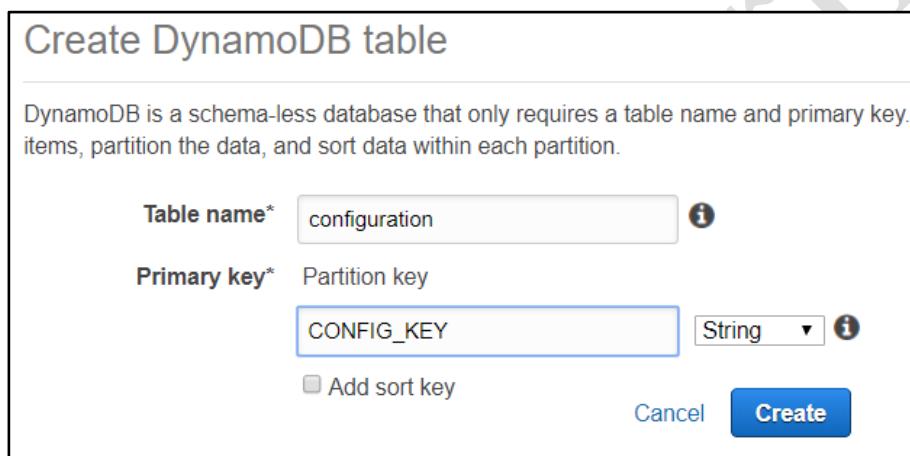
- 1) Go to Services → Database → DynamoDB, which allows to Create table



The image shows the AWS Services console. In the 'Database' section, the 'DynamoDB' service is highlighted with a green box. To the right, the 'Amazon DynamoDB' service page is displayed, featuring a large blue globe icon, the text 'Amazon DynamoDB', and a 'Create table' button, which is also highlighted with a green box.

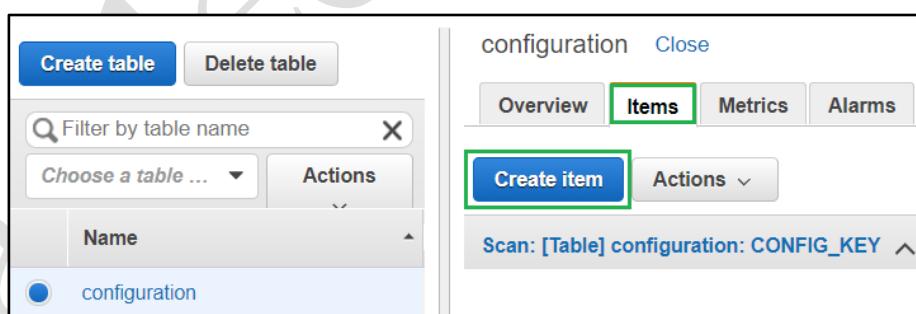
8.1. configuration table

- 2) Provide the Table Name and Primary Key Column with Data Type.

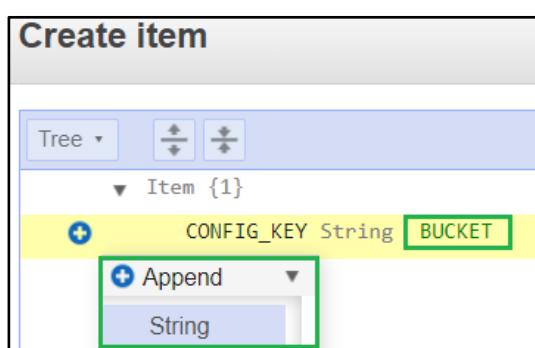


The image shows the 'Create DynamoDB table' dialog. It includes fields for 'Table name*' (set to 'configuration'), 'Primary key*' (set to 'Partition key' with value 'CONFIG_KEY' and type 'String'), and an 'Add sort key' checkbox. At the bottom are 'Cancel' and 'Create' buttons, with the 'Create' button highlighted with a green box.

- 3) Table is created, now create item.



The image shows the 'configuration' table details page. The 'Items' tab is selected. The 'Create item' button is highlighted with a green box. The table list on the left shows a single item named 'configuration'.



The image shows the 'Create item' dialog for the 'configuration' table. It displays a hierarchical tree structure with an item labeled 'Item {1}'. Underneath, there is a table with a row for 'CONFIG_KEY' (String type) and 'BUCKET'. The 'Append' button is highlighted with a green box. The 'CONFIG_KEY' and 'BUCKET' fields are also highlighted with green boxes.

- 4) Add CONFIG_KEY & CONFIG_VALUE as shown below and save it.

Create item

Tree ▾ 

▼ Item {2}

 CONFIG_KEY String : BUCKET
 CONFIG_VALUE String : lambda-glue-batch

Cancel **Save**

- 5) Add below configuration values –

| CONFIG_KEY | CONFIG_VALUE |
|---------------------|-----------------------------|
| BUCKET | lambda-glue-batch |
| READY-DIR-PATH | sales/ready/ |
| IN-PROCESS-DIR-PATH | sales/in-process/ |
| SUCCEEDED-DIR-PATH | sales/succeeded/ |
| FAILED-DIR-PATH | sales/failed/ |
| JOB | SalesJob |
| LAMBDA | sales-inspect-ready-trigger |

- 6) Repeat Steps 3 & 4 to add all the above configuration. Configuration table would look like below.

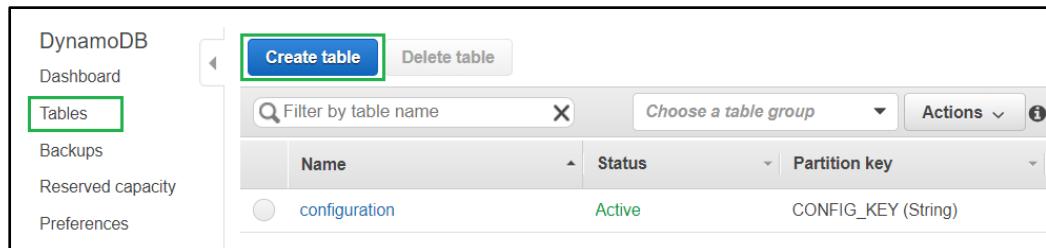
Create item **Actions ▾**

Scan: [Table] configuration: CONFIG_KEY ▾

| <input type="checkbox"/> | CONFIG_KEY  | CONFIG_VALUE |
|--------------------------|--|-----------------------------|
| <input type="checkbox"/> | BUCKET | lambda-glue-batch |
| <input type="checkbox"/> | FAILED-DIR-PATH | sales/failed/ |
| <input type="checkbox"/> | IN-PROCESS-DIR-PATH | sales/in-process/ |
| <input type="checkbox"/> | JOB | SalesJob |
| <input type="checkbox"/> | LAMBDA | sales-inspect-ready-trigger |
| <input type="checkbox"/> | READY-DIR-PATH | sales/ready/ |
| <input type="checkbox"/> | SUCCEEDED-DIR-PATH | sales/succeeded/ |

8.2. jobs table

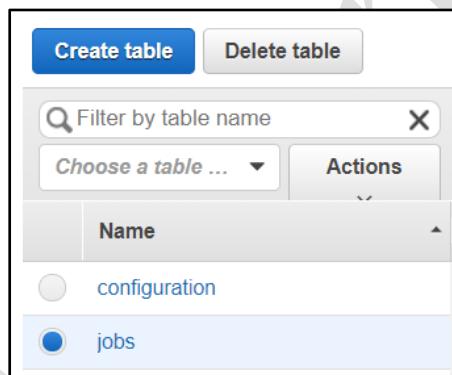
- 7) Click on Create table as shown below



- 8) Provide table name and primary key as JOB_ID and click Create.

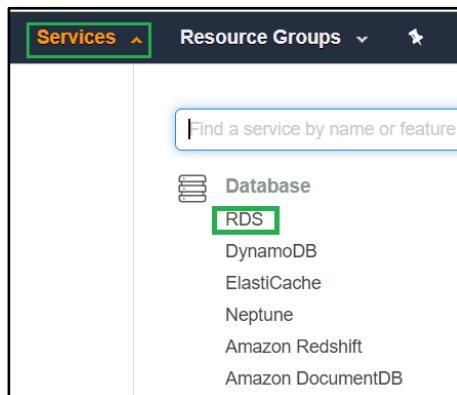
A screenshot of the 'Create DynamoDB table' dialog. It shows the table name 'jobs' and the primary key 'JOB_ID' as a partition key of type 'String'. There's a checkbox for 'Add sort key' which is unchecked. At the bottom are 'Cancel' and 'Create' buttons, with 'Create' being highlighted.

- 9) Jobs table would be created as shown below.

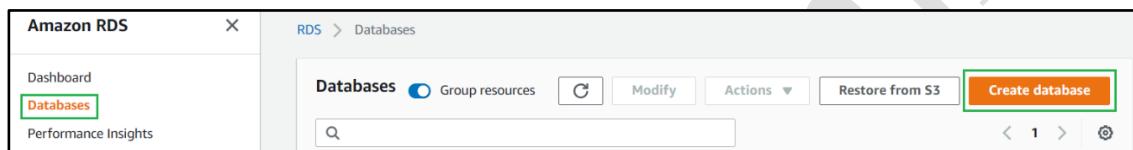


9. Create RDS Postgres database

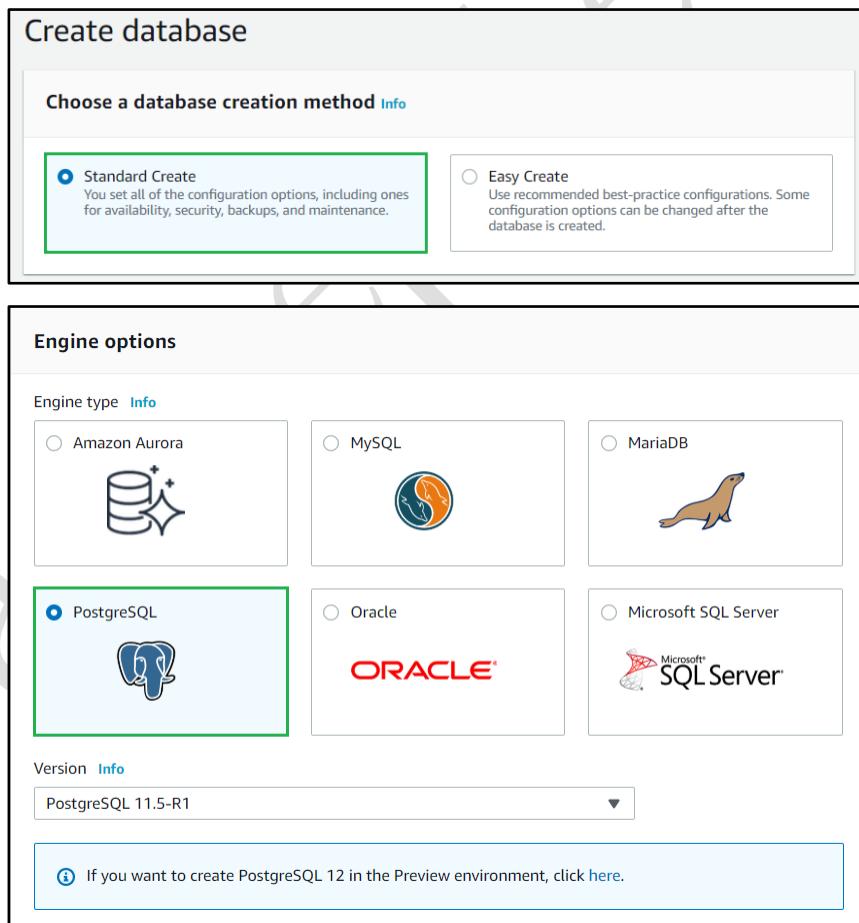
- 1) Click on Services → Analytics → AWS Glue



- 2) Go to databases and create database



- 3) Choose Standard Create, PostgreSQL with the default version



- 4) Choose Free tier from Templates section



- 5) Fill the settings section as shown

The screenshot shows the 'Settings' section with the following configuration:

- DB instance identifier: **lambda-glue-batch** (highlighted with a green border)
- Master username: **postgres** (highlighted with a green border)
- Master password: **.....** (highlighted with a green border)

- 6) From connectivity section, do the below setting as in order to make the DB accessible on your PC.

Note: Do not do this setting for your production DB's. This is just for Demo.

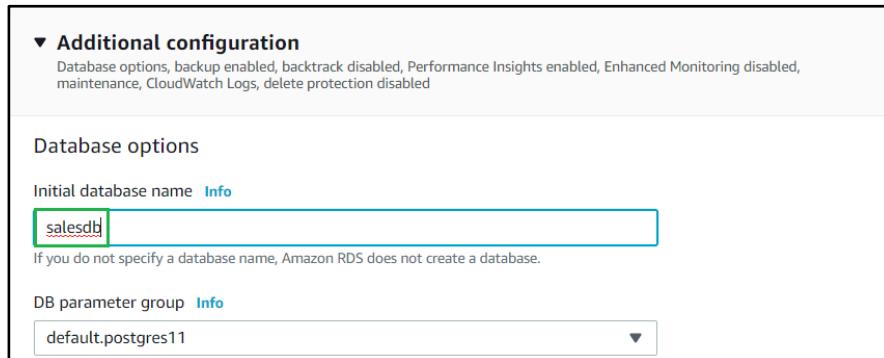
The screenshot shows the 'Connectivity' section with the following setting:

- Publicly accessible: **Yes** (highlighted with a green border)

Details for 'Yes': Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

Details for 'No': RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

- 7) In the Additional configuration section, Provide Database name in Database options section and leave rest to default values.



▼ Additional configuration
Database options, backup enabled, backtrack disabled, Performance Insights enabled, Enhanced Monitoring disabled, maintenance, CloudWatch Logs, delete protection disabled

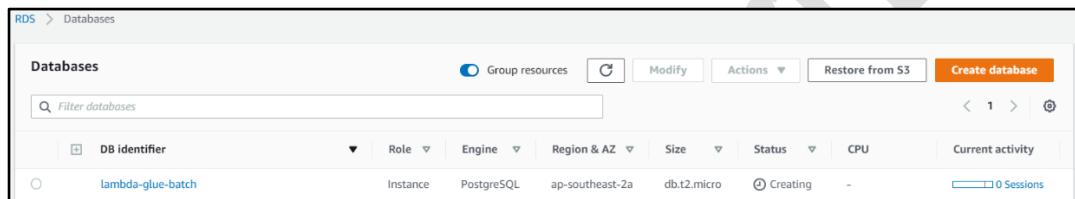
Database options

Initial database name [Info](#)
salesdb

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group [Info](#)
default.postgres11

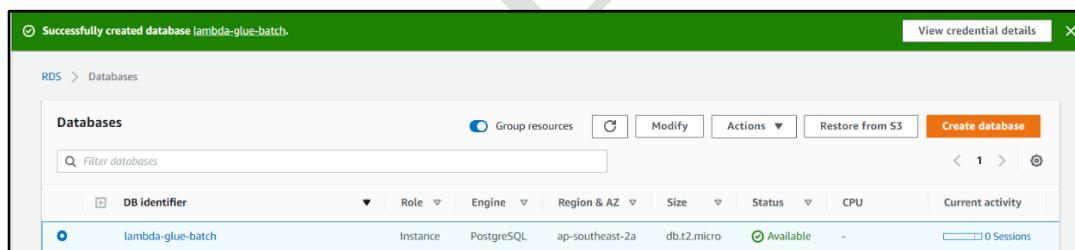
- 8) It takes few minutes to create the database and shows status as **creating**



RDS > Databases

| DB identifier | Role | Engine | Region & AZ | Size | Status | CPU | Current activity |
|-------------------|----------|------------|-----------------|-------------|----------|-----|------------------|
| lambda-glue-batch | Instance | PostgreSQL | ap-southeast-2a | db.t2.micro | Creating | - | 0 Sessions |

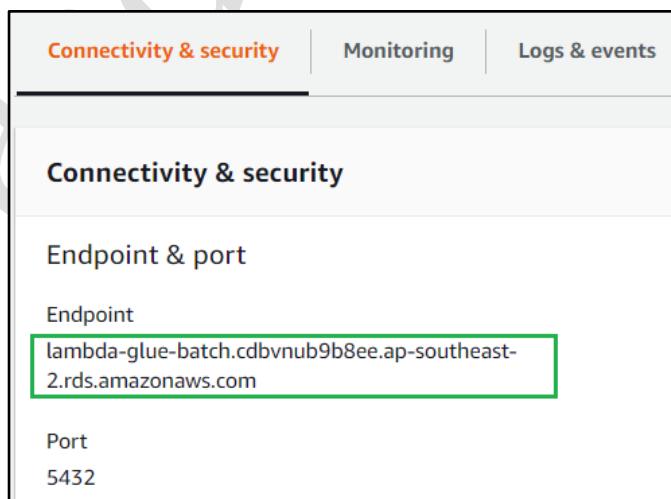
- 9) Once the database is created it shows that database is available. Click on DB Identifier.



RDS > Databases

| DB identifier | Role | Engine | Region & AZ | Size | Status | CPU | Current activity |
|-------------------|----------|------------|-----------------|-------------|-----------|-----|------------------|
| lambda-glue-batch | Instance | PostgreSQL | ap-southeast-2a | db.t2.micro | Available | - | 0 Sessions |

- 10) Check for the database endpoint.



Connectivity & security | Monitoring | Logs & events

Connectivity & security

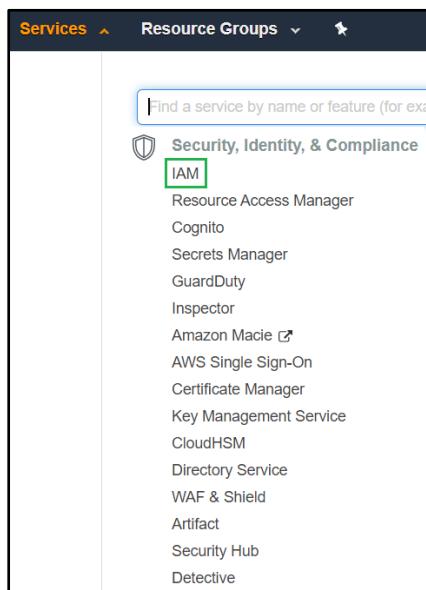
Endpoint & port

Endpoint
lambda-glue-batch.cdbvnub9b8ee.ap-southeast-2.rds.amazonaws.com

Port
5432

10. Create Roles

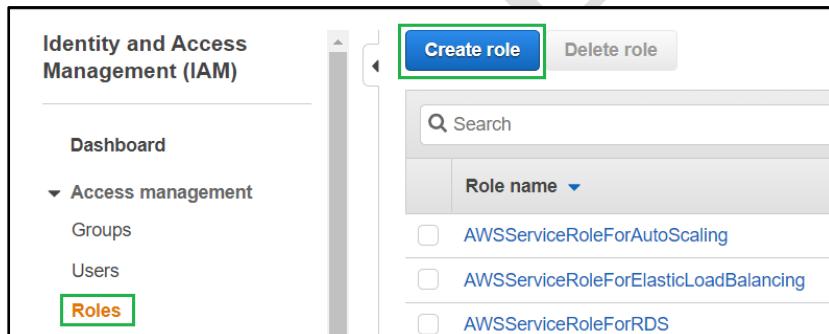
- 1) Select IAM found at: Services → Security, Identity, & Compliance → IAM



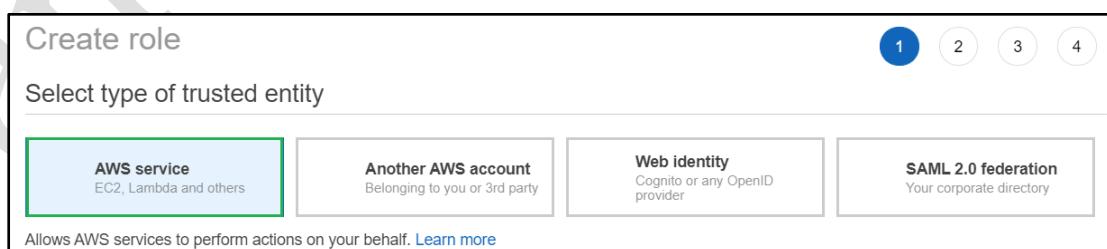
10.1. Glue Role

- 2) As per the Architecture diagram, Glue needs access to **S3**, **RDS** and **Glue Service Role**.

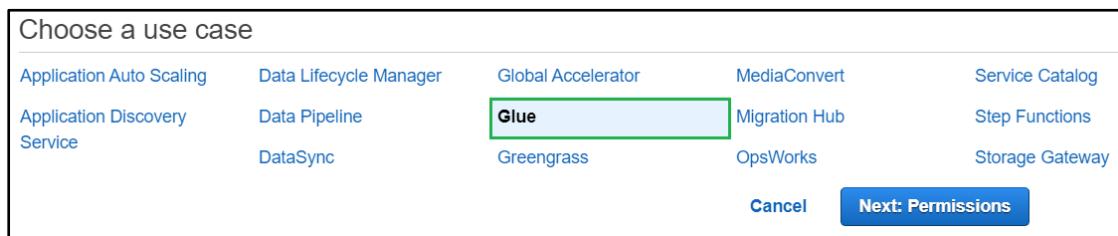
Click on Roles on the left pane and click Create role



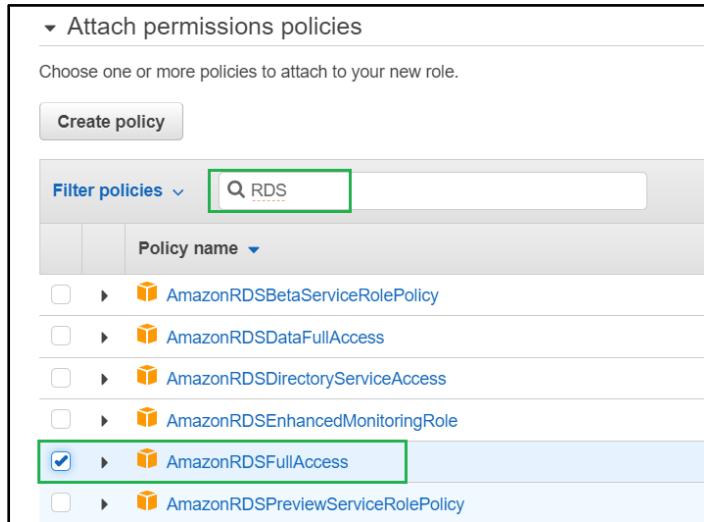
- 3) Select AWS service as shown below.



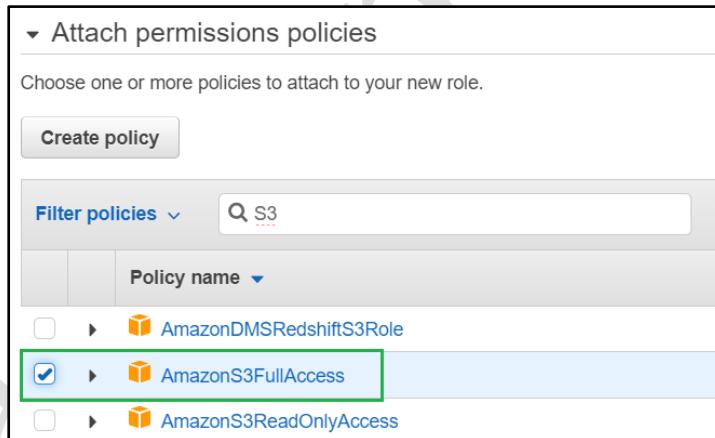
4) Choose Glue use case



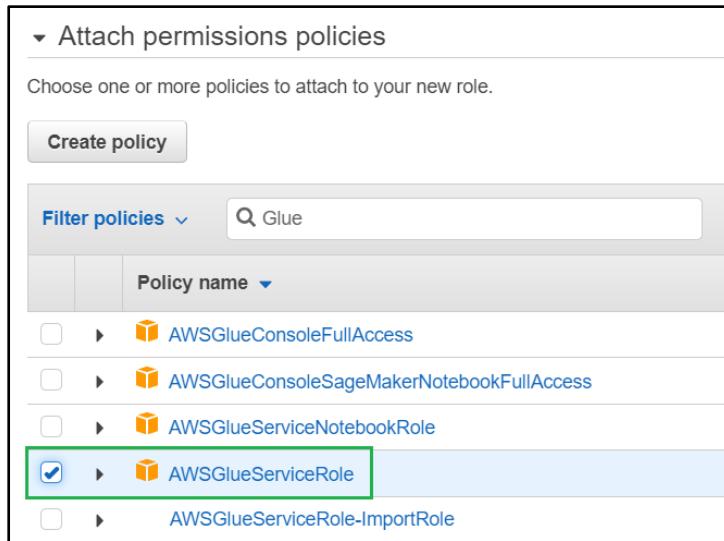
5) Search & Attach AmazonRDSFullAccess Policy



6) Search & Attach AmazonS3FullAccess Policy



- 7) Search & Attach AWSGlueServiceRole Policy, Click next and Review.



▼ Attach permissions policies

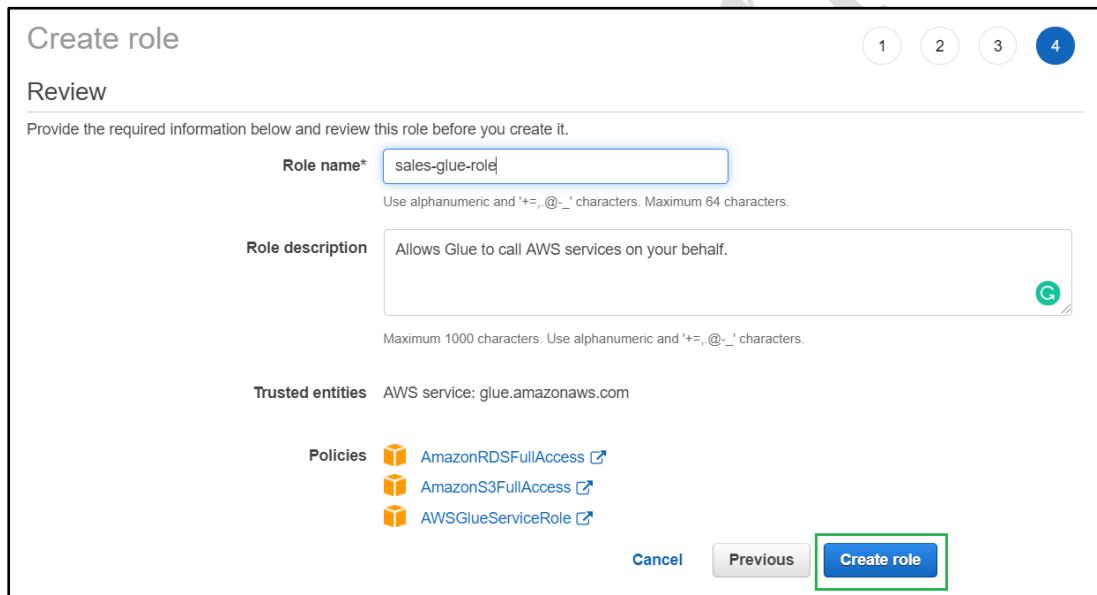
Choose one or more policies to attach to your new role.

Create policy

Filter policies ▾

| | Policy name ▾ |
|-------------------------------------|---|
| <input type="checkbox"/> | AWSGlueConsoleFullAccess |
| <input type="checkbox"/> | AWSGlueConsoleSageMakerNotebookFullAccess |
| <input type="checkbox"/> | AWSGlueServiceNotebookRole |
| <input checked="" type="checkbox"/> | AWSGlueServiceRole |
| <input type="checkbox"/> | AWSGlueServiceRole-ImportRole |

- 8) Give role name – sales-glue-role and click create role



Create role

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 Allows Glue to call AWS services on your behalf.

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

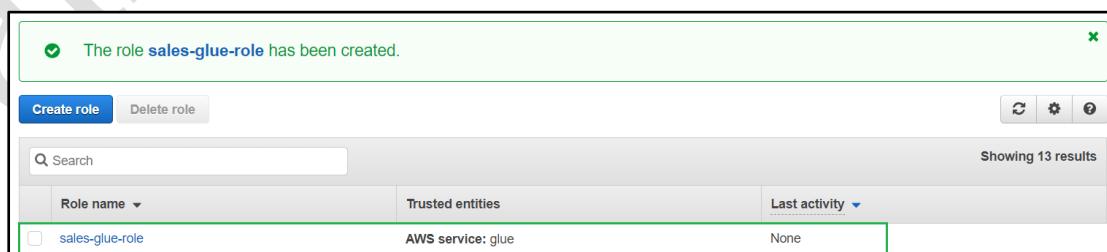
Trusted entities AWS service: glue.amazonaws.com

Policies

- AmazonRDSFullAccess
- AmazonS3FullAccess
- AWSGlueServiceRole

Create role

- 9) It creates and lists the role as shown below.



Create role **Delete role**

Search

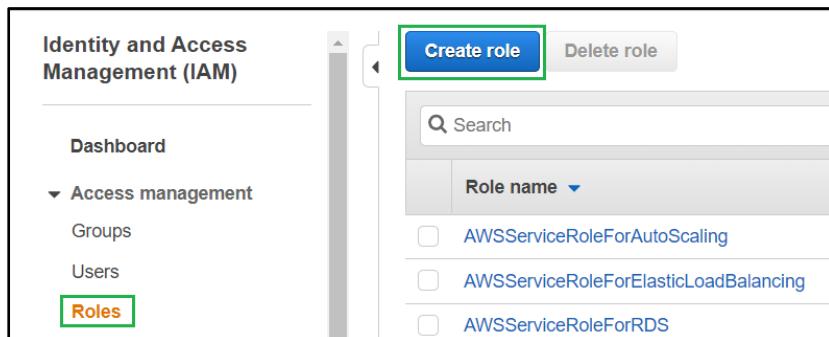
Showing 13 results

| Role name ▾ | Trusted entities | Last activity ▾ |
|--|-------------------|-----------------|
| <input type="checkbox"/> sales-glue-role | AWS service: glue | None |

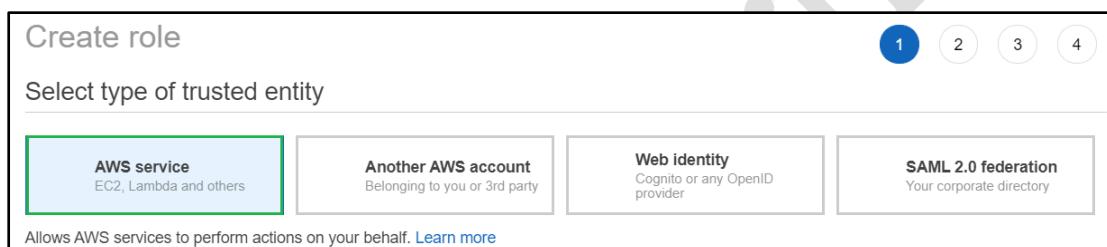
10.2. Lambda Role

10) Lambda needs access to invoke Glue, Lambda. Hence add Policies AWSLambdaFullAccess & AWSGlueServiceRole

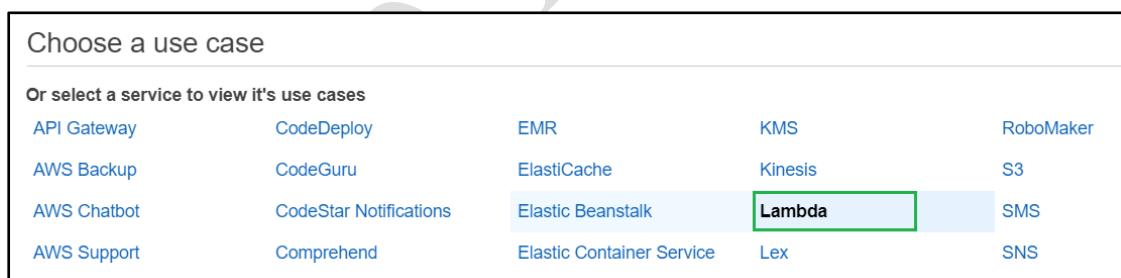
11) Create Role



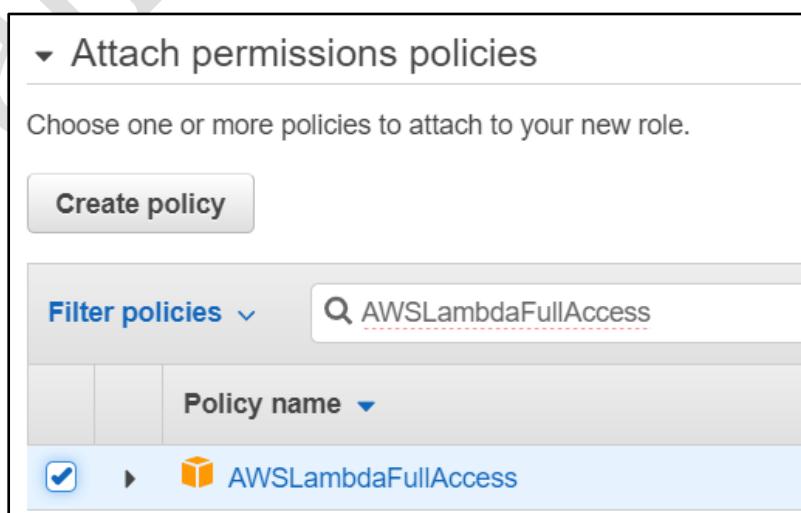
12) Select AWS service as shown below.



13) Choose Lambda use case



14) Search & attach AWSLambdaFullAccess & AWSGlueServiceRole Policies



▼ Attach permissions policies

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

Create policy

Filter policies

| | Policy name |
|-------------------------------------|--|
| <input checked="" type="checkbox"/> |  AWSGlueServiceRole |

15) Give role name – sales-lambda-role and click create role

Create role

Review

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

Role name* sales-lambda-role

Use alphanumeric and '+=_,@_-' characters. Maximum 64 characters.

Role description Allows Lambda functions to call AWS services on your behalf.

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

Trusted entities AWS service: lambda.amazonaws.com

Policies  AWSLambdaFullAccess  AWSGlueServiceRole

Create role

16) It creates role and lists the role as shown below.

The role sales-lambda-role has been created.

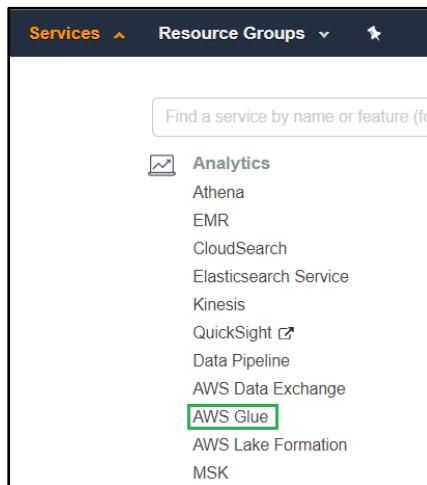
Create role **Delete role**

Search

| Role name | Trusted entities |
|--|---------------------|
| <input type="checkbox"/> sales-lambda-role | AWS service: lambda |
| <input type="checkbox"/> sales-glue-role | AWS service: glue |

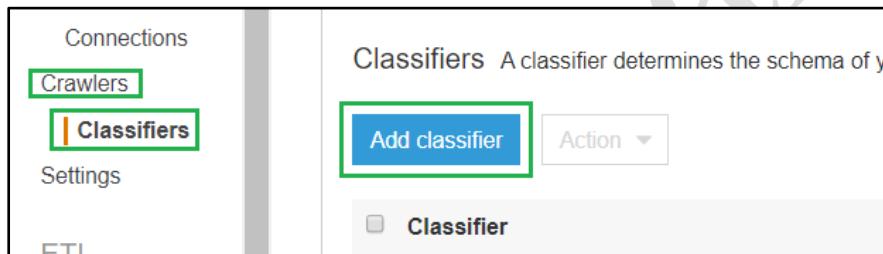
11. Setup Glue Job

- 1) Select AWS Glue Found at: Services → Analytics → AWS Glue



11.1. Create Classifier

- 2) Go to Classifiers under Crawlers Section and click Add Classifier



- 3) It opens up a pop-up, fill the data as shown.

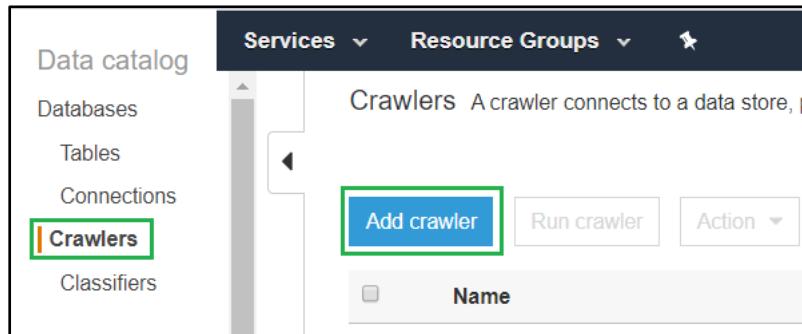
A screenshot of the 'Add classifier' pop-up form. It includes fields for 'Classifier name' (SalesClassifier), 'Classifier type' (CSV selected), 'Column delimiter' (Comma (,)), 'Quote symbol' (Double-quote (")), 'Column headings' (Has headings), and a 'Region,Country,Item Type,Sales Channel,Order Priority,Order Date' input field. Under 'Processing options', there is a checkbox for 'Allow files with single column'. A 'Create' button is at the bottom.

Headings specified is taken from my sample data used. Below is the heading data for reference.

Region,Country,Item Type,Sales Channel,Order Priority,Order Date,Order ID,Ship Date,Units Sold,Unit Price,Unit Cost,Total Revenue,Total Cost,Total Profit

11.2. Create Crawler

- 4) Now click on Crawlers → Add Crawler



- 5) Provide Crawler information

Add crawler

Add information about your crawler

Crawler info

Crawler source type

Data store

IAM Role

Schedule

Output

Review all steps

Crawler name

▶ Tags, description, security configuration, and classifiers (optional)

Next

Add crawler

Crawler info

SalesCrawler

Crawler source type

Data store

IAM Role

Schedule

Output

Review all steps

Specify crawler source type

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

Crawler source type

Data stores

Existing catalog tables

Back **Next**

Add crawler

Add a data store

Choose a data store

S3

Crawl data in

Specified path in my account

Specified path in another account

Include path

s3://bucket/prefix/object

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

▶ Exclude patterns (optional)

Back **Next**

Click on the folder shown on include path, it opens up a pop-up screen, select the created bucket.

Choose S3 path

S3
lambda-glue-batch

Select

Add crawler

Crawler info
SalesCrawler
Crawler source type
Data stores
Data store
IAM Role
Schedule
Output
Review all steps

Add a data store

Choose a data store
S3

Crawl data in
 Specified path in my account
 Specified path in another account

Include path
s3://lambda-glue-batch

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

Exclude patterns (optional)

Back Next

Add crawler

Crawler info
SalesCrawler
Crawler source type
Data stores
Data store
S3: s3://lambda-glu...
IAM Role
Schedule
Output
Review all steps

Add another data store

Yes
 No

Back Next

- 6) Choose the role which was created in earlier section – “sales-glue-role”

Add crawler

Crawler info
SalesCrawler
Crawler source type
Data stores
Data store
S3: s3://lambda-glu...
IAM Role
Schedule
Output
Review all steps

Choose an IAM role

The IAM role allows the crawler to run and access your Amazon S3 data stores. [Learn more](#)

Update a policy in an IAM role
 Choose an existing IAM role
 Create an IAM role

IAM role [?](#)
sales-glue-role

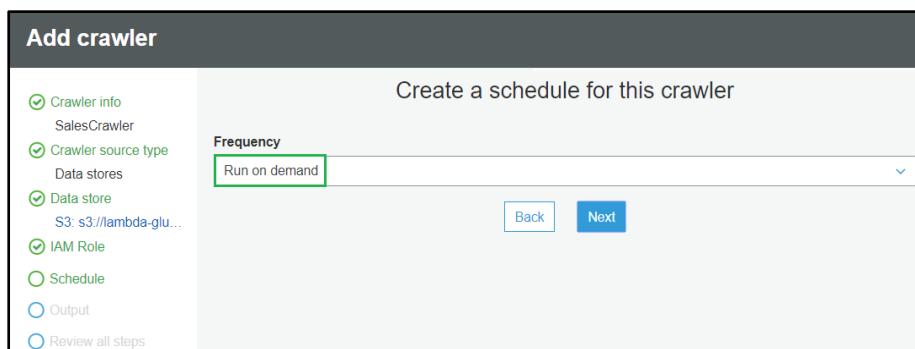
This role must provide permissions similar to the AWS managed policy, **AWSGlueServiceRole**, plus access to your data stores.

• s3://lambda-glue-batch

You can also create an IAM role on the [IAM console](#).

Back Next

- 7) Set frequency to run on demand. As per the architecture lambda invokes when required.



Add crawler

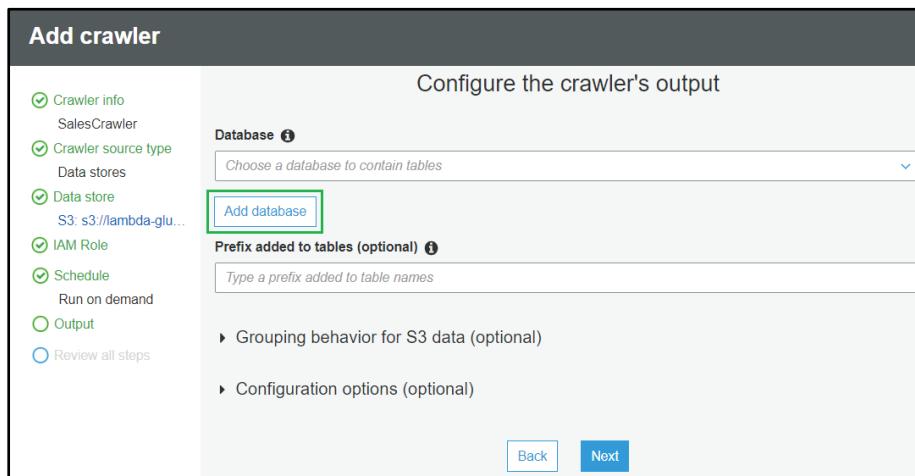
Create a schedule for this crawler

Frequency: Run on demand

Back Next

Crawler info: SalesCrawler
 Crawler source type: Data stores
 Data store: S3: s3://lambda-glu...
 IAM Role
 Schedule: Run on demand
 Output
 Review all steps

- 8) Click next and Add database



Add crawler

Configure the crawler's output

Database: Choose a database to contain tables

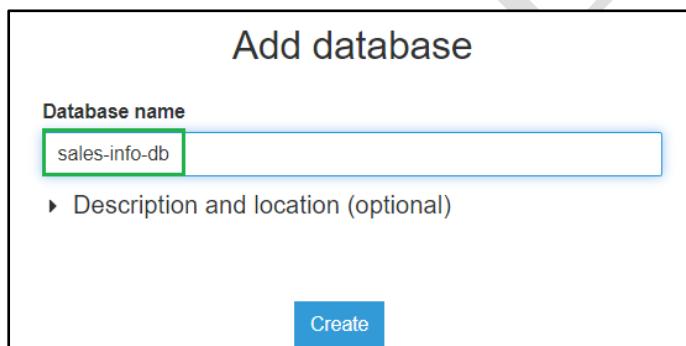
Prefix added to tables (optional): Type a prefix added to table names

Grouping behavior for S3 data (optional)
 Configuration options (optional)

Back Next

Crawler info: SalesCrawler
 Crawler source type: Data stores
 Data store: S3: s3://lambda-glu...
 IAM Role
 Schedule: Run on demand
 Output
 Review all steps

- 9) It opens up a pop-up, provide database name (Not the one created for RDS)



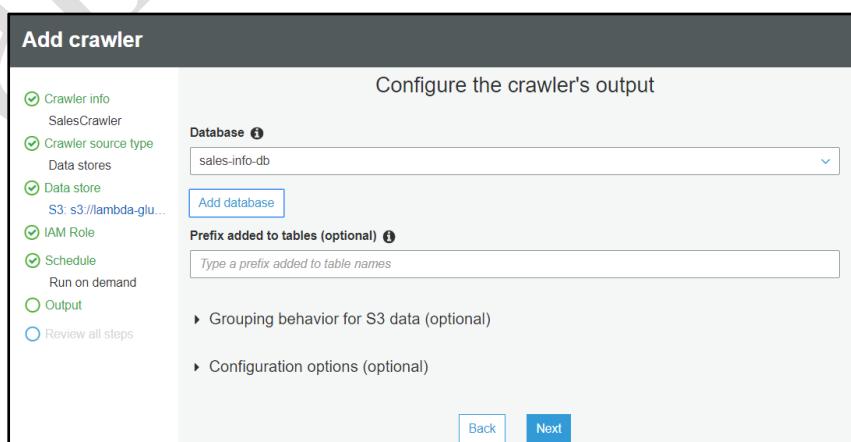
Add database

Database name: sales-info-db

Description and location (optional)

Create

- 10) Click next



Add crawler

Configure the crawler's output

Database: sales-info-db

Prefix added to tables (optional): Type a prefix added to table names

Grouping behavior for S3 data (optional)
 Configuration options (optional)

Back Next

Crawler info: SalesCrawler
 Crawler source type: Data stores
 Data store: S3: s3://lambda-glu...
 IAM Role
 Schedule: Run on demand
 Output
 Review all steps

11) Review details and Click Finish.

Add crawler

Crawler info
SalesCrawler

Crawler source type
Data stores

Data store
S3: s3://lambda-glu...

IAM Role
arn:aws:iam:::role/sales-glue-role

Schedule
Run on demand

Output
sales-info-db

Review all steps

Crawler info

Name: SalesCrawler
Tags: -

IAM role

IAM role: arn:aws:iam:::role/sales-glue-role

Schedule

Schedule: Run on demand

Output

Database: sales-info-db
Prefix added to tables (optional):
Create a single schema for each S3 path: false
Configuration options

[Back](#) [Finish](#)

12) Sales Crawler created as follows.

Crawler **SalesCrawler** was created to run on demand. [Run it now?](#)

[Add crawler](#) [Run crawler](#) [Action ▾](#) [Filter by tags and attributes](#)

| <input type="checkbox"/> | Name | Schedule | Status |
|--------------------------|--------------|----------|--------|
| <input type="checkbox"/> | SalesCrawler | | Ready |

13) Database which is created in Step 9 could be seen on the Databases tab as shown below.

AWS Glue

Data catalog

Databases

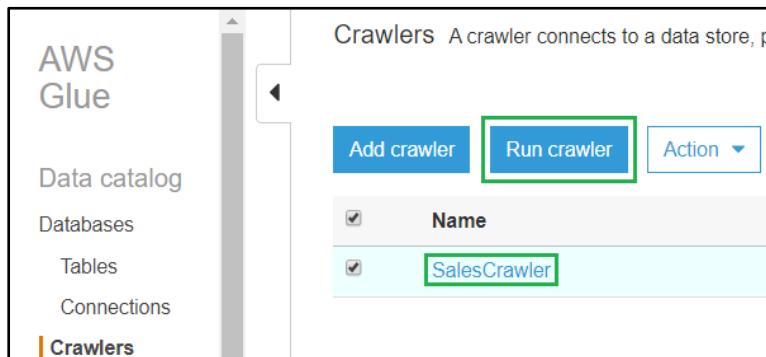
Tables

Databases A database is a set of associated table

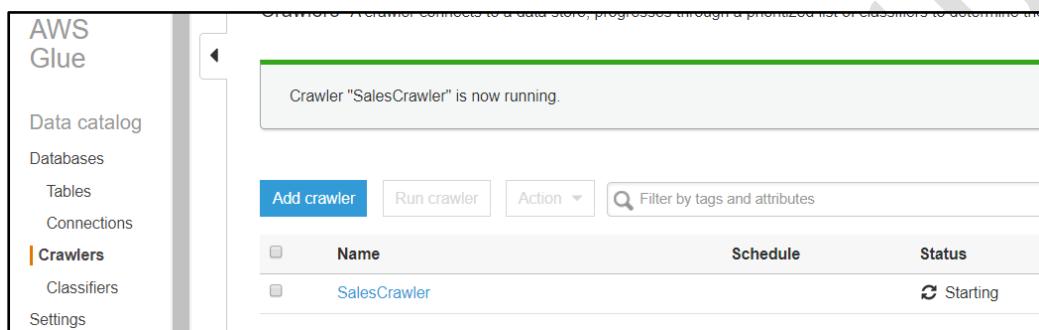
[Add database](#) [View tables](#) [Action ▾](#)

| <input type="checkbox"/> | Name |
|--------------------------|---------------|
| <input type="checkbox"/> | sales-info-db |

- 14) To create the table for sales-info-db, either run the created crawler or create table manually. In this demo, lets run the crawler.



- 15) Crawler starts running

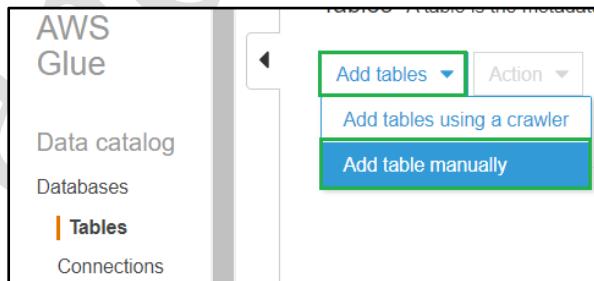


- 16) Crawler ran for 47 Secs and Stopped.

| Name | Schedule | Status | Logs | Last runtime | Median runtime |
|--------------|----------|----------|------|--------------|----------------|
| SalesCrawler | | Stopping | | 47 secs | 47 secs |

11.3. Create Table

- 17) To add tables manually, click on tables on left tab, Add tables → Add table manually



- 18) Setup table properties

Note: Table name is provided as in-process as crawler will read the data from in-process folder of s3 bucket.

Add table

Set up your table's properties

Table properties

Data store

Data format

Schema

Review

Table name: in-process

Database:

Description (optional)

19) Provide S3 directory to crawl, click on folder, it opens a pop up

Add table

Table properties
Name: in-process
Database: sales-info-db

Data store

Data format

Schema

Review

Add a data store

Data is located in

Specified path in my account

Specified path in another account

Include path:

Path must be in the form s3://bucket/prefix/. It must end with a slash.

20) Select the directory to crawl

Choose S3 path

S3

- lambda-glue-batch
- sales
 - failed
 - in-process
 - ready
 - succeeded

Add table

Table properties
Name: in-process
Database: sales-info-db

Data store

Data format

Schema

Review

Add a data store

Data is located in

Specified path in my account

Specified path in another account

Include path:

Path must be in the form s3://bucket/prefix/. It must end with a slash.

21) Choose data format

Add table

Table properties
Name: in-process
Database: sales-info-db

Data store
s3://lambda-glue-batch/sales/in-process/

Data format
 Schema
 Review

Choose a data format

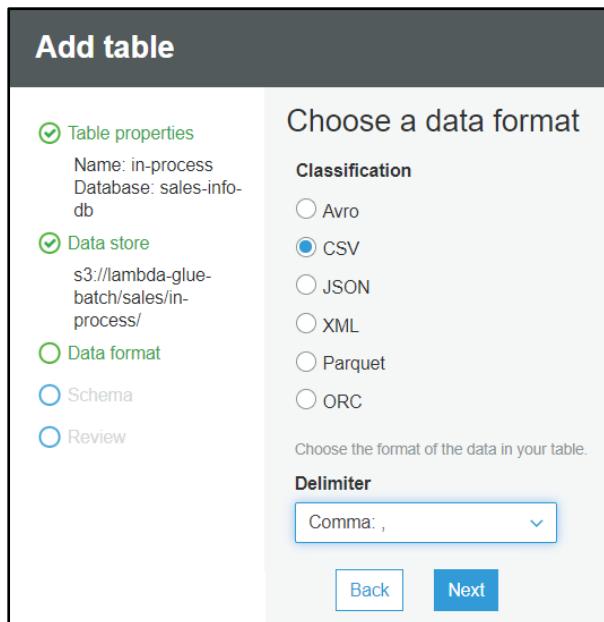
Classification

Avro
 CSV
 JSON
 XML
 Parquet
 ORC

Choose the format of the data in your table.

Delimiter
Comma: ,

Back **Next**



22) Define schema, click on Add column.

Add table

Table properties
Name: in-process
Database: sales-info-db

Data store
s3://lambda-glue-batch/sales/in-process/

Data format
CSV

Schema
 Review

Define a schema

Add column

Column name

Add column

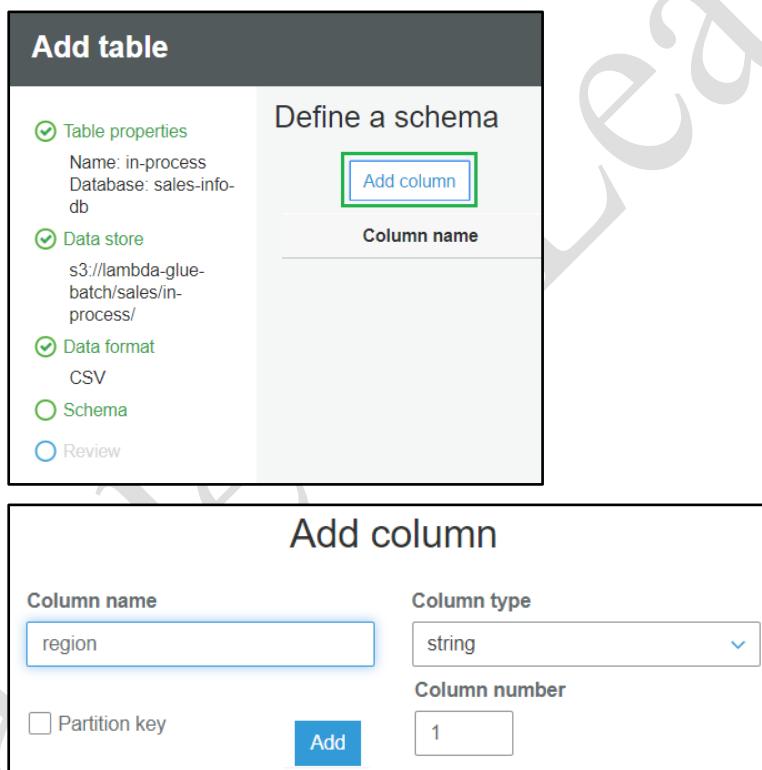
Column name region

Column type string

Partition key

Column number 1

Add



Similarly, add all the columns.

Add table

Table properties

Name: in-process
Database: sales-info-db

Data store

s3://lambda-glue-batch/sales/in-process/

Data format

CSV

Schema

Review

Define a schema

[Add column](#)

| | Column name | Data type |
|---|----------------|-----------|
| 1 | region | string |
| 2 | country | string |
| 3 | item type | string |
| 4 | sales channel | string |
| 5 | order priority | string |
| 6 | order date | string |
| 7 | order id | bigint |
| 8 | | |

23) Click next → Review → Finish

Add table

Table properties

Name: in-process
Database: sales-info-db

Data store

s3://lambda-glue-batch/sales/in-process/

Data format

CSV

Schema

Review

Table properties

Name: in-process
Database: sales-info-db
Description:

Data store

Type: S3
Location: s3://lambda-glue-batch/sales/in-process/

Data format

Classification: CSV

Schema

Columns: region, country, item type, sales channel, order priority, order date, order id, ship date, units sold, unit price, unit cost, total revenue, total cost, total profit
Partition keys:

[Back](#) [Finish](#)

24) Table is created.

AWS Glue

Data catalog

Databases

Tables

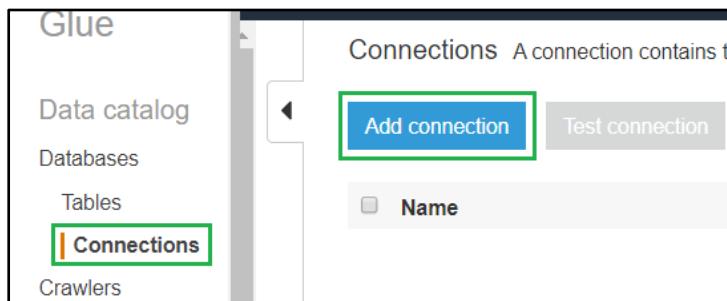
Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target for a data processing job.

[Add tables](#) [Action](#) Filter by attributes or search by keyword

| Name | Database | Location | Classification |
|------------|---------------|--------------------------------------|----------------|
| in-process | sales-info-db | s3://lambda-glue-batch/sales/in-p... | csv |

11.4. Create Connection

25) Go to connections on left tab and click Add connection.



26) Setup connection details

Add connection

Set up your connection's properties.

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

Connection name
sales-db-connection

Connection type
Amazon RDS

Database engine
PostgreSQL

Next

Provide instance, db name, user name and password.

Add connection

Set up access to your database.

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

Instance
lambda-glue-batch

Database name
salesdb

Username
postgres

Password
.....

Back **Next**

Add connection

Connection properties
sales-db-connection
Type: PostgreSQL

Connection access

Review all steps

Connection properties

| | |
|------------------------|-------------------------|
| Name | sales-db-connection |
| Type | Amazon RDS (PostgreSQL) |
| Require SSL connection | false |
| Description (optional) | - |

Connection access

| | |
|-----------------|-------------------|
| Instance | lambda-glue-batch |
| Username | postgres |
| VPC Id | |
| Subnet | |
| Security groups | |

[Back](#) [Finish](#)

27) Now test by selecting created connection and click Test connection

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

| | |
|---|--|
| <input type="checkbox"/> Name | |
| <input checked="" type="checkbox"/> sales-db-connection | |

Testing **sales-db-connection** access to your data store is in progress. This can take a few moments.

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

| Name | Type |
|---|------|
| <input checked="" type="checkbox"/> sales-db-connection | JDBC |

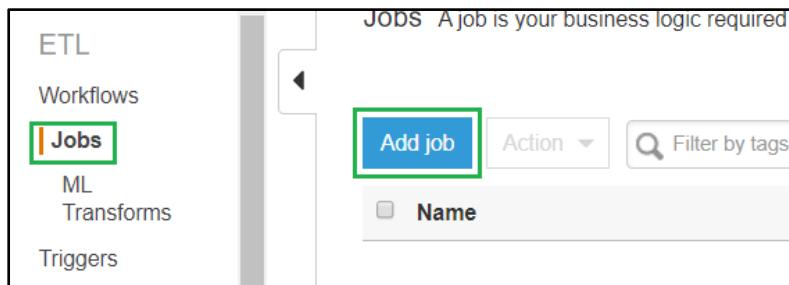
sales-db-connection connected successfully to your instance.

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

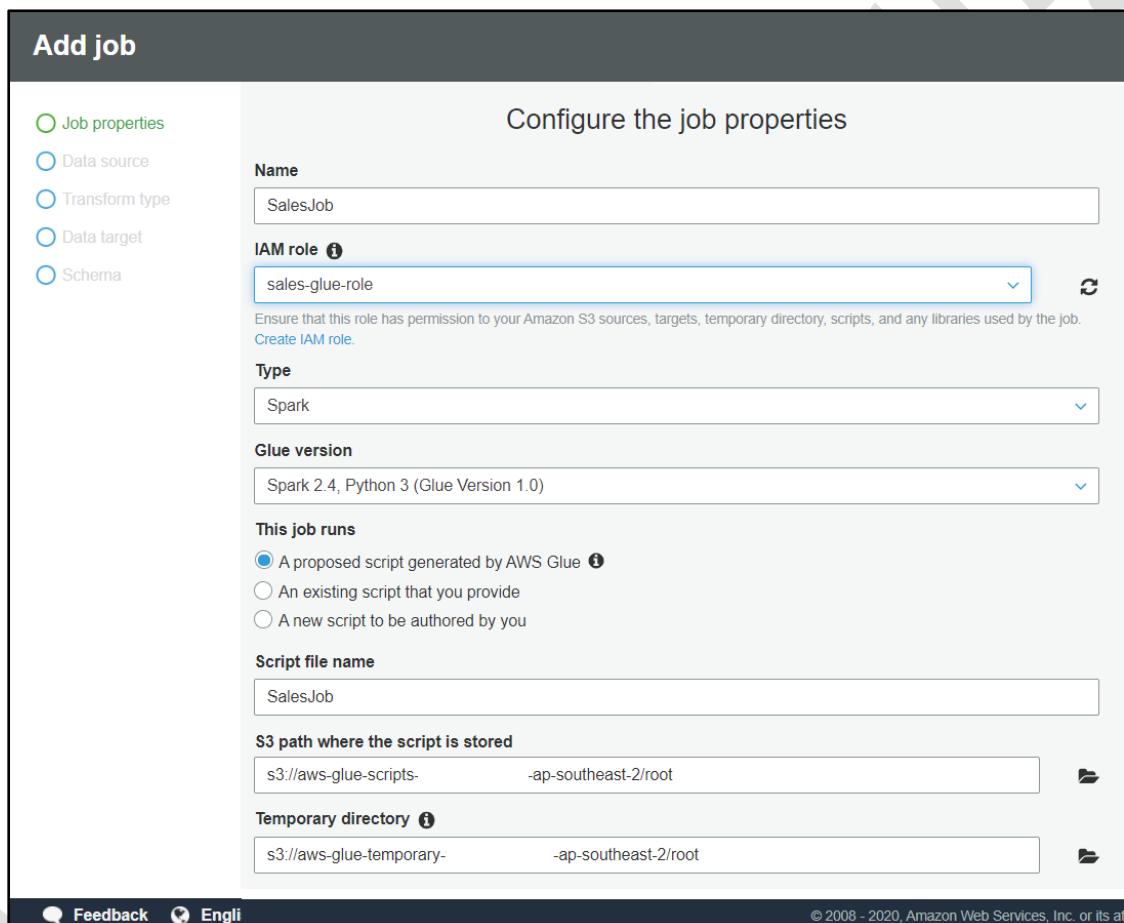
| Name | Type |
|---|------|
| <input checked="" type="checkbox"/> sales-db-connection | JDBC |

11.5. Create Job

28) Go to jobs on left tab and click Add Job



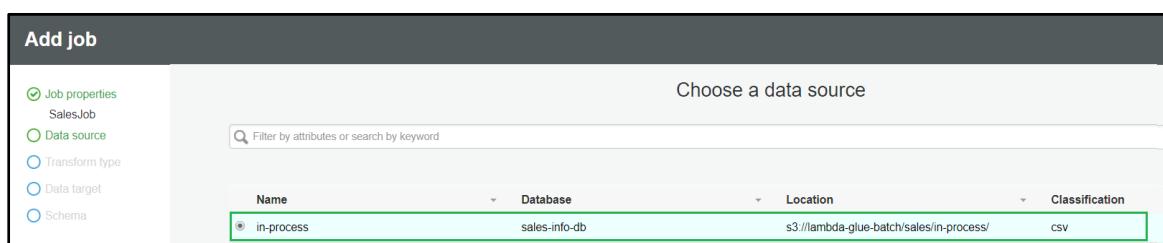
29) Provide Name and IAM Role, leave the rest to default values.



The screenshot shows the 'Add job' configuration page. The left sidebar has 'Job properties' selected (highlighted with a green box). The main area is titled 'Configure the job properties'. It includes fields for 'Name' (SalesJob), 'IAM role' (sales-glue-role), 'Type' (Spark), 'Glue version' (Spark 2.4, Python 3 (Glue Version 1.0)), 'This job runs' (radio button selected for 'A proposed script generated by AWS Glue'), 'Script file name' (SalesJob), 'S3 path where the script is stored' (s3://aws-glue-scripts-ap-southeast-2/root), and 'Temporary directory' (s3://aws-glue-temporary-ap-southeast-2/root). At the bottom, there are 'Feedback' and 'English' buttons, and a copyright notice: '© 2008 - 2020, Amazon Web Services, Inc. or its affiliates'.

From the above screen shot it is clear that, Glue will generate Spark based code and stores it on s3 bucket as mentioned in above screen shot.

30) Choose Data Source and click next



The screenshot shows the 'Add job' configuration page with the 'Data source' tab selected (highlighted with a green box). The left sidebar also has 'Data source' selected. The main area shows a table with one row. The table columns are 'Name', 'Database', 'Location', and 'Classification'. The row contains 'in-process', 'sales-info-db', 's3://lambda-glue-batch/sales/in-process/', and 'csv' respectively. There is also a 'Filter by attributes or search by keyword' search bar at the top of the table.

31) Leave Transform Type to default values

Add job

Job properties
SalesJob

Data source
in-process

Transform type
Change schema

Data target

Schema

Choose a transform type

Change schema
Change schema of your source data and create a new target dataset

Find matching records
Use machine learning to find matching records within your source data

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

32) Select the target database

Add job

Job properties
SalesJob

Data source
in-process

Transform type
Change schema

Data target

Schema

Choose a data target

Create tables in your data target

Use tables in the data catalog and update your data target

Data store
JDBC

Connection
sales-db-connection

[Add connection](#)

Database name ?
salesdb

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

33) Mapping would be shown as below, click save job and edit script

Add job

Job properties
SalesJob

Data source
in-process

Transform type
Change schema

Data target
sales-db-connection

Schema

Map the source columns to target columns.

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with **Map to target**. You can **Clear** all mappings and **Reset** to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

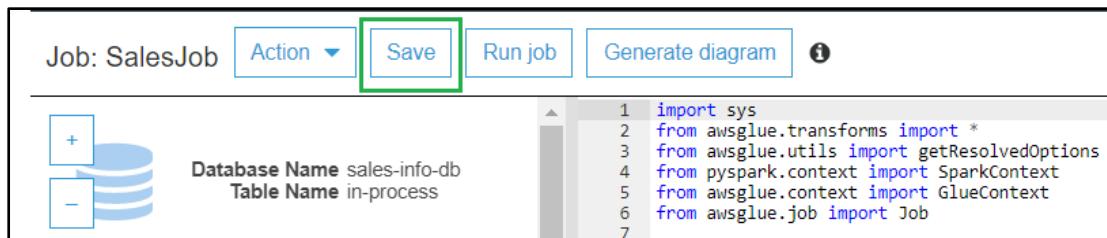
| Source | Target |
|----------------|----------------|
| Column name | Column name |
| Data type | Data type |
| Map to target | |
| region | region |
| string | string |
| region | region |
| country | country |
| string | string |
| item type | item type |
| string | string |
| sales channel | sales channel |
| string | string |
| order priority | order priority |
| string | string |
| order date | order date |
| string | string |
| order id | order id |
| bigint | long |
| ship date | ship date |
| string | string |
| units sold | units sold |
| bigint | long |
| unit price | unit price |
| double | double |
| unit cost | unit cost |
| double | double |
| total revenue | total revenue |
| double | double |
| total cost | total cost |
| double | double |
| total profit | total profit |
| double | double |

[Add column](#) [Clear](#) [Reset](#)

[Feedback](#) [English \(US\)](#)

© 2006 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

34) Save the script



Job: SalesJob Action ▾ **Save** Run job Generate diagram ⓘ

Database Name sales-info-db
Table Name in-process

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
```

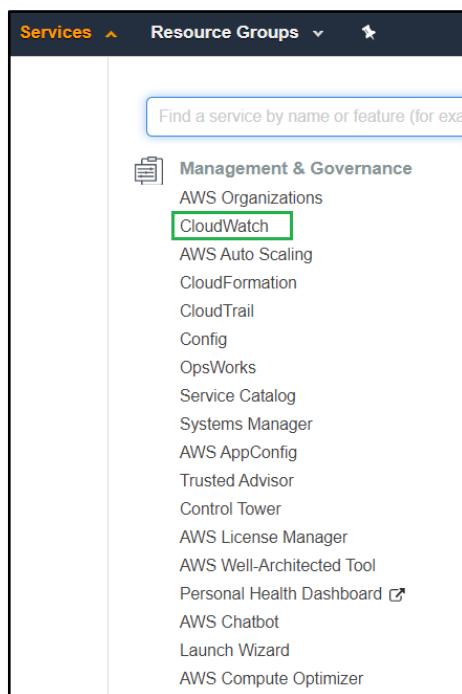
35) Job added



| Name | Type | ETL language | Script location |
|----------|-------|--------------|------------------------|
| SalesJob | Spark | python | s3://aws-glue-scrip... |

12. Setup Cloud Watch Event Rule

1) Select Cloud Watch Found at: Services → Management & Governance → CloudWatch



2) Click on Events → Rules → Create Rule



- 3) Select Event Pattern, Service Name and Event Type and then click Add Target

Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

Event Pattern ? Schedule ?

Build event pattern to match events by service

Service Name: Glue
Event Type: Glue Job State Change

Event Pattern Preview

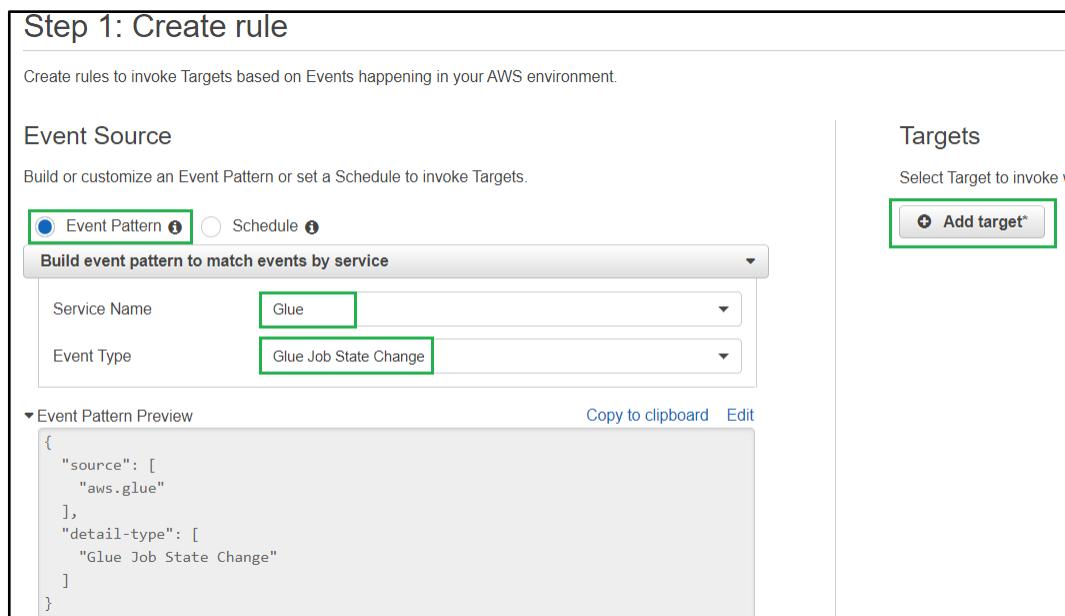
```
{
  "source": [
    "aws.glue"
  ],
  "detail-type": [
    "Glue Job State Change"
  ]
}
```

[Copy to clipboard](#) [Edit](#)

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

Add target*



- 4) Configure Target

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

Lambda function

Function*: cloudwatch-event-trigger

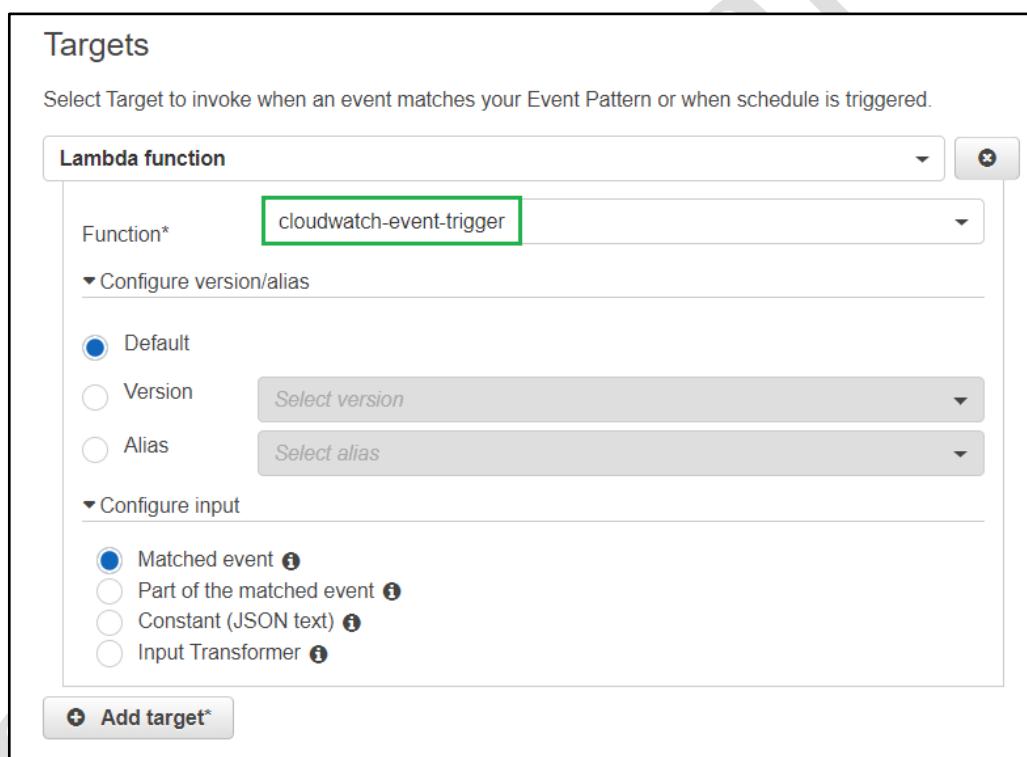
Configure version/alias

Default
 Version: Select version
 Alias: Select alias

Configure input

Matched event ?
 Part of the matched event ?
 Constant (JSON text) ?
 Input Transformer ?

Add target*



- 5) Provide name – “sales-glue-process-completed” and click create rule.

Step 2: Configure rule details

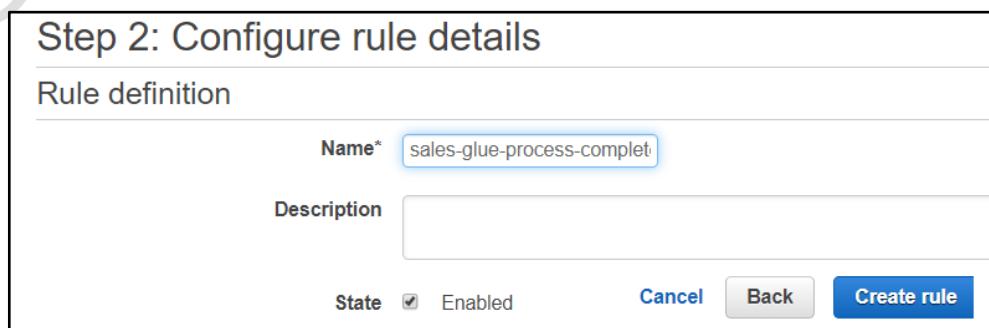
Rule definition

Name*: sales-glue-process-complet

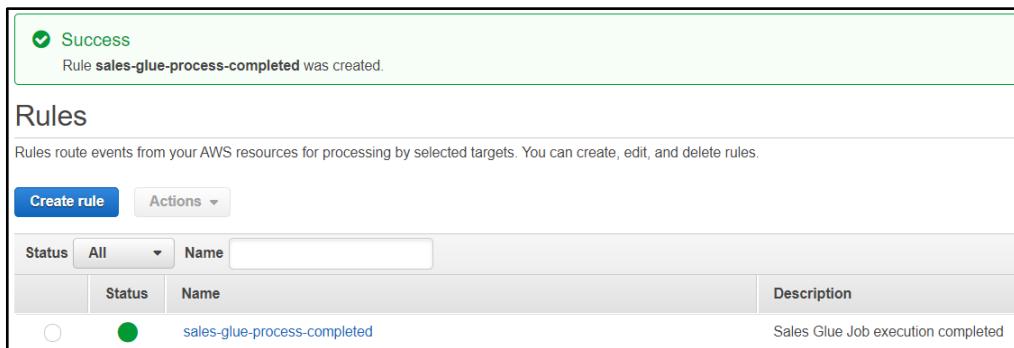
Description:

State Enabled

Create rule



6) Rule created



The screenshot shows the AWS Lambda Rules page. At the top, a green success message box says "Success" and "Rule sales-glue-process-completed was created." Below this, the "Rules" section title is displayed. A sub-header "Rules route events from your AWS resources for processing by selected targets. You can create, edit, and delete rules." is present. A "Create rule" button is on the left, and an "Actions" dropdown is on the right. A filter bar allows filtering by "Status" (All) and "Name". A table lists the rule "sales-glue-process-completed" with a status of "Active" (green dot) and a description "Sales Glue Job execution completed".

13. Build Lambdas

- 1) Build below 3 code repositories.
 - a) s3-lambda-glue
 - b) cloudwatch-lambda-s3
 - c) lambda-lambda
- 2) Clean & Build the code once from the terminal

```
D:\Sandeep\Work\Code\s3-lambda-glue>gradle clean build

BUILD SUCCESSFUL in 4s
3 actionable tasks: 3 executed
D:\Sandeep\Work\Code\s3-lambda-glue>
```

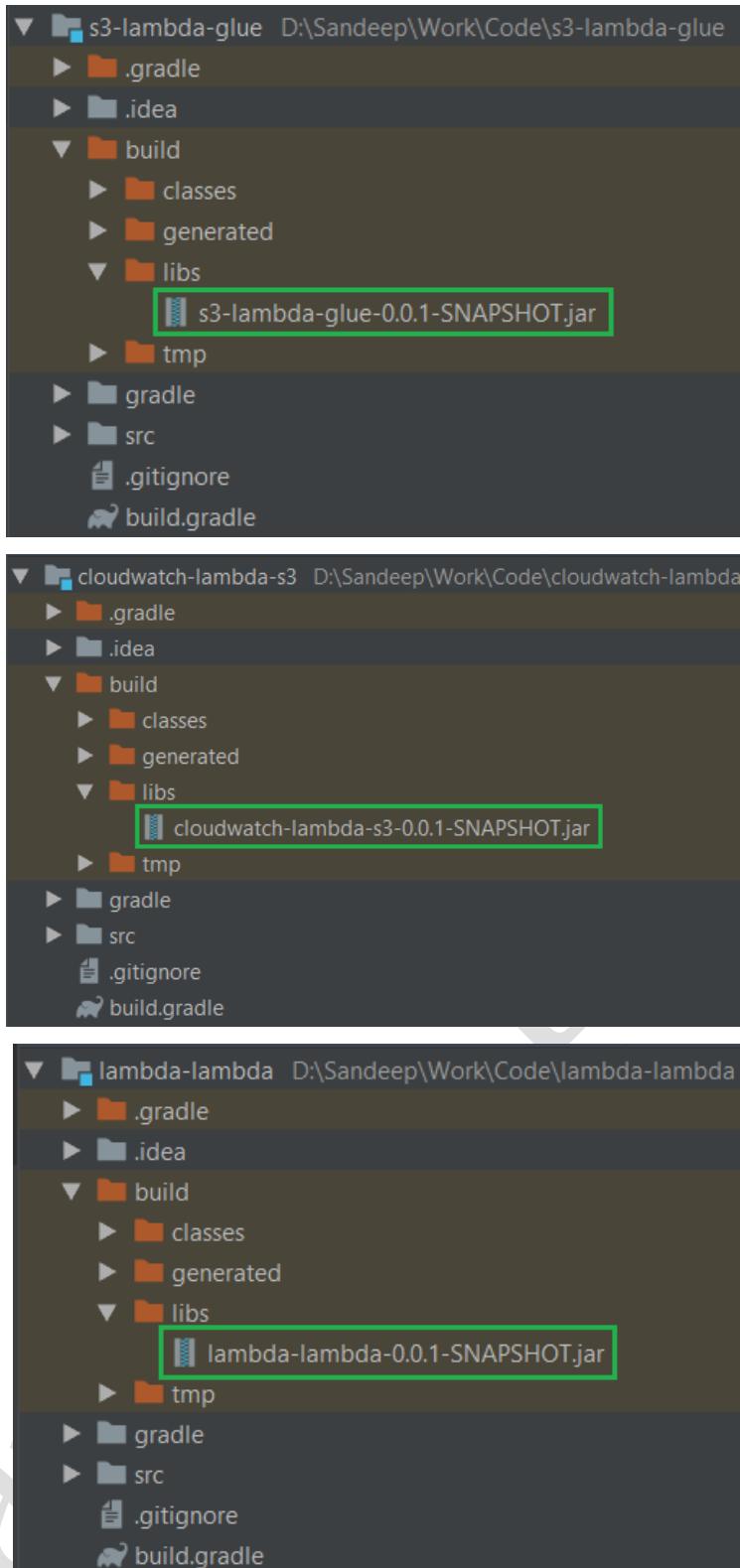
```
D:\Sandeep\Work\Code\cloudwatch-lambda-s3>gradle clean build

BUILD SUCCESSFUL in 4s
3 actionable tasks: 3 executed
D:\Sandeep\Work\Code\cloudwatch-lambda-s3>
```

```
D:\Sandeep\Work\Code\lambda-lambda>gradle clean build

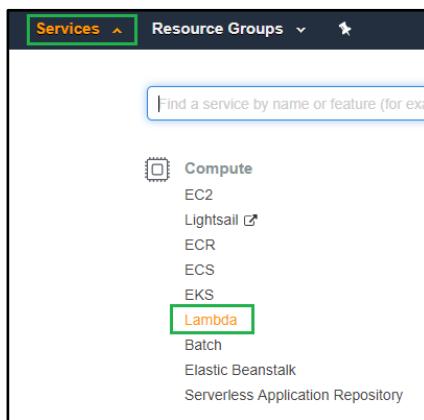
BUILD SUCCESSFUL in 4s
3 actionable tasks: 3 executed
D:\Sandeep\Work\Code\lambda-lambda>
```

- 3) Use the generated jars for lambda deployment.
 - i. s3-lambda-glue-0.0.1-SNAPSHOT.jar
 - ii. cloudwatch-lambda-s3-0.0.1-SNAPSHOT.jar
 - iii. lambda-lambda-0.0.1-SNAPSHOT.jar

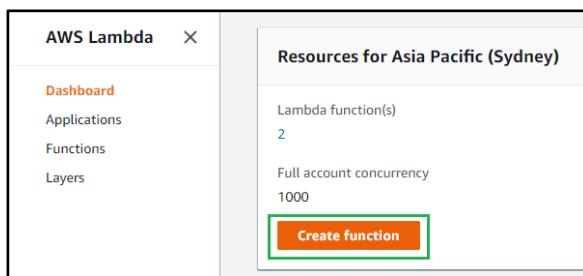


14. Deploying Lambda

- 1) Go to Services → Compute → Lambda



- 2) Click create function.



14.1 sales-glue-trigger

- 3) Create new function

The screenshot shows the 'Basic information' step of the Lambda function creation wizard. It includes fields for 'Function name' (set to 'sales-glue-trigger'), 'Runtime' (set to 'Java 8'), and 'Permissions' (set to 'Use an existing role'). The 'Execution role' section shows 'sales-lambda-role' selected. At the bottom are 'Cancel' and 'Create function' buttons.

Basic information

Function name
Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function.

Permissions [Info](#)
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can also choose an existing role or create a new one.

▼ Choose or create an execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

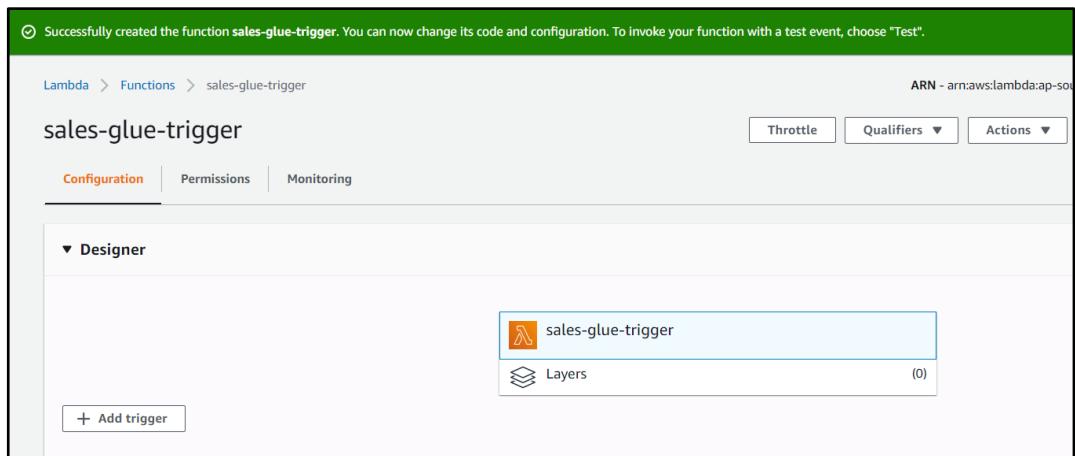
Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have the required permissions.

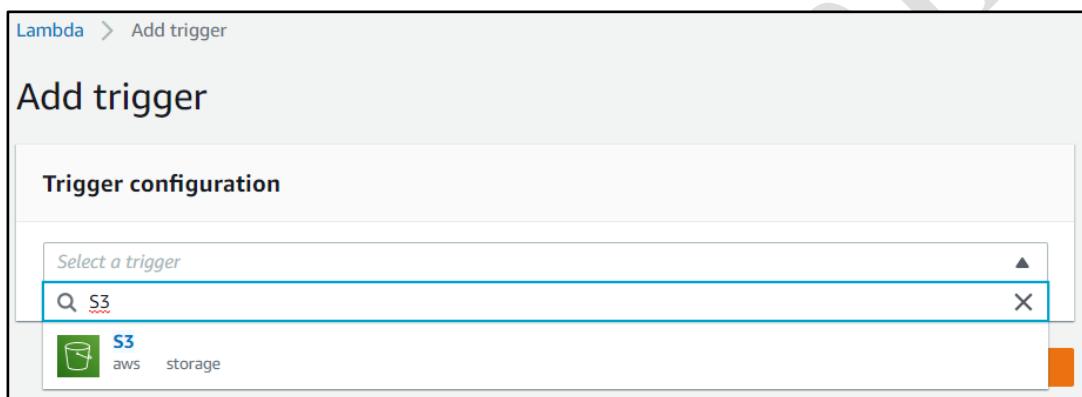
[View the sales-lambda-role role](#) on the IAM console.

[Cancel](#) [Create function](#)

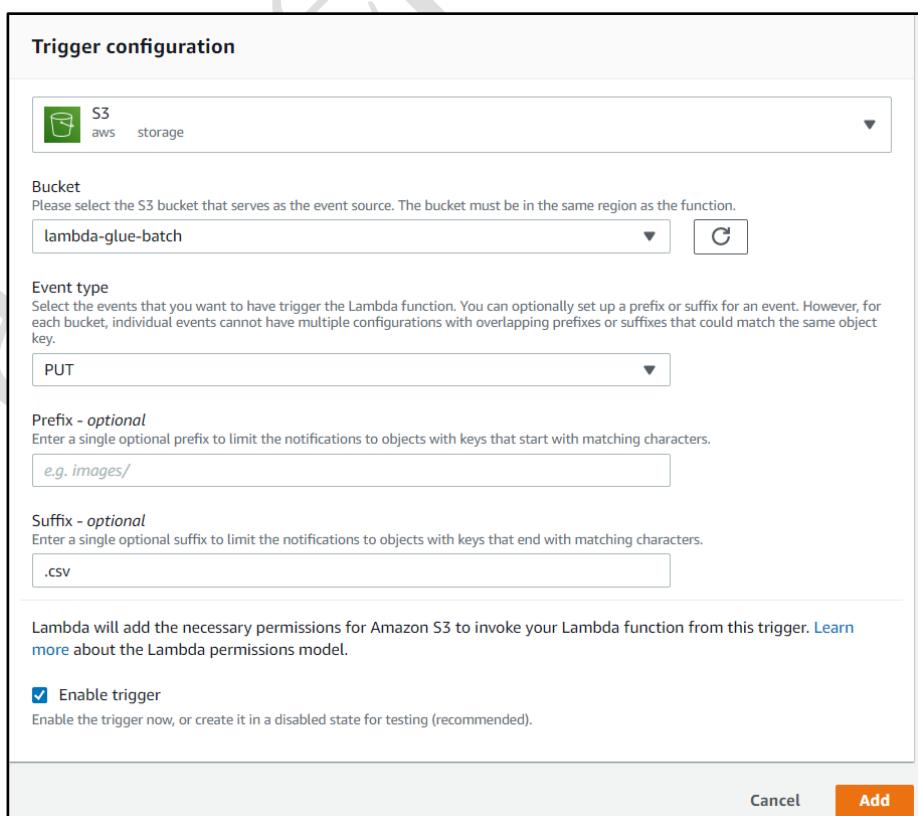
- 4) Lambda is created as shown below.



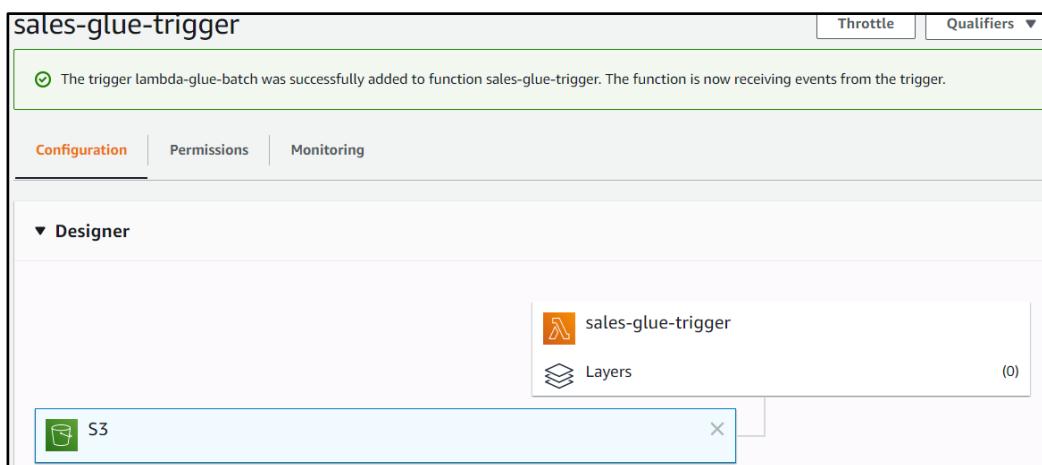
- 5) Click on add the trigger & Select S3



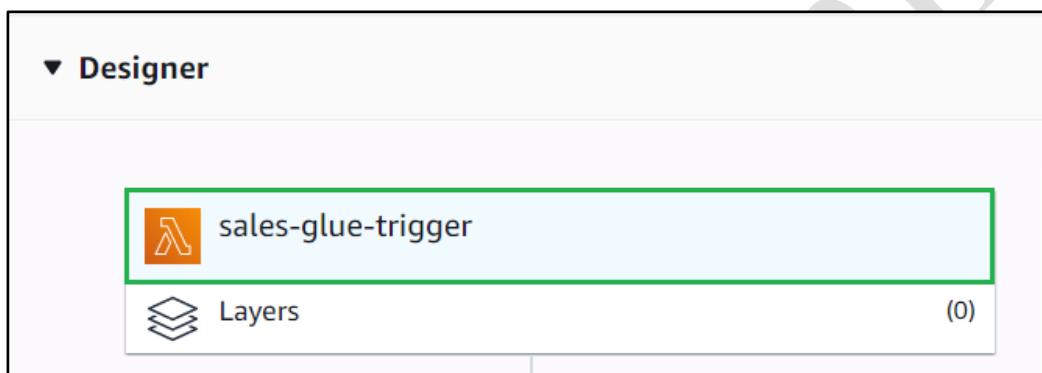
- 6) Configure trigger details, Add event type as **PUT** (when file is PUT in bucket, it triggers Lambda) for the files suffixed with **.csv** extension. then click Add.



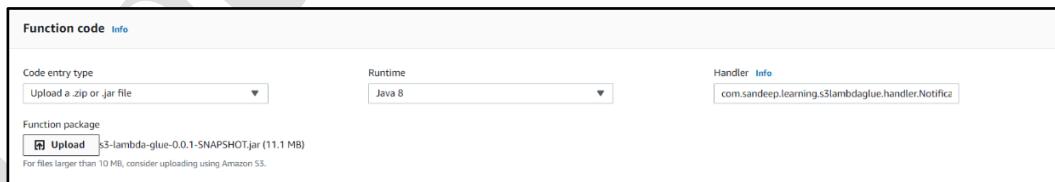
- 7) S3 will be attached as trigger to Lambda as shown below.



- 8) Click on Lambda in the Designer pane.



- 9) In the Function code Pane –
upload jar - s3-lambda-glue-0.0.1-SNAPSHOT.jar
provide the Handler info –
com.sandeep.learning.s3lambdaglue.handler.NotificationHandler::handleRequest
click save.



- 10) Once uploaded and saved, Lambda is ready for testing.

14.2 sales-glue-processed-trigger

11) Create new function

Basic information

Function name
Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function.

Permissions [Info](#)
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can change the execution role later.

▼ Choose or create an execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permissions to invoke AWS services.

[View the sales-lambda-role role](#) on the IAM console.

[Cancel](#) [Create function](#)

12) Lambda is created as shown below.

Successfully created the function **sales-glue-processed-trigger**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > sales-glue-processed-trigger ARN - arn:aws:lambda:ap-southeast-1:123456789012: sales-glue-processed-trigger

sales-glue-processed-trigger

Throttle Qualifiers Actions

Configuration Permissions Monitoring

▼ Designer

 sales-glue-processed-trigger
 Layers (0)

13) Click on add the trigger & Select Cloud Watch Events

Lambda > Add trigger

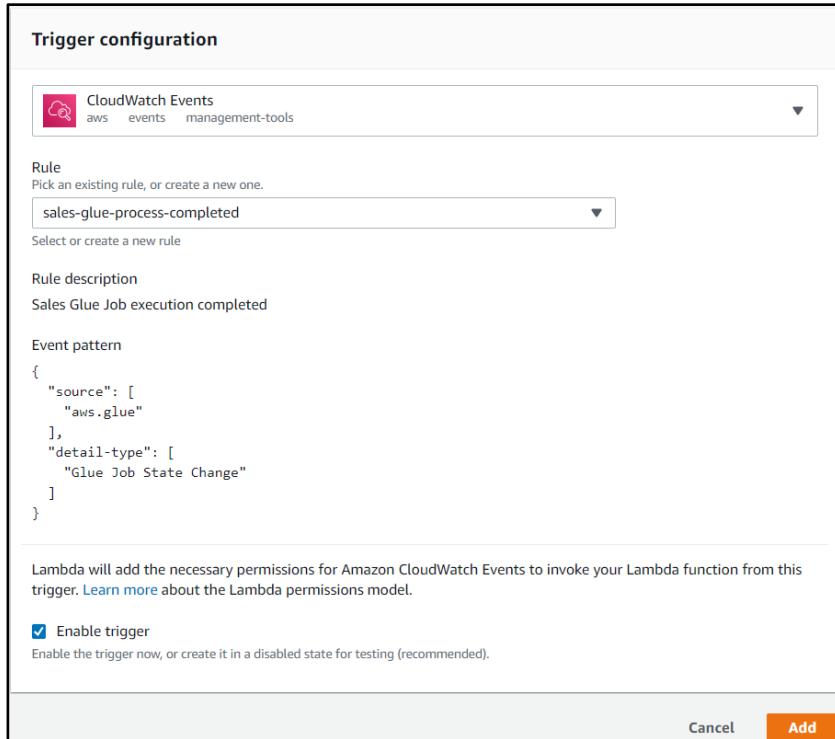
Add trigger

Trigger configuration

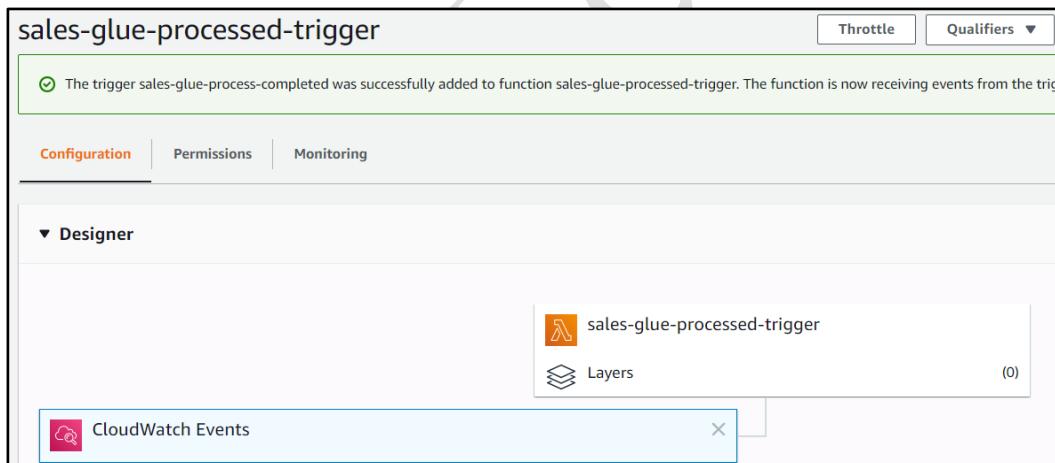
Select a trigger

 **CloudWatch Events**
aws events management-tools

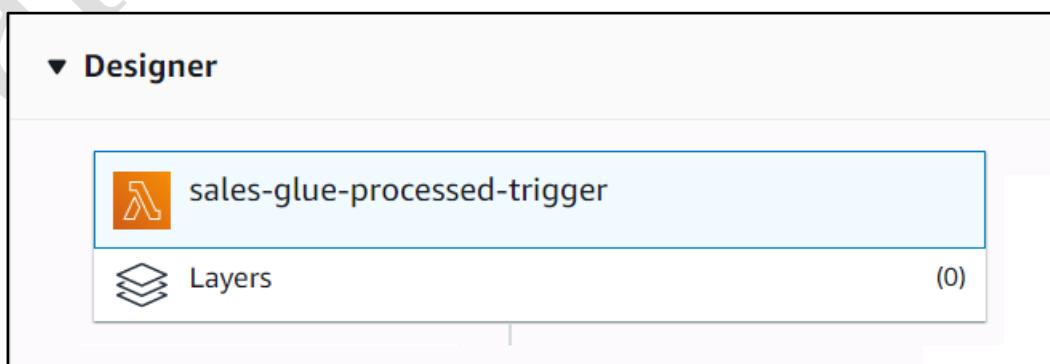
14) Configure trigger details, by selecting the rule sales-glue-process-completed created earlier.



15) CloudWatch Events will be attached as trigger to Lambda as shown below.



16) Click on Lambda in the Designer pane.



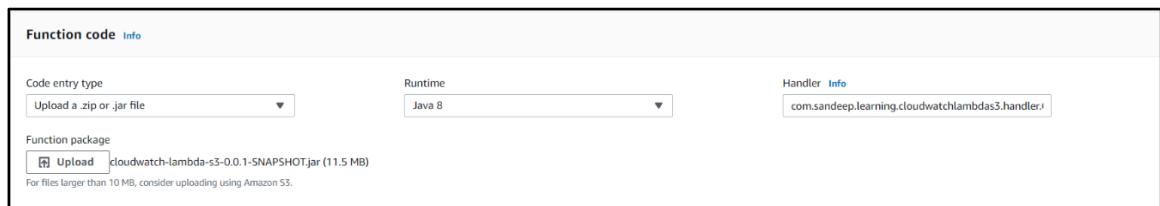
17) In the Function code Pane –

upload jar - cloudwatch-lambda-s3-0.0.1-SNAPSHOT.jar

provide the Handler info –

com.sandeep.learning.cloudwatchlambdas3.handler.CloudWatchEventHandler::handleRequest

click save.

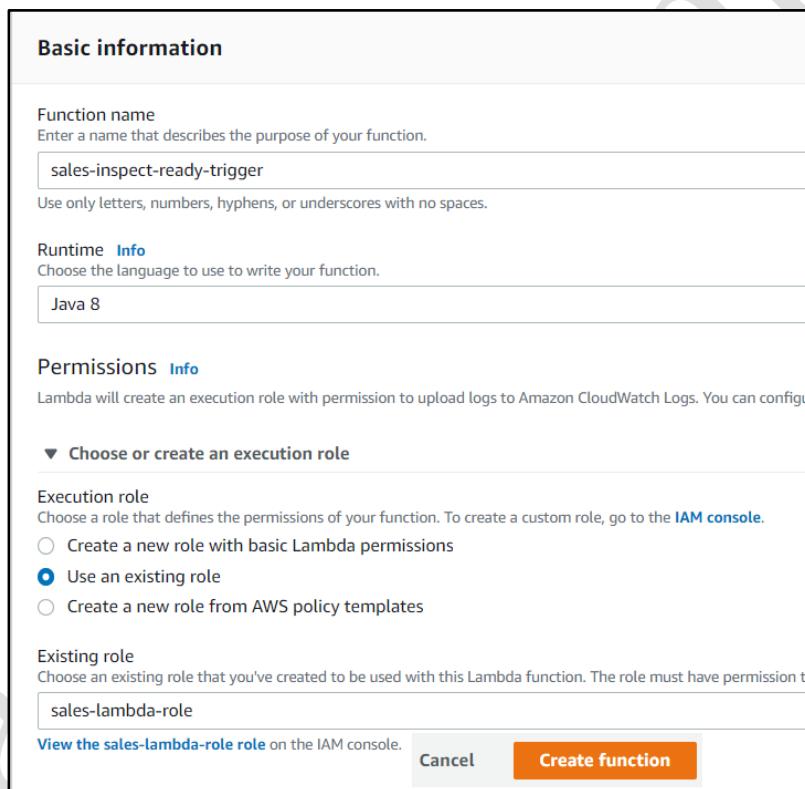


The screenshot shows the 'Function code' pane of the AWS Lambda console. It includes fields for 'Code entry type' (set to 'Upload a zip or .jar file'), 'Runtime' (set to 'Java 8'), and 'Handler' (set to 'com.sandeep.learning.cloudwatchlambdas3.handler'). A file named 'cloudwatch-lambda-s3-0.0.1-SNAPSHOT.jar' (11.5 MB) is uploaded. A note at the bottom states: 'For files larger than 10 MB, consider uploading using Amazon S3.'

18) Once uploaded and saved, Lambda is ready for testing.

14.3 sales-inspect-ready-trigger

19) Create new function



The screenshot shows the 'Basic information' dialog for creating a new Lambda function. It includes fields for 'Function name' (set to 'sales-inspect-ready-trigger'), 'Runtime' (set to 'Java 8'), and 'Permissions' (set to 'sales-lambda-role'). The 'Choose or create an execution role' section shows 'Use an existing role' selected. The 'Create function' button is highlighted in orange at the bottom right.

Basic information

Function name
Enter a name that describes the purpose of your function.
sales-inspect-ready-trigger
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function.
Java 8

Permissions [Info](#)
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure this role to have additional permissions.

▼ Choose or create an execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

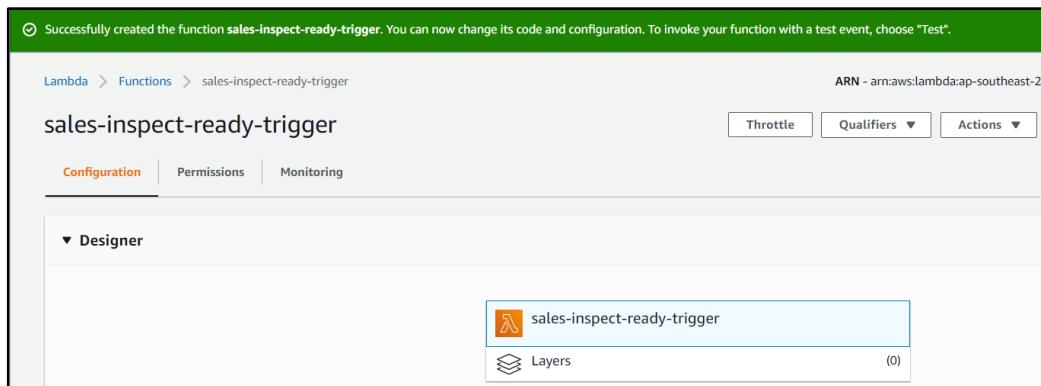
Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to

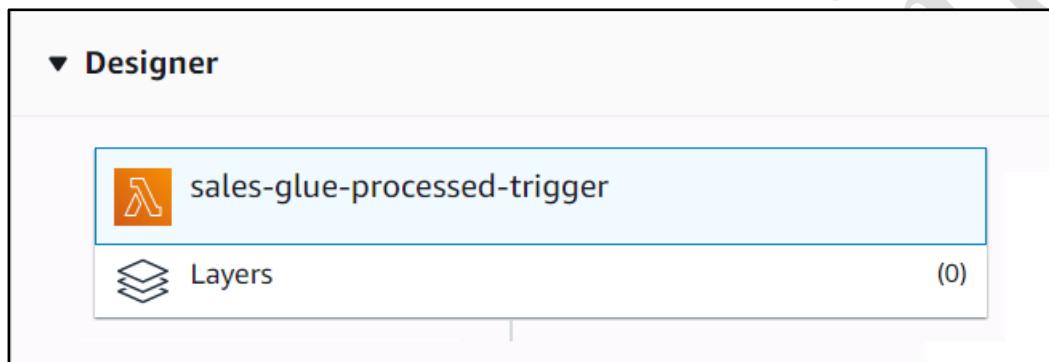
sales-lambda-role
[View the sales-lambda-role role](#) on the IAM console.

Create function

20) Lambda is created as shown below.



21) Click on Lambda in the Designer pane.



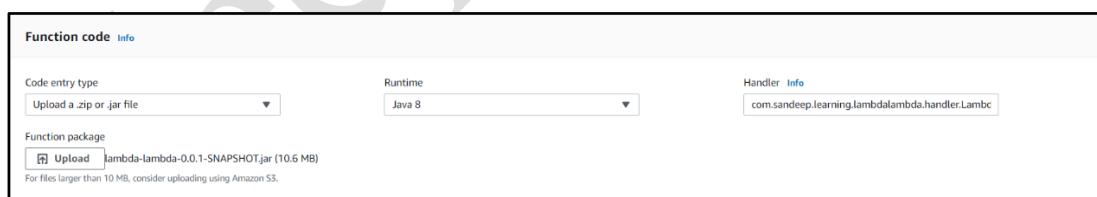
22) In the Function code Pane –

upload jar - lambda-lambda-0.0.1-SNAPSHOT.jar

provide the Handler info –

com.sandeep.learning.lambdalambda.handler.LambdaCustomEventHandler::handleRequest

click save.



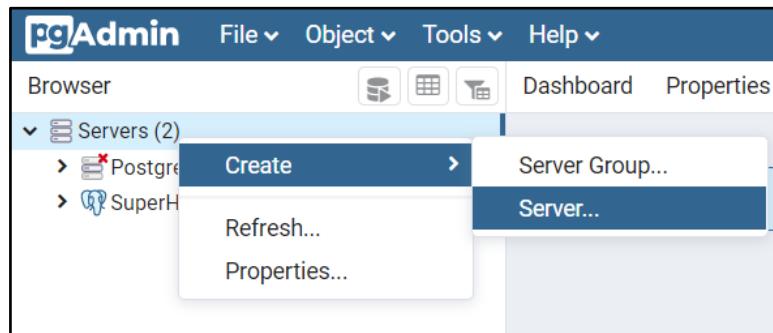
23) Once uploaded and saved, Lambda is ready for testing.

15. Testing & Verification

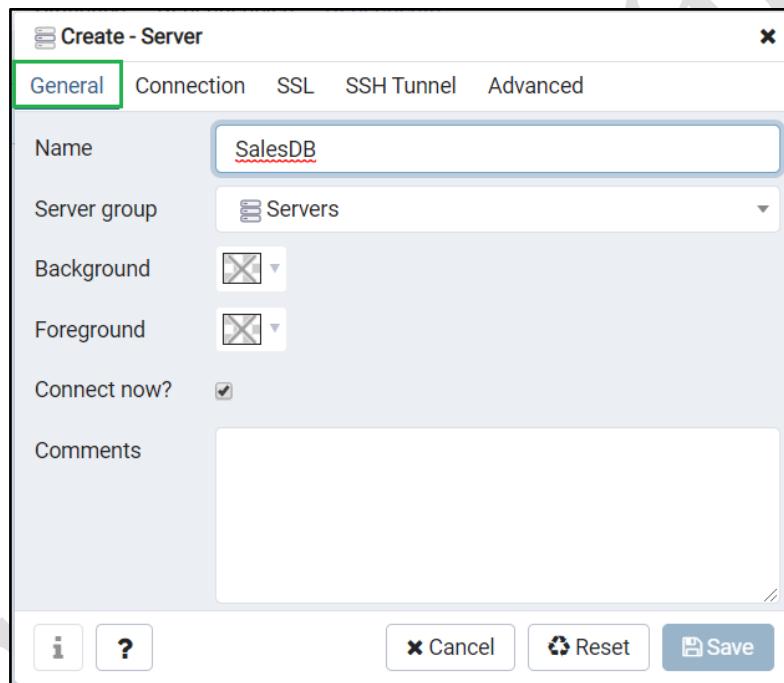
15.1 Setting up pgAdmin Tool

Note: pgAdmin tool can be installed from [here](#).

- 1) Connection to database from pgAdmin tool
- 2) Right click on Servers → Create → Server

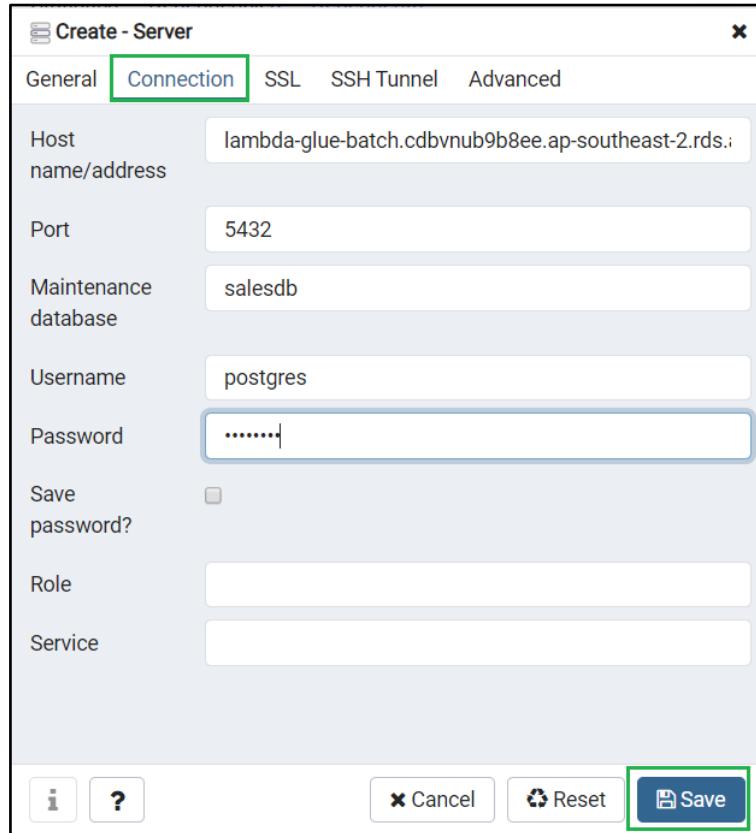


- 3) Provide below details

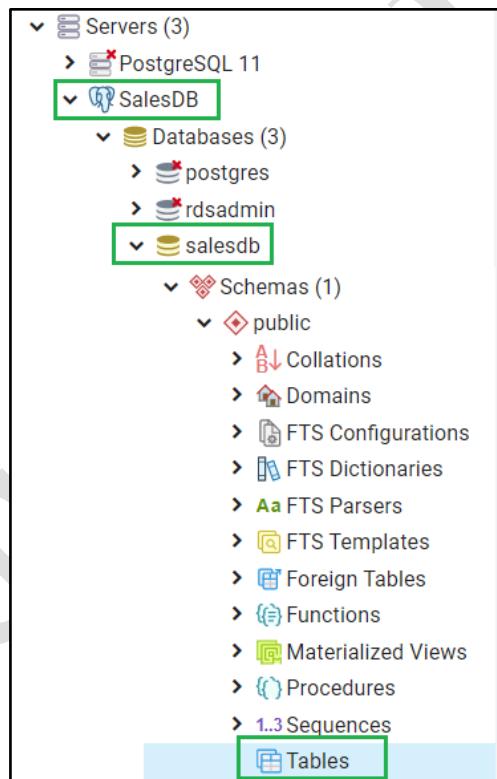


- 4) Select Connection tab and provide –

- Host name from RDS → Databases → Connectivity & security → Endpoint
- Maintenance database from RDS → Databases → Configuration → DB name
- Port from RDS → Databases → Connectivity & security → Port
- Username → postgres
- Password → Use the password which was provided during database creation



- 5) Click Save, on the left pane check the tables



From the above screen shot you can see that there are no tables available here.

15.2 Testing Scenario

- 6) Now upload 2 files (data1.csv, data2.csv) to S3 Bucket ready folder in one shot and once the files are moved to in-process folder upload one more file (data3.csv).
- data1.csv – has 5 records as shown below.

| | Region,Country,Item Type,Sales Channel,Order Priority,Order Date, |
|---|---|
| 1 | Sub-Saharan Africa,South Africa,Fruits,Offline,M,7/27/2012,443368 |
| 2 | Middle East and North Africa,Morocco,Clothes,Online,M,9/14/2013,6 |
| 3 | Australia and Oceania,Papua New Guinea,Meat,Offline,M,5/15/2015,9 |
| 4 | Sub-Saharan Africa,Djibouti,Clothes,Offline,H,5/17/2017,880811536 |

data2.csv – has 7 records as shown below.

| | Region,Country,Item Type,Sales Channel,Order Priority,Order Date, |
|---|---|
| 1 | Europe,Slovakia,Beverages,Offline,L,10/26/2016,174590194,12/4/201 |
| 2 | Asia,Sri Lanka,Fruits,Online,L,11/7/2011,830192887,12/18/2011,137 |
| 3 | Sub-Saharan Africa,Seychelles ,Beverages,Online,M,1/18/2013,42579 |
| 4 | Sub-Saharan Africa,Tanzania,Beverages,Online,L,11/30/2016,6598781 |
| 5 | Sub-Saharan Africa,Ghana,Office Supplies,Online,L,3/23/2017,60124 |
| 6 | Sub-Saharan Africa,Tanzania,Cosmetics,Offline,L,5/23/2016,7390080 |

data3.csv – has 7 records as shown below.

| | Region,Country,Item Type,Sales Channel,Order Priority,Order Date, |
|---|---|
| 1 | Europe,Slovakia,Beverages,Offline,L,10/26/2016,174590194,12/4/201 |
| 2 | Asia,Sri Lanka,Fruits,Online,L,11/7/2011,830192887,12/18/2011,137 |
| 3 | Sub-Saharan Africa,Seychelles ,Beverages,Online,M,1/18/2013,42579 |
| 4 | Sub-Saharan Africa,Tanzania,Beverages,Online,L,11/30/2016,6598781 |
| 5 | Sub-Saharan Africa,Ghana,Office Supplies,Online,L,3/23/2017,60124 |
| 6 | Sub-Saharan Africa,Tanzania,Cosmetics,Offline,L,5/23/2016,7390080 |

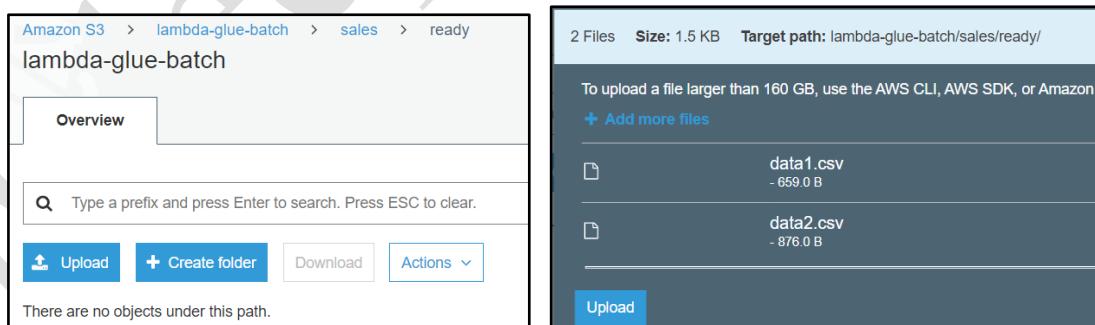
15.3 Expected Behaviour vs Actual Behaviour

| S. No | Expected Behaviour | Actual Behaviour |
|-------|---|--|
| 1 | sales-glue-trigger lambda is invoked upon files upload. Moves data1.csv and data2.csv to in-process folder, triggers glue job and updates the dynamo db jobs table with job id, files and job status. | Aligns with expected behaviour. Refer point 9 and point 12. |
| 2 | When the data1.csv and data2.csv are in-process, upload data3.csv file. sales-glue-trigger will identify that job is in-process and leaves the data3.csv file in ready folder and lets the glue to process data1.csv and data2.csv files. | Aligns with expected behaviour. Refer point 13, 2 nd half of the screenshot. |

| | | |
|---|---|--|
| 3 | Upon glue state changes, sales-glue-processed-trigger lambda moves data1.csv and data2.csv files to succeeded or failed directory based on glue job state. Updates the job id status and triggers call to sales-inspect-ready-trigger . | Aligns with expected behaviour. Refer point 14, 15 1 st half of the screenshot. |
| 4 | sales-inspect-ready-trigger will check the ready directory and identifies that data3.csv is to be processed and moves the file to in-process directory, invokes glue job and inserts new job id in the dynamo db table. | Aligns with expected behaviour. Refer point 16 1 st half of screenshot and point 14. |
| 5 | Upon glue state changes, sales-glue-processed-trigger lambda moves data3.csv files to succeeded or failed directory based on glue job state. Updates the job id status and triggers call to sales-inspect-ready-trigger . | Aligns with expected behaviour. Refer point 15 2 nd half of the screenshot and point 17. |
| 6 | sales-inspect-ready-trigger will check the ready directory and identifies that no files are available for processing and stops. | Aligns with expected behaviour. Refer point 16 2 nd half of the screenshot. |
| 7 | RDS salesdb → in-process table should contain a total of $5+7+7 = 19$ records. | Aligns with expected behaviour. Refer point 19. |

15.4 Testing In-Progress

- 7) Upload data1.csv and data2.csv to s3 bucket ready directory.



Amazon S3 > lambda-glue-batch > sales > ready

lambda-glue-batch

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions

There are no objects under this path.

2 Files Size: 1.5 KB Target path: lambda-glue-batch/sales/ready/

To upload a file larger than 160 GB, use the AWS CLI, AWS SDK, or Amazon S3 API.

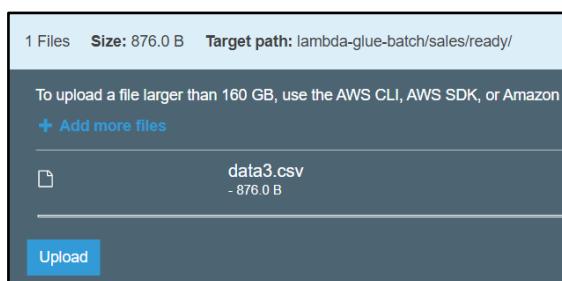
+ Add more files

data1.csv - 659.0 B

data2.csv - 876.0 B

Upload

- 8) Files moved immediately in-process directory so uploading data3.csv



1 Files Size: 876.0 B Target path: lambda-glue-batch/sales/ready/

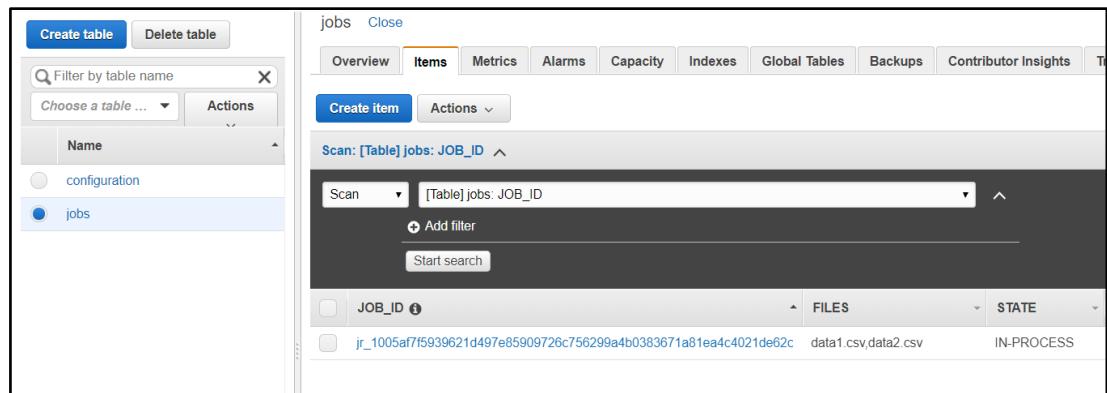
To upload a file larger than 160 GB, use the AWS CLI, AWS SDK, or Amazon S3 API.

+ Add more files

data3.csv - 876.0 B

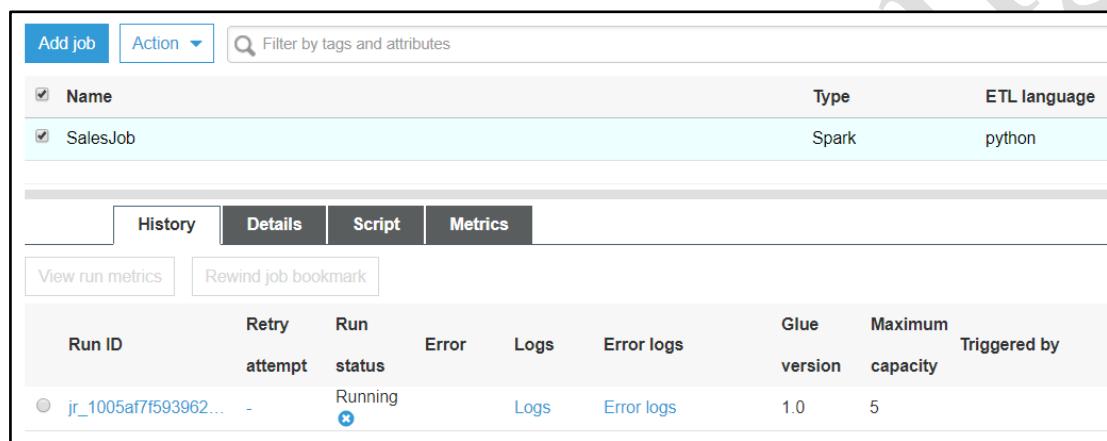
Upload

9) Dynamo DB Jobs table status



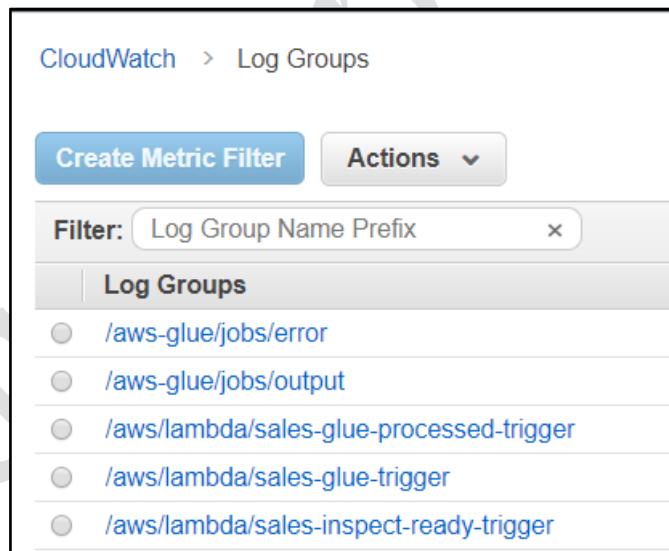
The screenshot shows the AWS DynamoDB console. On the left, there is a sidebar with 'Create table' and 'Delete table' buttons, and a search bar for 'Filter by table name' with 'Choose a table ...' and 'Actions' dropdowns. Below that is a list of tables with 'Name' columns, showing 'configuration' and 'jobs'. The 'jobs' table is selected. On the right, the 'Items' tab is selected in the top navigation bar. Below it, a 'Scan' dropdown is set to 'Scan' and a dropdown for 'Table' is set to 'jobs: JOB_ID'. A 'Start search' button is present. The results table shows one item: 'jr_1005af7f5939621d497e85909726c756299a4b0383671a81ea4c4021de62c' with 'FILES' 'data1.csv,data2.csv' and 'STATE' 'IN-PROCESS'.

10) Glue Job state



The screenshot shows the AWS Glue console. At the top, there are 'Add job' and 'Action' buttons, and a search bar for 'Filter by tags and attributes'. Below that, a table lists jobs with 'Name' and 'Type' columns. 'SalesJob' is listed as 'Spark' type and 'python' ETL language. Below the table are tabs for 'History', 'Details', 'Script', and 'Metrics'. Under 'History', there is a table with columns 'Run ID', 'Retry attempt', 'Run status', 'Error', 'Logs', 'Error logs', 'Glue version', 'Maximum capacity', and 'Triggered by'. One run is listed: 'jr_1005af7f5939621d497e85909726c756299a4b0383671a81ea4c4021de62c' with a run status of 'Running' and logs available.

11) Go to CloudWatch → Log Groups to check logs for corresponding components.



The screenshot shows the AWS CloudWatch Log Groups page. At the top, there is a 'Create Metric Filter' button and an 'Actions' dropdown. Below that is a 'Filter' input field with 'Log Group Name Prefix' set to '/aws-glue/jobs/'. A table titled 'Log Groups' lists several log groups: '/aws-glue/jobs/error', '/aws-glue/jobs/output', '/aws/lambda/sales-glue-processed-trigger', '/aws/lambda/sales-glue-trigger', and '/aws/lambda/sales-inspect-ready-trigger'.

12) Lambda Logs for sales-glue-trigger

Below log is for data2.csv file which actually processed both files.

```
        },
        "object": {
            "key": "sales/ready/data2.csv",
            "size": 876,
            "eTag": "7f78be16b69065f1ab0bf4cba90b6b7e",
            "versionId": "",
            "sequencer": "005E48C42BA1271977",
            "urlDecodedKey": "sales/ready/data2.csv"
        },
        "s3SchemaVersion": "1.0"
    },
    "userIdentity": {
        "principalId": "ATJKA09H9J2FB"
    },
    "glacierEventData": null
}
]
]

▶ 04:25:20      READY STATE FILES ARE - 2
▶ 04:25:20      [sales/ready/]
▶ 04:25:20      [sales/ready/data1.csv]
▶ 04:25:20      [sales/ready/data2.csv]
▶ 04:25:20      IN-PROCESS STATE FILES ARE - 0
▶ 04:25:20      [sales/in-process/]
▶ 04:25:20      FILE [data1.csv] MOVED TO IN-PROCESS STATE
▶ 04:25:21      FILE [data2.csv] MOVED TO IN-PROCESS STATE
▶ 04:25:21      Starting Glue job [SalesJob] for files [data1.csv,data2.csv]
▶ 04:25:21      Glue Job Id is [jr_1005af7f5939621d497e85909726c756299a4b0383671a81ea4c4021de62ce35]
▶ 04:25:21      END RequestId: bf1befee-2855-4d7d-b269-52fd213a5b3d
```

This is PUT trigger for data2.csv file

Below is the log for data1.csv file, which identified that other lambda is processing both files.

```
Message
No older events found at the moment. Retry.

START RequestId: 0b9f3b41-049e-4c99-8f32-3fc69cf2a3e Version: $LATEST
Event generated is [{"Records": [{"awsRegion": "ap-southeast-2", "eventName": "ObjectCreated:Put", "eventSource": "aws:s3", "eventTime": "2020-02-16T04:25:13.540Z", "eventVersion": "2.1", "requestParameters": {"key": "sales/ready/data1.csv", "size": 876, "eTag": "7f78be16b69065f1ab0bf4cba90b6b7e", "versionId": "", "sequencer": "005E48C42BA1271977", "urlDecodedKey": "sales/ready/data1.csv"}, "s3SchemaVersion": "1.0"}]}]
READY STATE FILES ARE - 2
[sales/ready/]
[sales/ready/data1.csv]
[sales/ready/data2.csv]
IN-PROCESS STATE FILES ARE - 1
[sales/in-process/]
[sales/in-process/data1.csv]
JOB RUN IN-PROGRESS.
END RequestId: 0b9f3b41-049e-4c99-8f32-3fc69cf2a3e
REPORT RequestId: 0b9f3b41-049e-4c99-8f32-3fc69cf2a3e Duration: 1315.09 ms Billed Duration: 1400 ms Memory Size: 512 MB Max Memory Used: 153 MB Init Duration: 3063.21 ms
```

This is PUT trigger for data1.csv.
As PUT trigger for data2.csv has moved both files to in-process state, this lambda identifies that data2.csv lambda is processing both files and will die, doing nothing

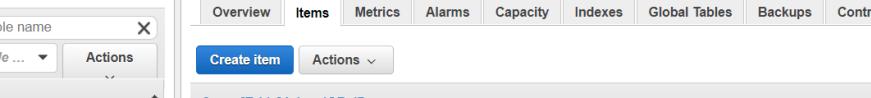
13) Here is the overall snapshot to determine behaviour of lambdas

```
Message
No older events found at the moment. Retry.

START RequestId: bf1befee-2855-4d7d-b269-52fd213a5b3d Version: $LATEST
Event generated is [{"Records": [{"awsRegion": "ap-southeast-2", "eventName": "ObjectCreated:Put", "eventSource": "aws:s3", "eventTime": "2020-02-16T04:25:13.540Z", "eventVersion": "2.1", "requestParameters": {"key": "sales/ready/data1.csv", "size": 876, "eTag": "7f78be16b69065f1ab0bf4cba90b6b7e", "versionId": "", "sequencer": "005E48C42BA1271977", "urlDecodedKey": "sales/ready/data1.csv"}, "s3SchemaVersion": "1.0"}]}]
READY STATE FILES ARE - 2
[sales/ready/]
[sales/ready/data1.csv]
[sales/ready/data2.csv]
IN-PROCESS STATE FILES ARE - 0
[sales/in-process/]
FILE [data1.csv] MOVED TO IN-PROCESS STATE
FILE [data2.csv] MOVED TO IN-PROCESS STATE
Starting Glue job [SalesJob] for files [data1.csv,data2.csv]
Glue Job Id is [jr_1005af7f5939621d497e85909726c756299a4b0383671a81ea4c4021de62ce35]
END RequestId: bf1befee-2855-4d7d-b269-52fd213a5b3d
REPORT RequestId: bf1befee-2855-4d7d-b269-52fd213a5b3d Duration: 1874.34 ms Billed Duration: 1900 ms Memory Size: 512 MB Max Memory Used: 154 MB Init Duration: 3053.05 ms
START RequestId: 23aa8f58-7c17-4633-970c-502a55c67c9c Version: $LATEST
Event generated is [{"Records": [{"awsRegion": "ap-southeast-2", "eventName": "ObjectCreated:Put", "eventSource": "aws:s3", "eventTime": "2020-02-16T04:27:17.201Z", "eventVersion": "2.1", "requestParameters": {"key": "sales/ready/data3.csv", "size": 876, "eTag": "7f78be16b69065f1ab0bf4cba90b6b7e", "versionId": "", "sequencer": "005E48C42BA1271977", "urlDecodedKey": "sales/ready/data3.csv"}, "s3SchemaVersion": "1.0"}]}]
READY STATE FILES ARE - 1
[sales/ready/]
[sales/ready/data1.csv]
[sales/ready/data2.csv]
IN-PROCESS STATE FILES ARE - 2
[sales/in-process/]
[sales/in-process/data1.csv]
[sales/in-process/data2.csv]
JOB RUN IN-PROGRESS.
END RequestId: 23aa8f58-7c17-4633-970c-502a55c67c9c
REPORT RequestId: 23aa8f58-7c17-4633-970c-502a55c67c9c Duration: 369.15 ms Billed Duration: 400 ms Memory Size: 512 MB Max Memory Used: 154 MB
```

Execution when data1.csv and data2.csv uploaded to s3
Execution when data3.csv is uploaded to s3

14) Dynamo DB Job state changed to SUCCEEDED & data3.csv is picked up for processing.



The screenshot shows the AWS Lambda console interface. At the top, there are buttons for 'Create table' and 'Delete table'. Below this is a search bar with the placeholder 'Filter by table name' and a 'Choose a table ...' dropdown. A sidebar on the left lists tables: 'configuration' (radio button not selected) and 'jobs' (radio button selected). The main content area is titled 'jobs' and shows the table structure. The 'Items' tab is selected, showing a scan operation for the 'jobs' table with the primary key 'JOB_ID'. The results table has columns: 'JOB_ID', 'FILES', and 'STATE'. Two items are listed: one with 'JOB_ID' 'jr_1005af75939621d497e85909726c756299a4b0383671a81ea4c4021de62c', 'FILES' 'data1.csv,data2.csv', and 'STATE' 'SUCCEEDED'; and another with 'JOB_ID' 'jr_ee951a50bf4493e39f5515b55317506dea7b984c8a26d86a2af2b16350d7d', 'FILES' 'data3.csv', and 'STATE' 'IN-PROCESS'.

15) Let's check logs for sales-glue-processed-trigger lambda

No older events found at the moment. [Retry](#)

START RequestId: 3303f31b-058f-477c-a8d5-9a218e4a330e Version: \$LATEST

Cloud Watch Event is [{account: region: ap-southeast-2, detail: {jobName=SalesJob, severity=INFO, state=SUCCEEDED, jobRunId=jr_1005af7f5939621d497e85909726c756}}

FILE [data1.csv] MOVED TO SUCCEEDED STATE

FILE [data2.csv] MOVED TO SUCCEEDED STATE

Moving data1.csv and data2.csv files to SUCCEEDED directory & triggering call to sales-inspect-ready-trigger lambda

Invoking [sales-inspect-ready-trigger] Lambda Function

END RequestId: 3303f31b-058f-477c-a8d5-9a218e4a330e

REPORT RequestId: 3303f31b-058f-477c-a8d5-9a218e4a330e Duration: 7075.52 ms Billed Duration: 7100 ms Memory Size: 512 MB Max Memory Used: 153 MB Init Duration: 2881.51 ms

START RequestId: 831cd727-2ce0-4918-a1c5-db5b95a64d19 Version: \$LATEST

Cloud Watch Event is [{account: region: ap-southeast-2, detail: {jobName=SalesJob, severity=INFO, state=SUCCEEDED, jobRunId=jr_ee951a50bf4493e39f5151b55317506c}}

FILE [data3.csv] MOVED TO SUCCEEDED STATE

Invoking [sales-inspect-ready-trigger] Lambda Function

Moving data3.csv file to SUCCEEDED directory & triggering call to sales-inspect-ready-trigger lambda

END RequestId: 831cd727-2ce0-4918-a1c5-db5b95a64d19

REPORT RequestId: 831cd727-2ce0-4918-a1c5-db5b95a64d19 Duration: 773.77 ms Billed Duration: 800 ms Memory Size: 512 MB Max Memory Used: 153 MB

16) Let's check logs for sales-inspect-ready-trigger lambda

17) Dynamo DB Job state changed to SUCCEEDED for data3.csv file.

jobs Close

Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Contributor Insights

Create item Actions

Scan: [Table] jobs: JOB_ID

Scan [Table] jobs: JOB_ID

+ Add filter

Start search

| JOB_ID | FILES | STATE |
|---|---------------------|-----------|
| jr_1005af71f5939621d497e85909726c756299a4b0383671a81ea4c4021de62c | data1.csv,data2.csv | SUCCEEDED |
| jr_ee951a50bf4493e39f5515b55317506dea7b984c8a26d86a2af2b16350d7d | data3.csv | SUCCEEDED |

18) 3 Files found in SUCCEEDED directory.

Amazon S3 > lambda-glue-batch > sales > succeeded

lambda-glue-batch

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions

Name

- data1.csv
- data2.csv
- data3.csv

19) Below screenshot shows all the data from the files is inserted into the database (5+7+7 = 19 Records)

| region | country | item type | sales channel | order priority | order date | order id | ship date | units sold | unit price | unit cost | total revenue |
|-------------------|---------------|-----------------|---------------|----------------|------------|-----------|------------|------------|------------|-----------|---------------|
| 1 Region | Country | Item Type | Sales Channel | Order Priority | Order Date | [null] | Ship Date | [null] | 9.33 | 6.92 | |
| 2 Sub-Saharan... | South Africa | Fruits | Offline | M | 7/27/2012 | 443368995 | 7/28/2012 | 1593 | | | |
| 3 Middle East... | Morocco | Clothes | Online | M | 9/14/2013 | 667593514 | 10/19/2013 | 4611 | 109.28 | 35.84 | |
| 4 Australia ... | Papua New ... | Meat | Offline | M | 5/15/2015 | 940995585 | 6/4/2015 | 360 | 421.89 | 364.69 | |
| 5 Sub-Saharan... | Djibouti | Clothes | Offline | H | 5/17/2017 | 880811536 | 7/2/2017 | 562 | 109.28 | 35.84 | |
| 6 Region | Country | Item Type | Sales Channel | Order Priority | Order Date | [null] | Ship Date | [null] | [null] | [null] | |
| 7 Europe | Slovakia | Beverages | Offline | L | 10/26/2016 | 174590194 | 12/4/2016 | 3973 | 47.45 | 31.79 | |
| 8 Asia | Sri Lanka | Fruits | Online | L | 11/7/2011 | 830192887 | 12/18/2011 | 1379 | 9.33 | 6.92 | |
| 9 Sub-Saharan... | Seychelles | Beverages | Online | M | 1/18/2013 | 425793445 | 2/16/2013 | 597 | 47.45 | 31.79 | |
| 10 Sub-Saharan... | Tanzania | Beverages | Online | L | 11/30/2016 | 659878194 | 1/16/2017 | 1476 | 47.45 | 31.79 | |
| 11 Sub-Saharan... | Ghana | Office Supplies | Online | L | 3/23/2017 | 601245963 | 4/15/2017 | [null] | [null] | [null] | |
| 12 Sub-Saharan... | Tanzania | Cosmetics | Offline | L | 5/23/2016 | 739008080 | 5/24/2016 | [null] | [null] | [null] | |

Successfully run. Total query runtime: 357 msec. 19 rows affected.

16. Performance Stats

Executed couple of runs on the sample data with Worker Type set to Standard.

| S. No | No. of Records | Start Time | End Time | Total Time |
|-------|----------------|------------|------------|------------|
| 1 | 100000 | 2:57:00 PM | 3:10:00 PM | 13 Minutes |
| 2 | 100000 | 3:17:00 PM | 3:20:00 PM | 3 Minutes |
| 3 | 100000 | 3:30:00 PM | 3:33:00 PM | 3 Minutes |
| 4 | 1500000 | 4:17:00 PM | 4:29:00 PM | 12 Minutes |
| 5 | 1500000 | 4:32:00 PM | 4:33:00 PM | 1 Minute |
| 6 | 1500000 | 4:37:00 PM | 4:38:00 PM | 1 Minute |
| 7 | 1500000 | 4:42:00 PM | 4:54:00 PM | 12 Minutes |
| 8 | 1500000 | 4:58:00 PM | 4:59:00 PM | 1 Minute |

Observations:

- From instances #1,4,7 took more time as Glue infra is warmed up.
- During #2,3,5,6,8 instances, Glue was already warm up, hence took less time for the execution

17. Troubleshooting & Learnings

| S. No | Scenario | Exception / Description | Reference |
|-------|---|--|---|
| 1 | When AWS Glue is processing the file, it is throwing exceptions | com.amazon.ws.emr.hadoop.fs.shaded.org.apache.http.conn.HttpHostConnectException: Connect to 167.254.77.1:8088 [/167.254.77.1] failed: Connection refused (Connection refused) | Modify Inbound Outbound Rules |
| 2 | AWS Glue is not able to connect to RDS Postgres | VPC S3 endpoint validation failed for SubnetId: subnet-xxxxx. VPC: vpc-xxxxx. Reason: Could not find S3 endpoint or NAT gateway for subnetId: subnet-xxxx in Vpc vpc-xxxxx. | Setup Amazon S3 VPC Endpoint |
| 3 | Unable to find Event Class in Java for CloudWatch Event Type | | AWS Lambda - CloudWatch Event type |
| 4 | There is no trigger action from one Lambda to call another Lambda | | AWS Lambda to Lambda Trigger using Custom Classes |