Lambda Glue Architecture for Batch Processing Documentation

Table of Contents

1. Use case

To migrate the data from on prem servers to AWS.

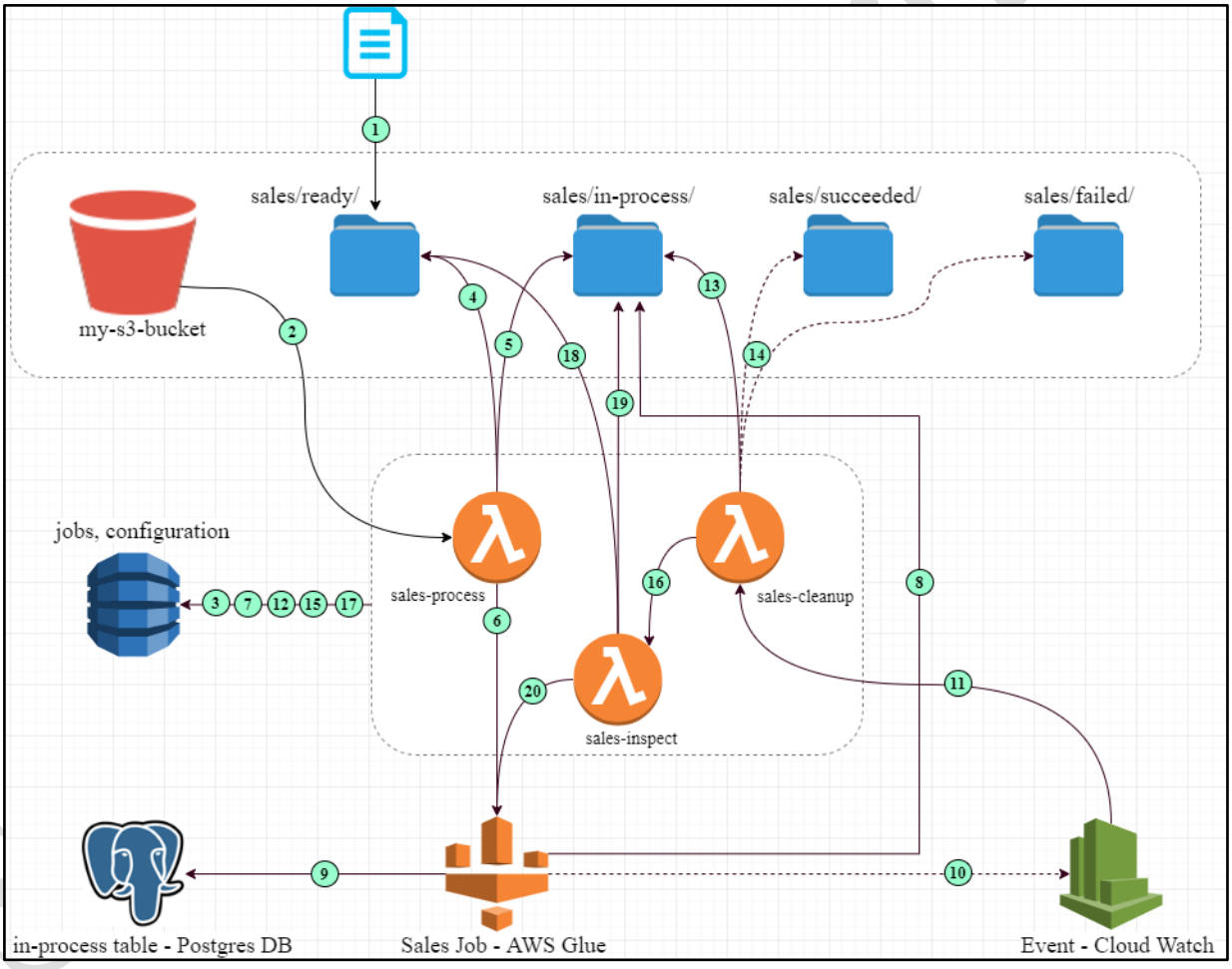
2. Project Description

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.

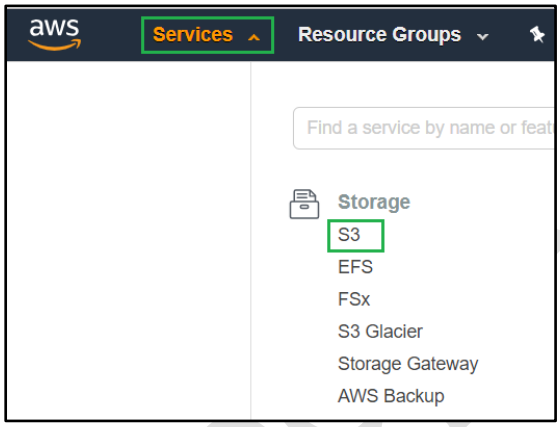
6. Lambda Glue Batch Build Steps

To run the application, follow below steps.

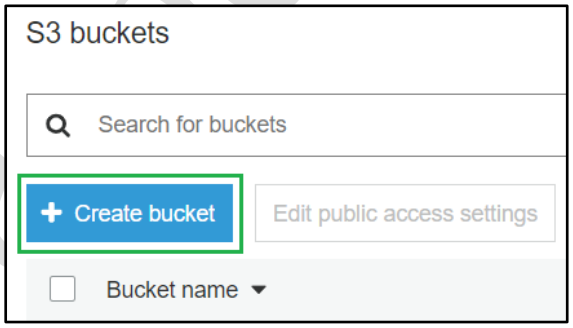
1) Create S3 bucket
2) Lambda configuration in Dynamo DB
3) Create RDS Postgres database
4) Create Roles
5) Setup Glue Job
6) Setup Cloud Watch Event Rule
7) Deploying Lambda's
8) Testing
9) Performance Statistics
10) Troubleshooting & Learnings

7. Create S3 bucket

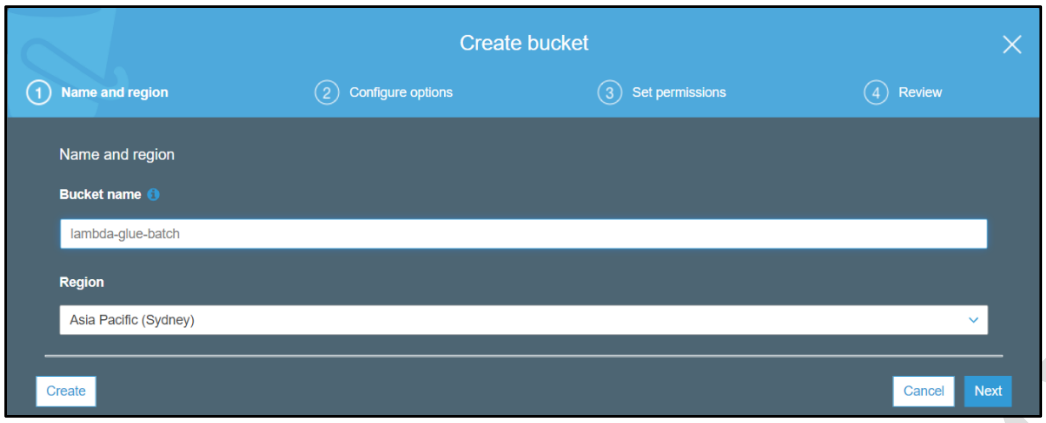
1) Click on Services → Storage → S3



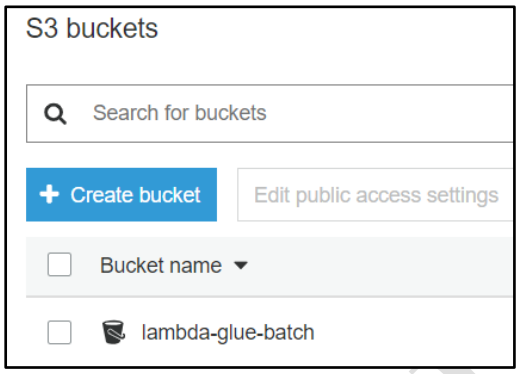
2) Click on create bucket



3) Provide a bucket name



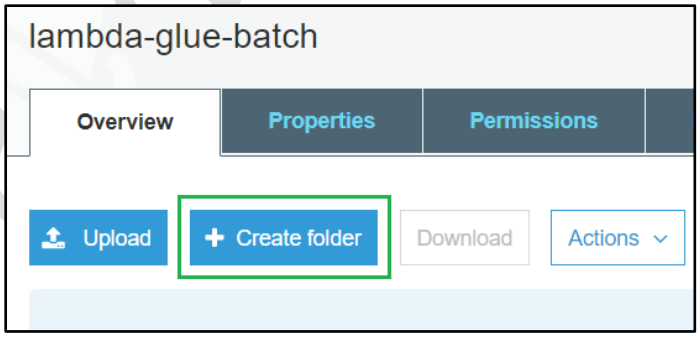
4) Bucket gets created as shown below.



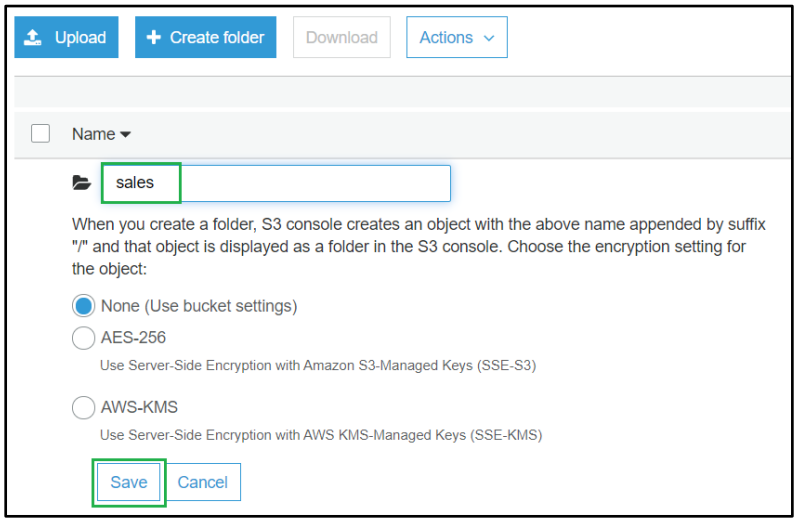
5) Inside the bucket create the below directory structure

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

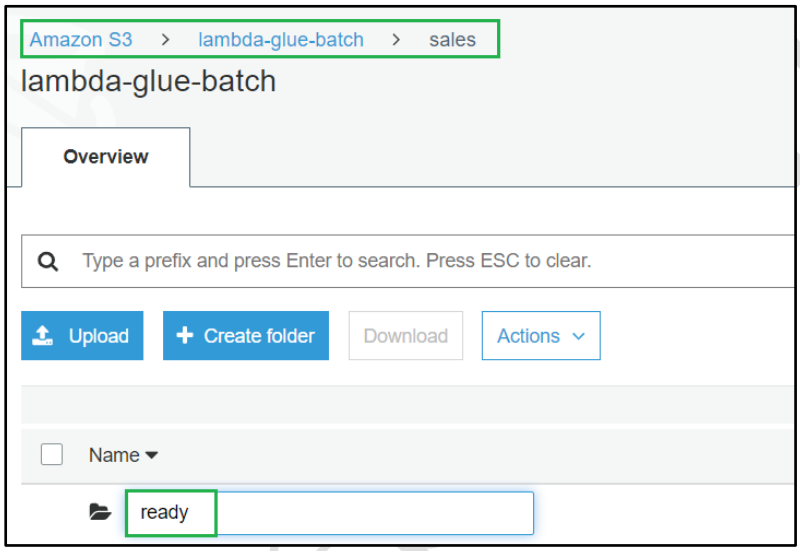
6) To create the above structure, click on create folder



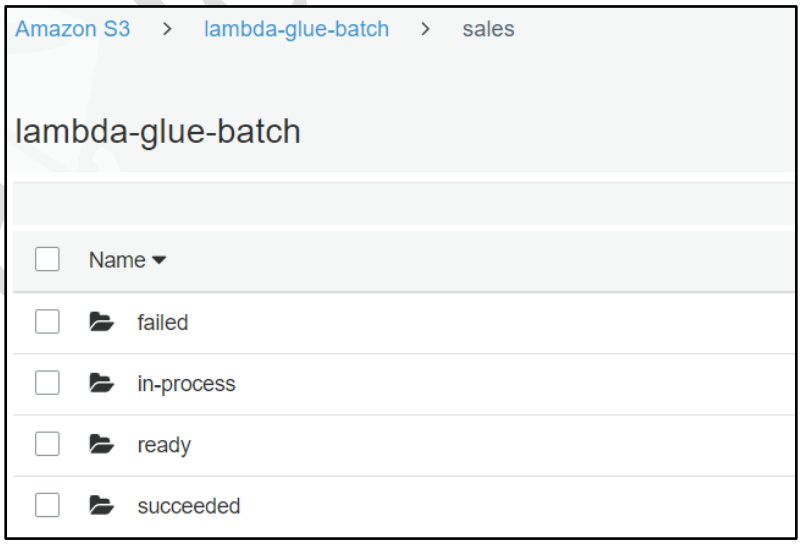
7) Provide **"sales"** as directory path



8) Create "**ready**" directory inside sales directory



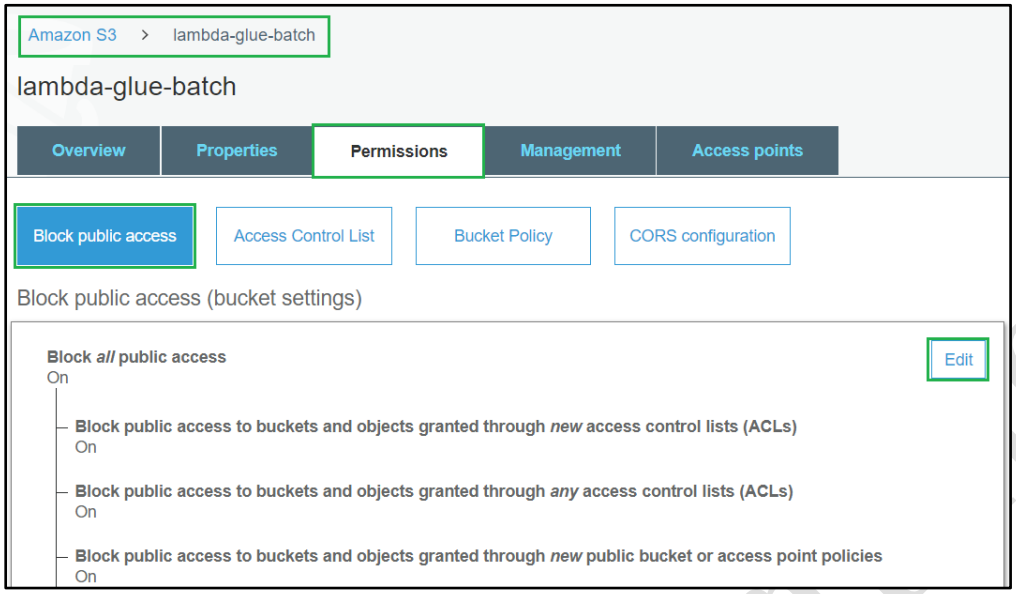
9) Similarly, repeat step 7 and 8 to create in-process, succeeded, failed directories.



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

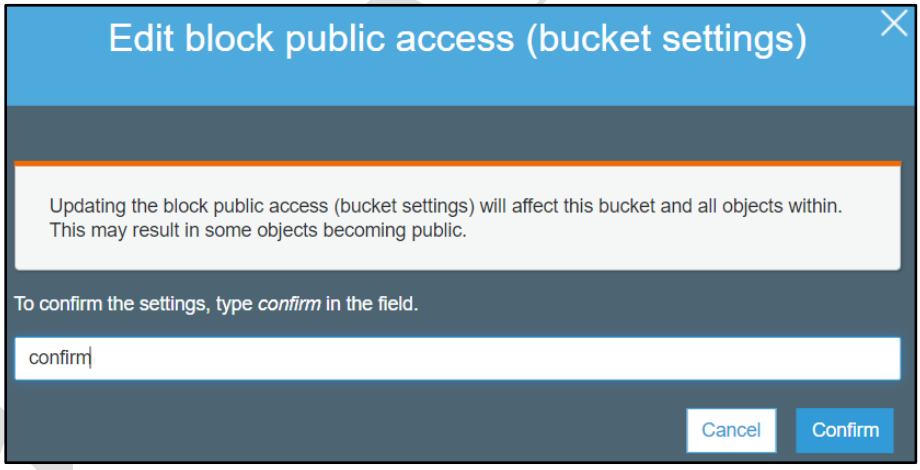
Note: This setting is for demo purpose. Do not use this setting in production.

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

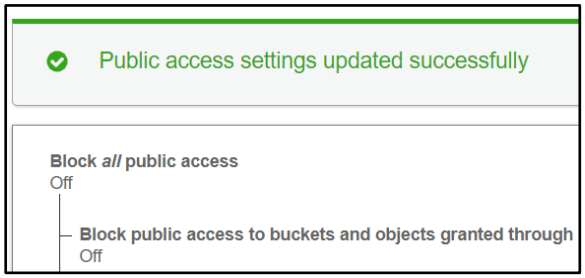


11) Uncheck Block all public access and click save

12) Confirm the unblock

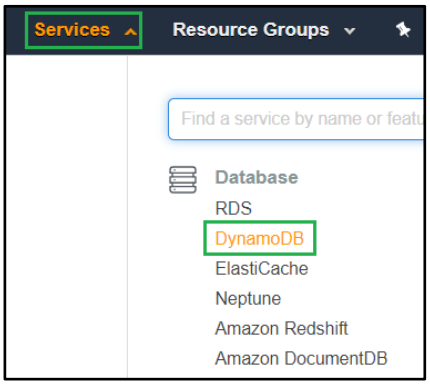


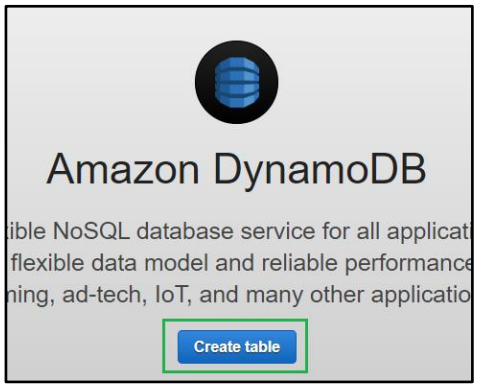
13) Settings will be updated



8. Lambda configuration in Dynamo DB

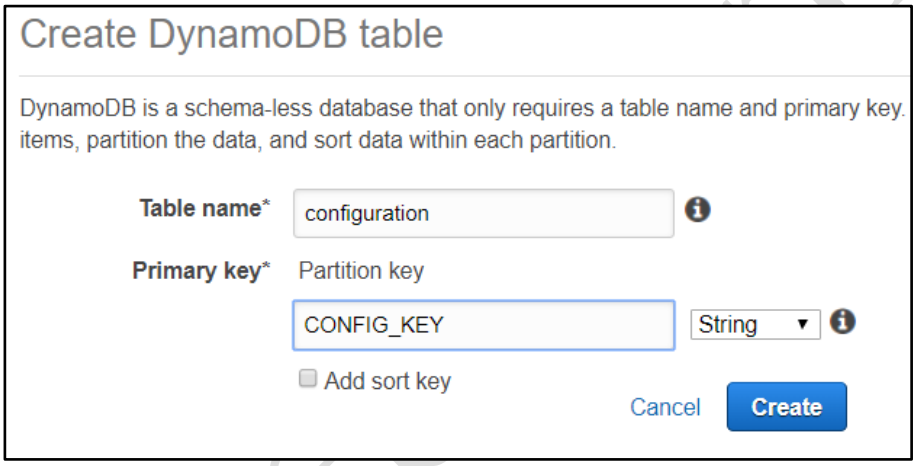
1) Go to Services → Database → DynamoDB, which allows to Create table



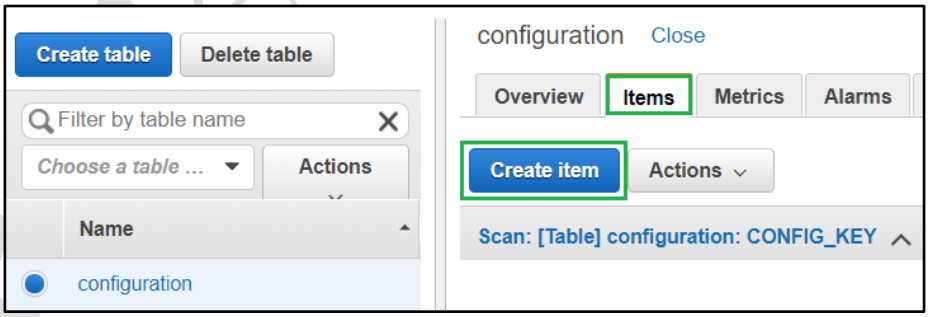


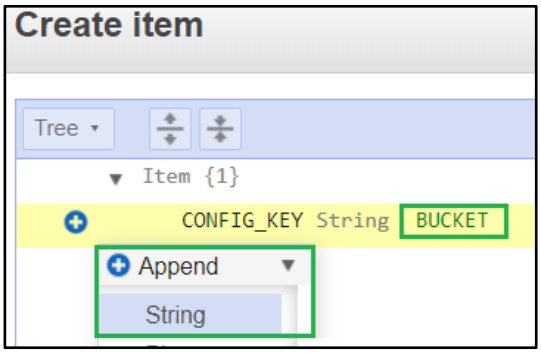
8.1. configuration table

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

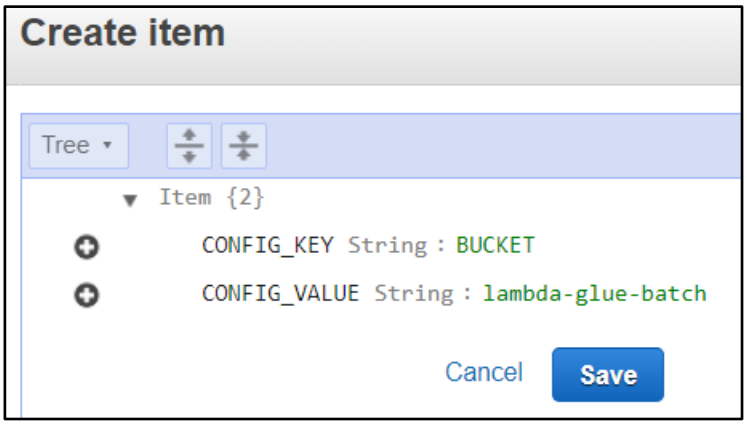


3) Table is created, now create item.





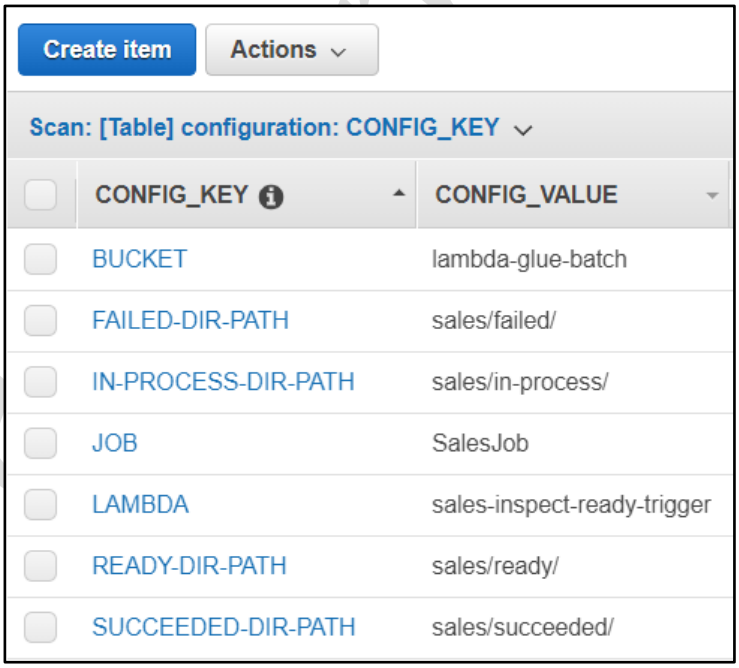
4) Add CONFIG_KEY & CONFIG_VALUE as shown below and save it.



5) Add below configuration values –

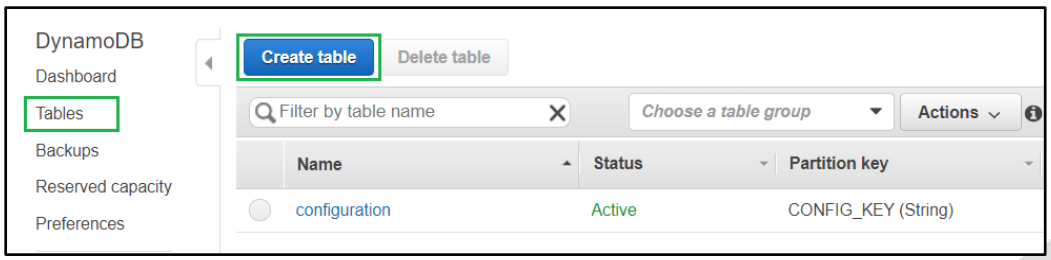
CONFIG_KEY	CONFIG_VALUE
BUCKET	lambda-glue-batch
READY-DIR-PATH	sales/ready/
IN-PROCESS-DIR-PATH	sales/in-process/
SUCCEEDED-DIR-PATH	sales/succeeded/
FAILED-DIR-PATH	sales/failed/
JOB	SalesJob
LAMBDA	sales-inspect-ready-trigger

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

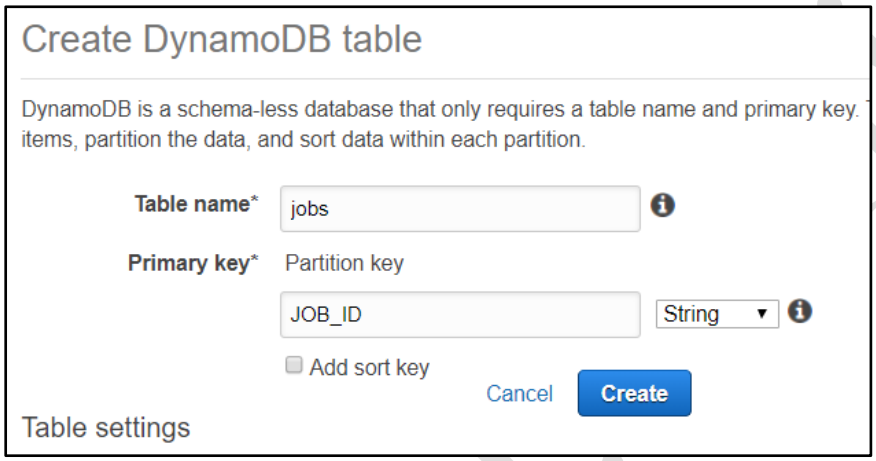


8.2. jobs table

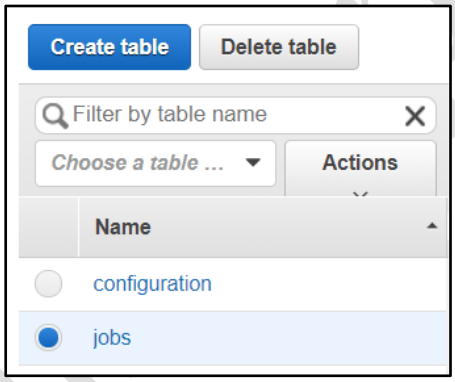
7) Click on Create table as shown below



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

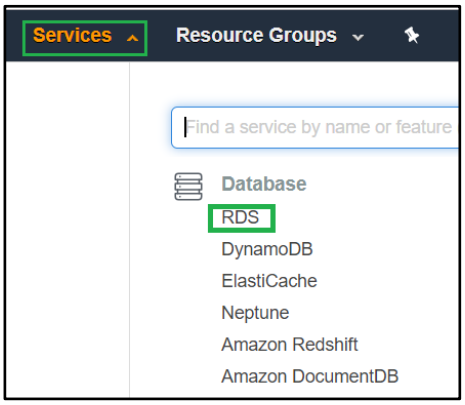


9) Jobs table would be created as shown below.

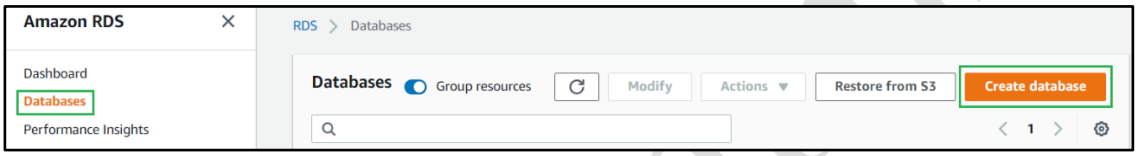


9. Create RDS Postgres database

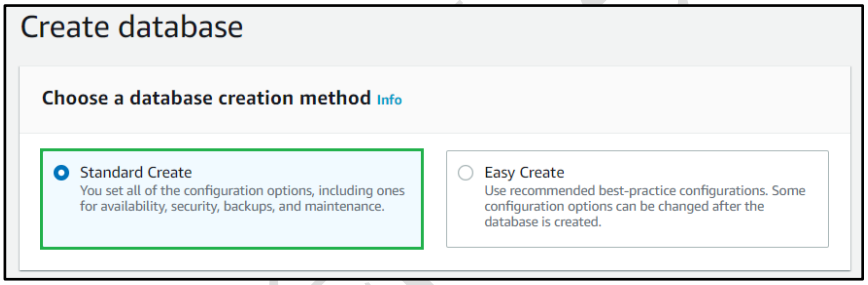
1) Click on Services → Analytics → AWS Glue

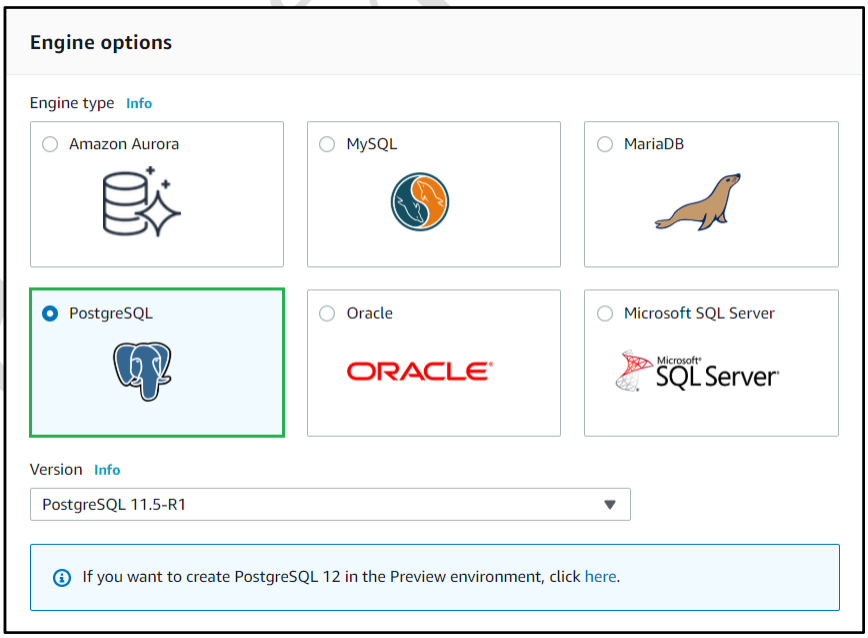


2) Go to databases and create database



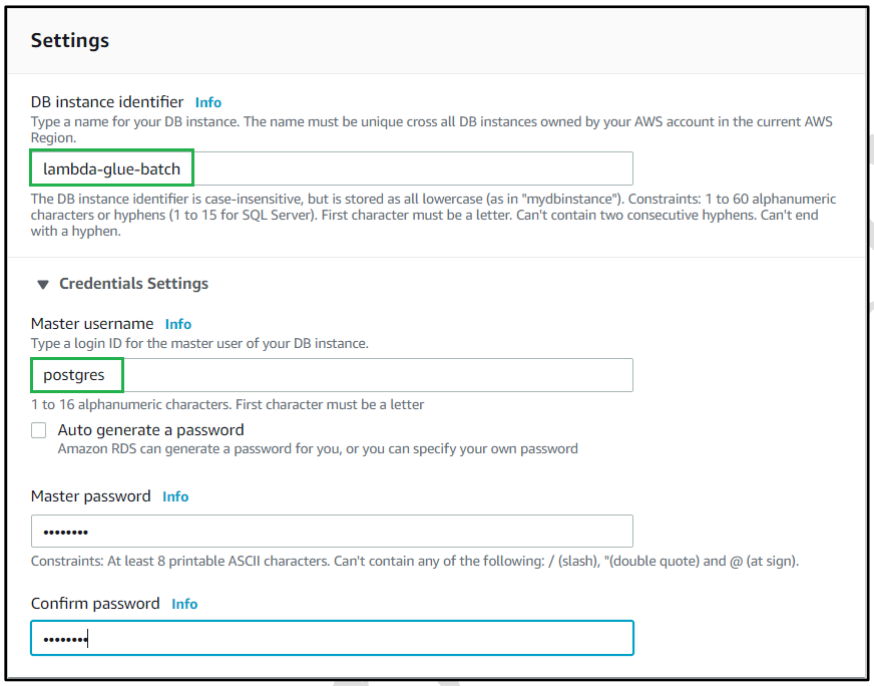
3) Choose Standard Create, PostgreSQL with the default version





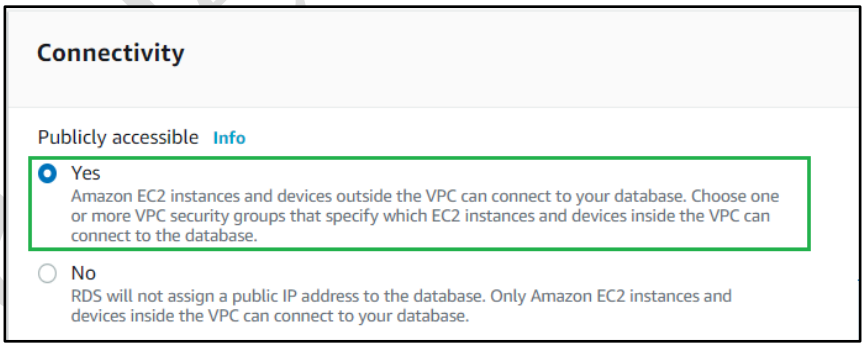
4) Choose Free tier from Templates section

5) Fill the settings section as shown

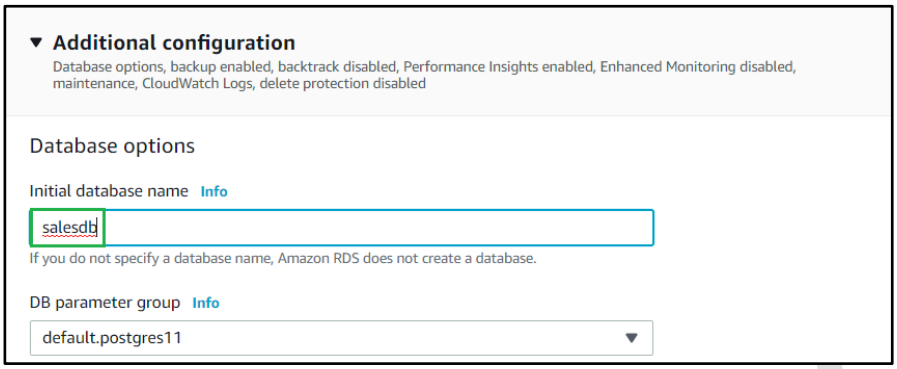


6) From connectivity section, do the below setting as in order to make the DB accessible on your PC.

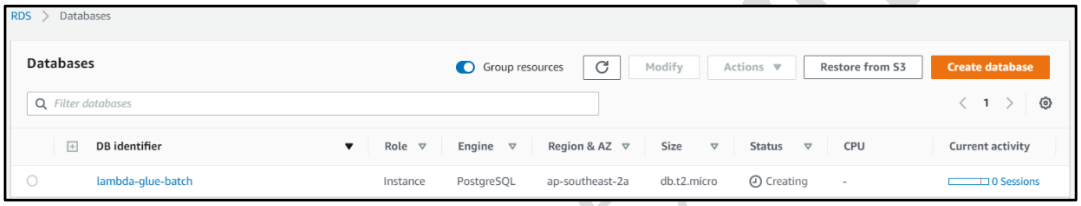
Note: Do not do this setting for your production DB's. This is just for Demo.



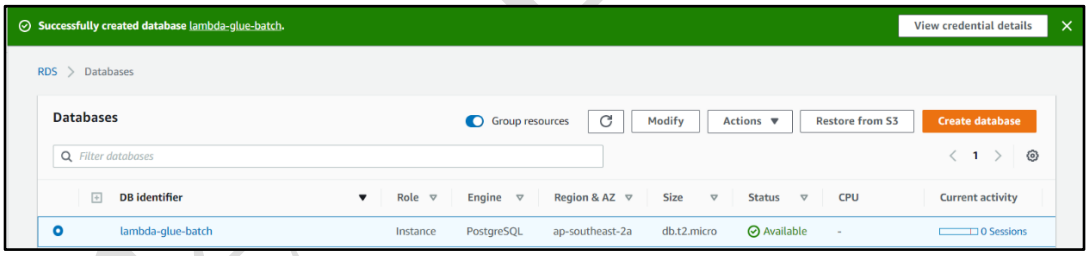
7) In the Additional configuration section, Provide Database name in Database options section and leave rest to default values.



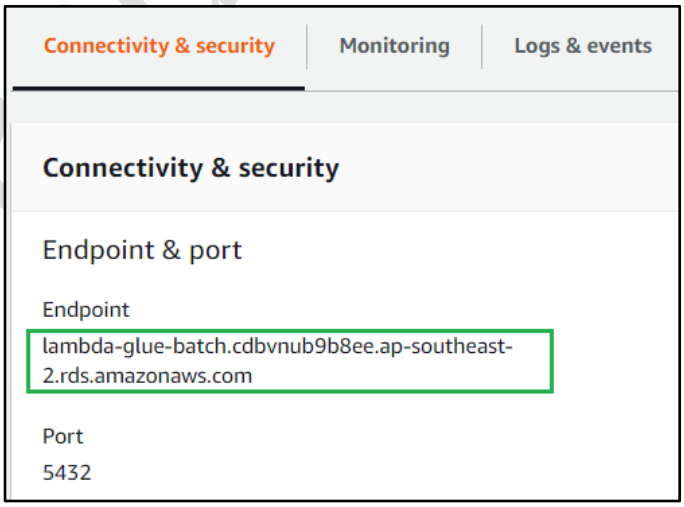
8) It takes few minutes to create the database and shows status as **creating**



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

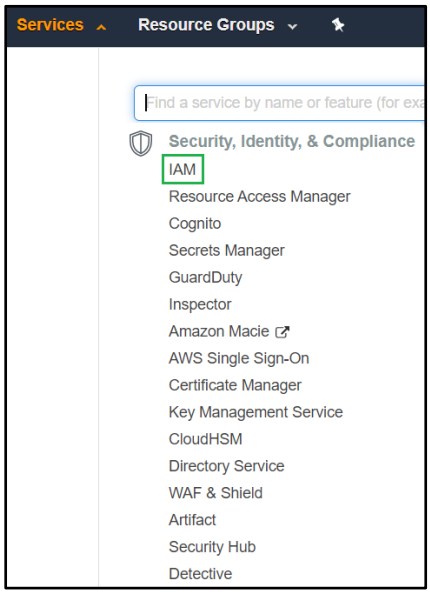


10) Check for the database endpoint.



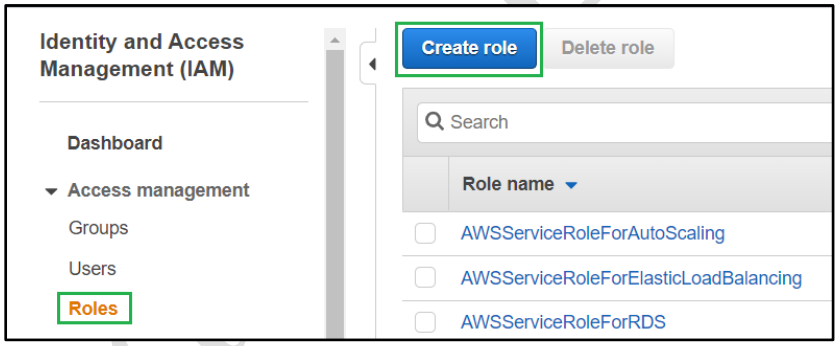
10. Create Roles

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

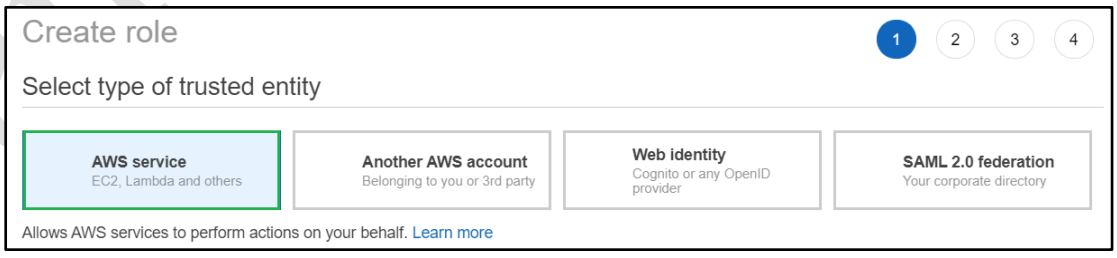


10.1. Glue Role

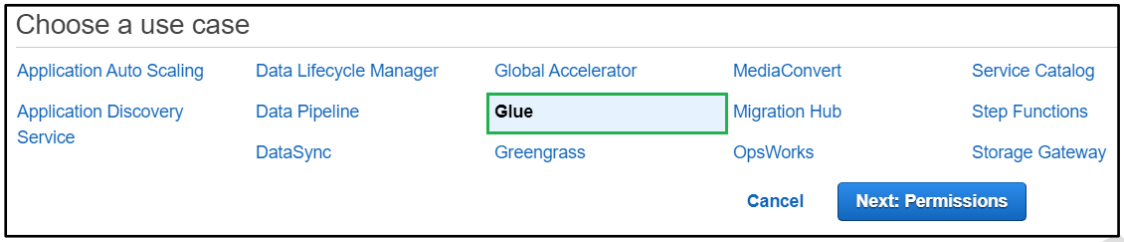
2) As per the Architecture diagram, Glue needs access to **S3**, **RDS** and **Glue** Service Role. Click on Roles on the left pane and click Create role



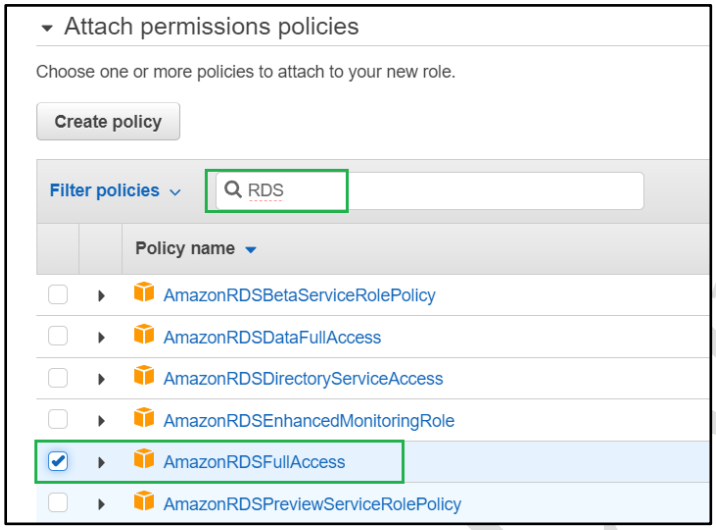
3) Select AWS service as shown below.



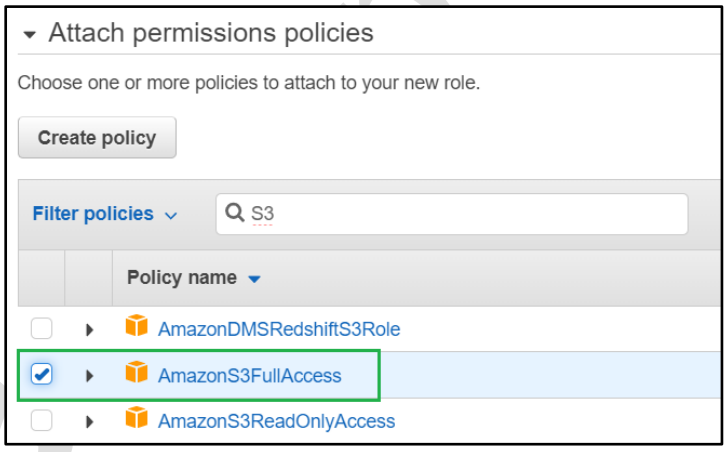
4) Choose Glue use case



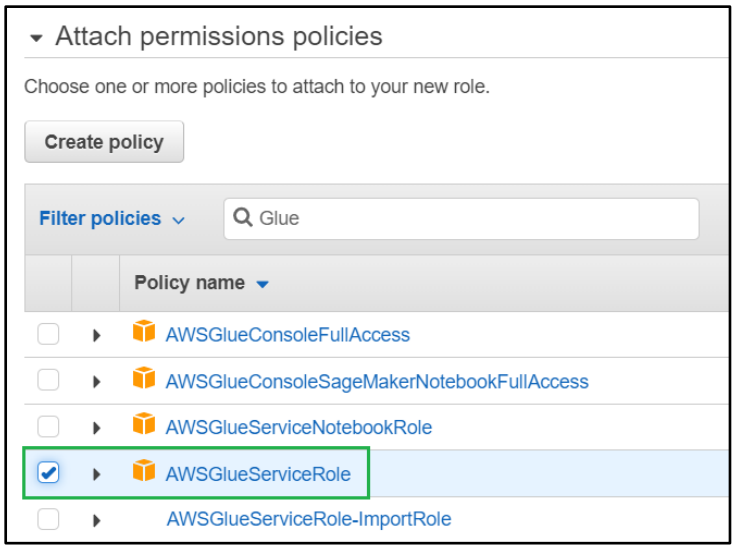
5) Search & Attach AmazonRDSFullAccess Policy



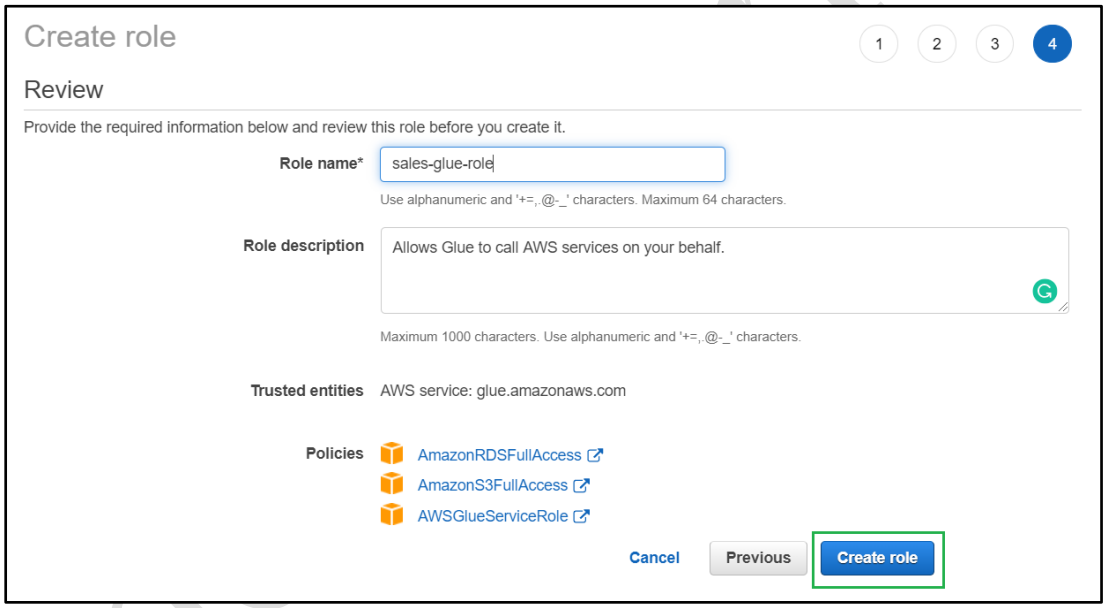
6) Search & Attach AmazonS3FullAccess Policy



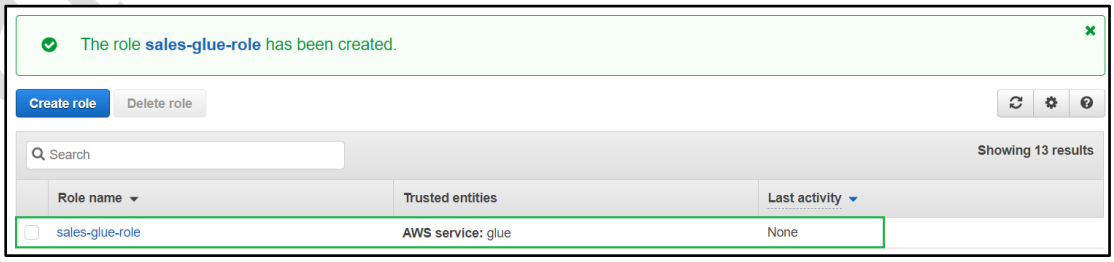
7) Search & Attach AWSGlueServiceRole Policy, Click next and Review.



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



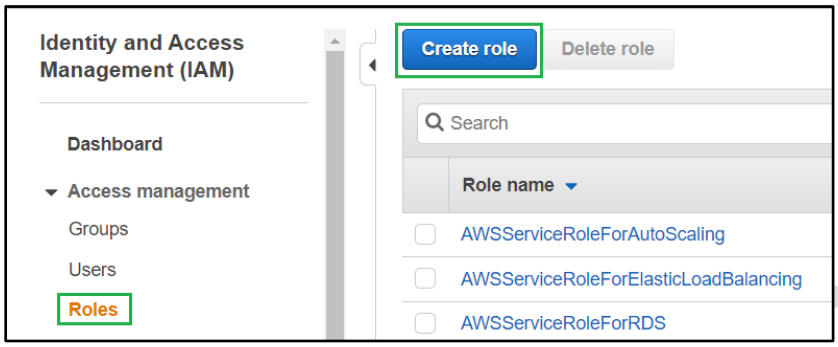
9) It creates and lists the role as shown below.



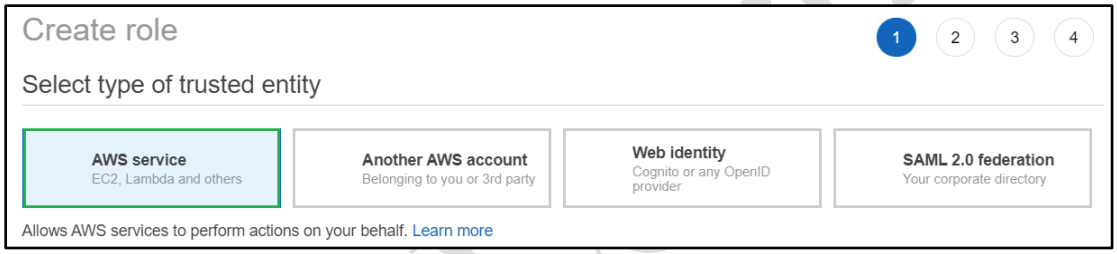
10.2. Lambda Role

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

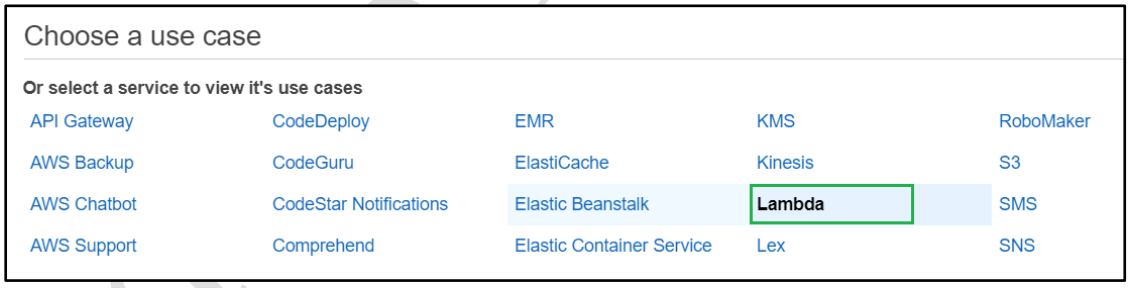
11) Create Role



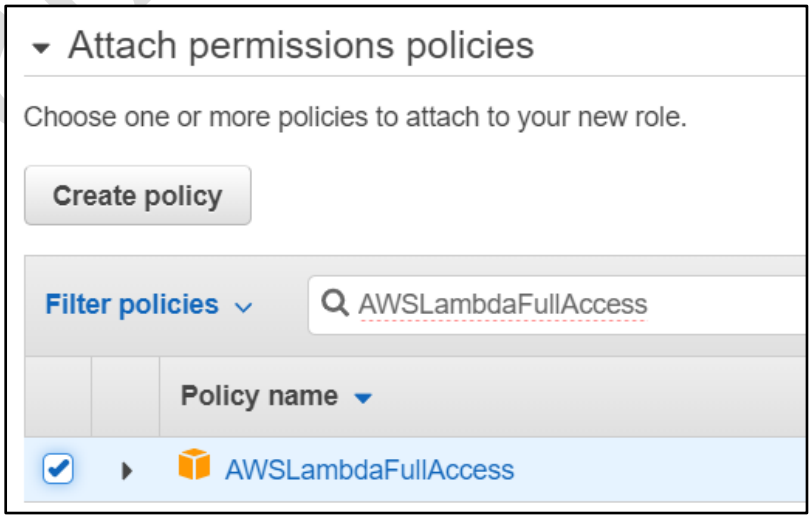
12) Select AWS service as shown below.

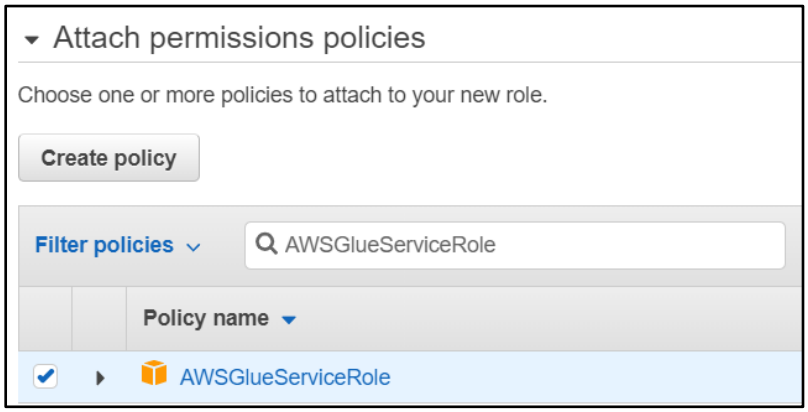


13) Choose Lambda use case

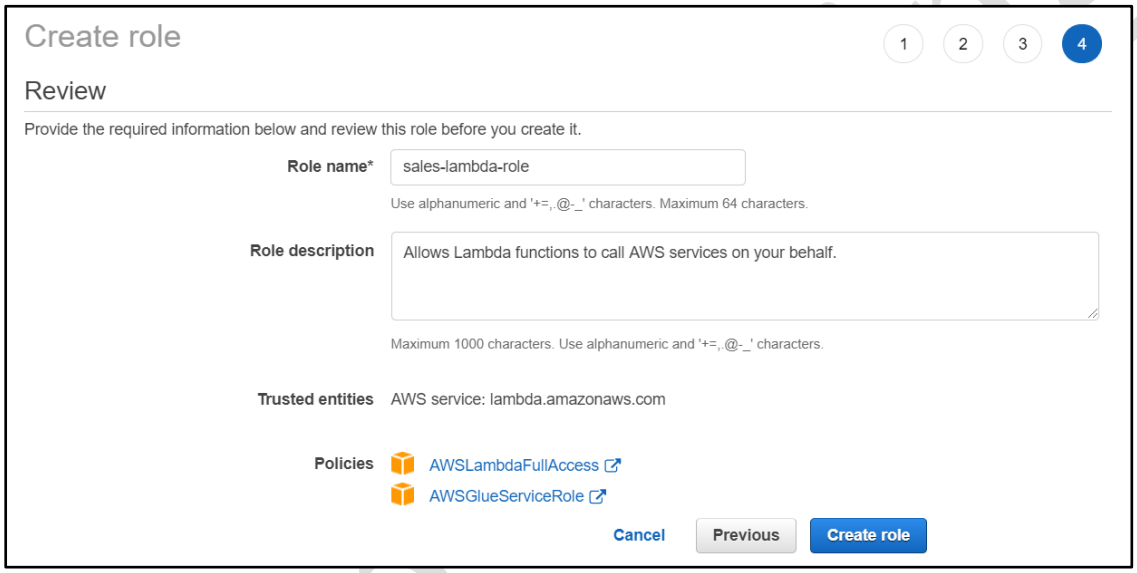


14) Search & attach AWSLambdaFullAccess & AWSGlueServiceRole Policies

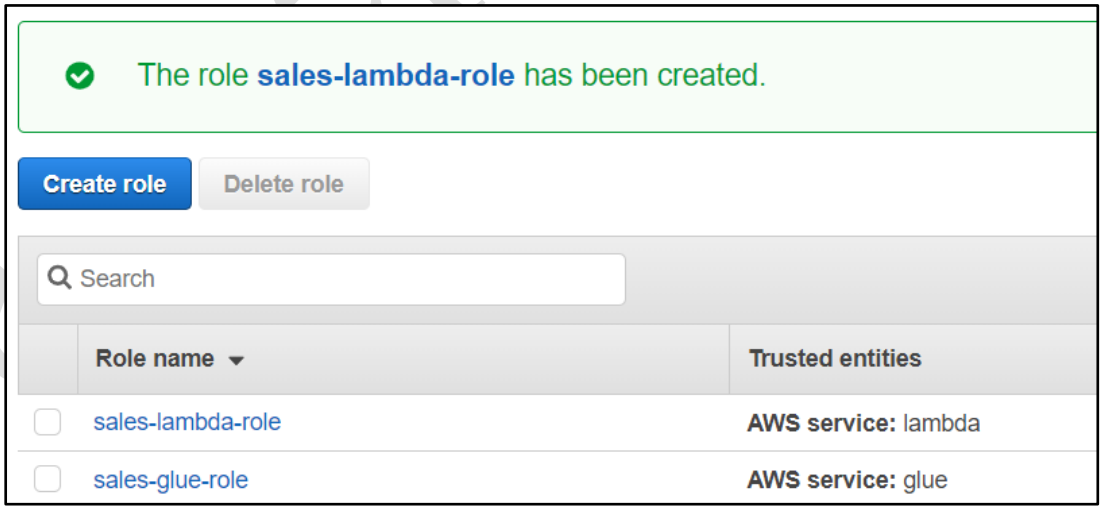




15) Give role name – sales-lambda-role and click create role

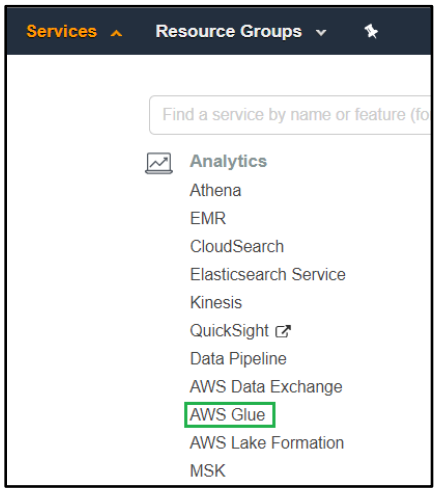


16) It creates role and lists the role as shown below.



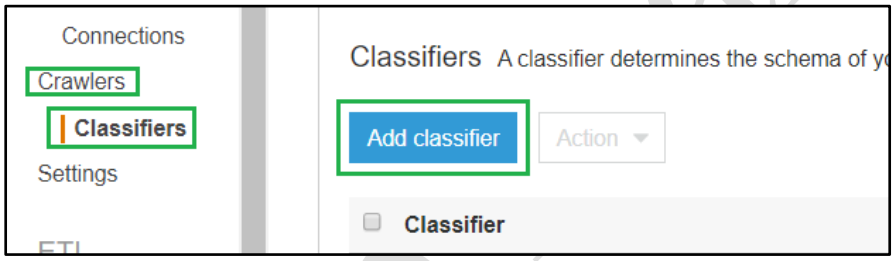
11. Setup Glue Job

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

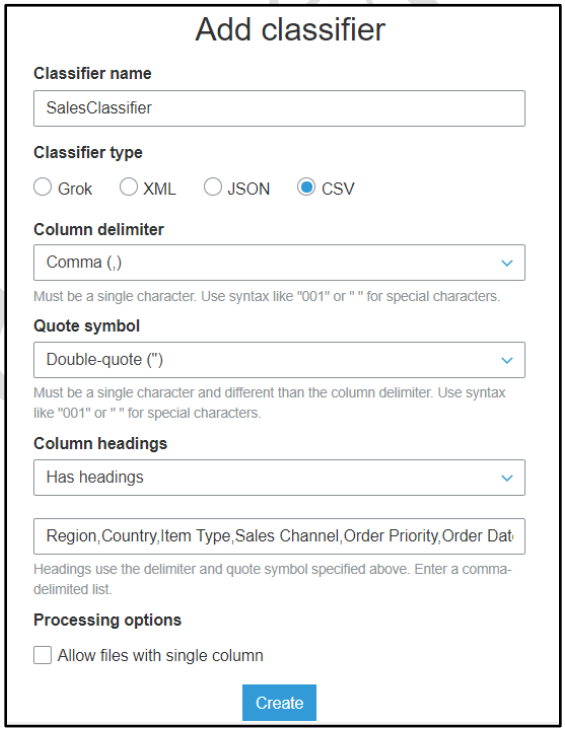


11.1. Create Classifier

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



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

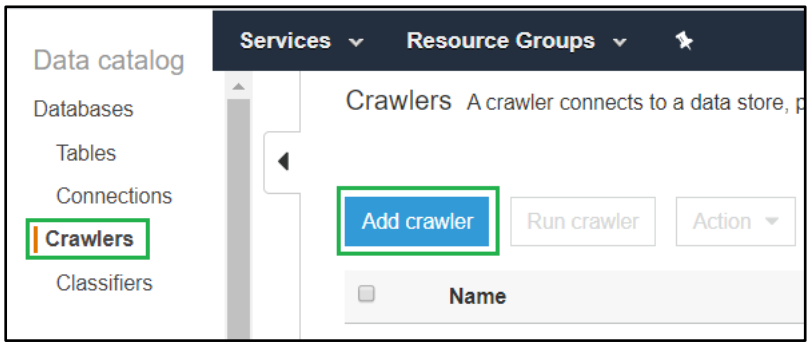


Headings specified is taken from my sample data used. Below is the heading data for reference.

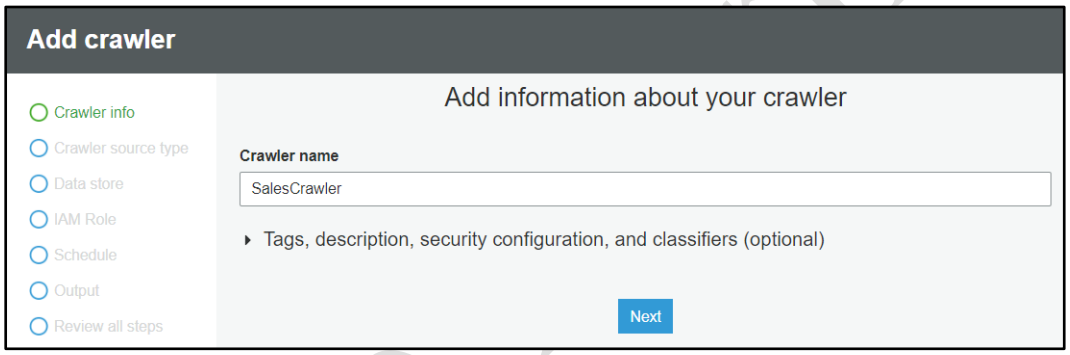
Region,Country,Item Type,Sales Channel,Order Priority,Order Date,Order ID,Ship Date,Units Sold,Unit Price,Unit Cost,Total Revenue,Total Cost,Total Profit

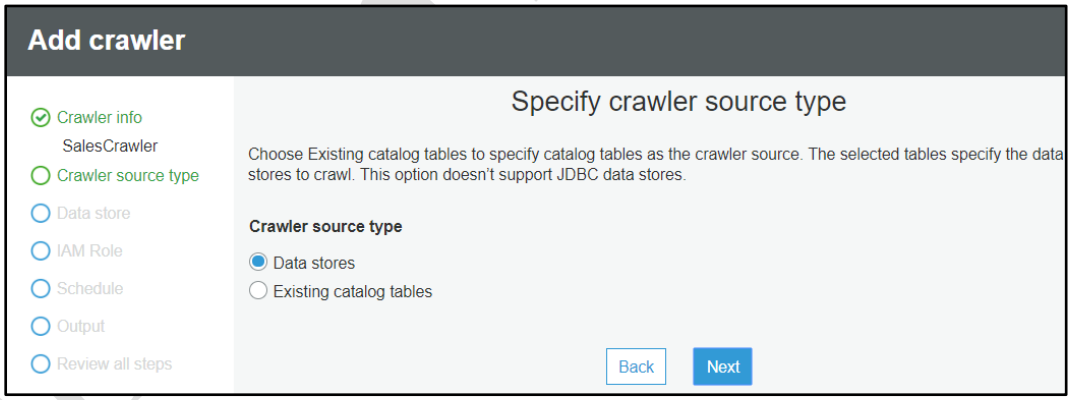
11.2. Create Crawler

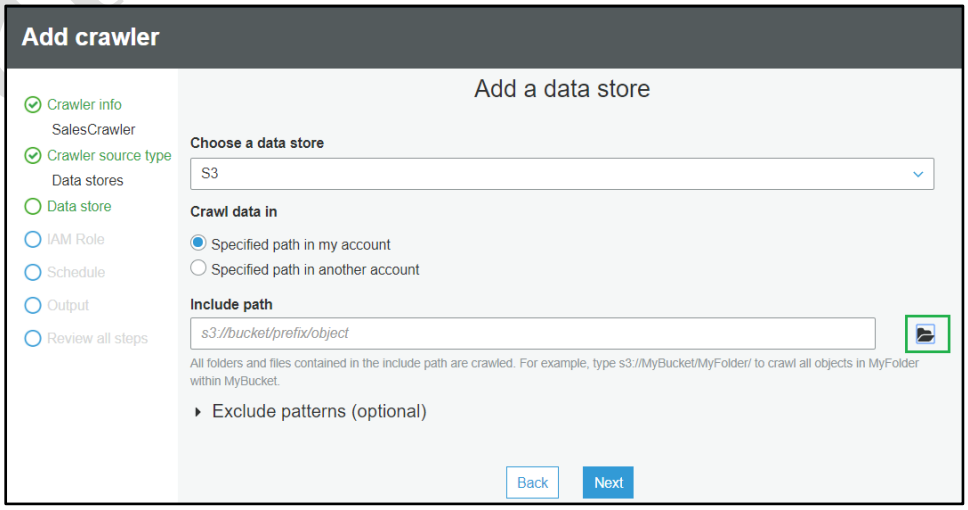
4) Now click on Crawlers → Add Crawler



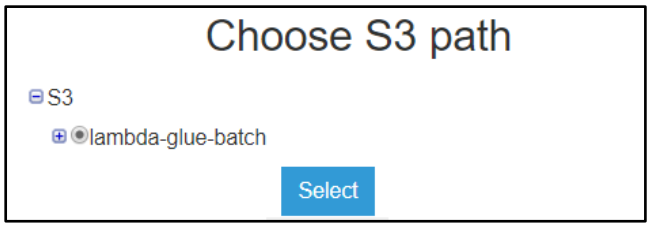
5) Provide Crawler information

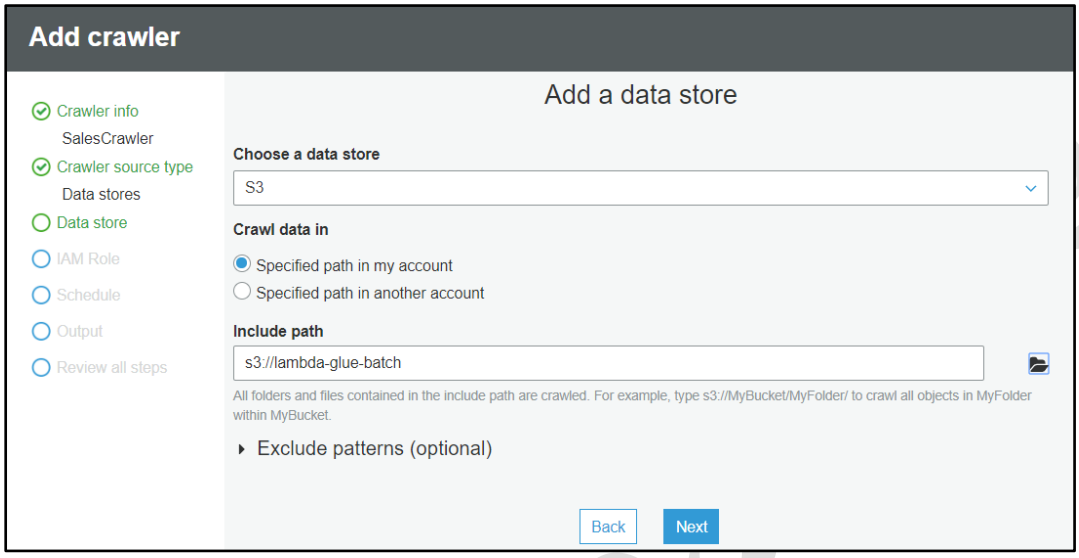


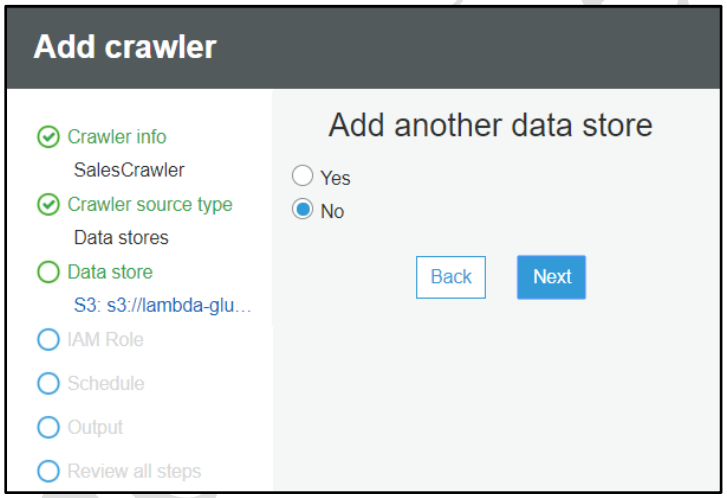




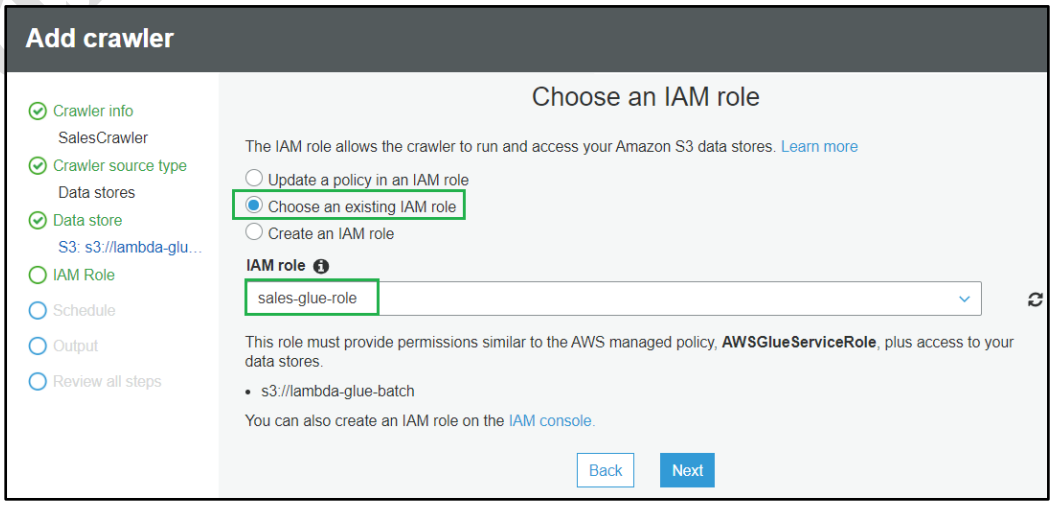
Click on the folder shown on include path, it opens up a pop-up screen, select the created bucket.



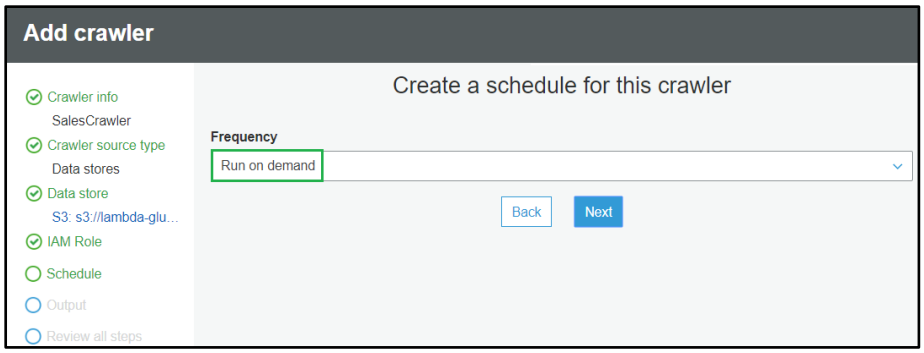




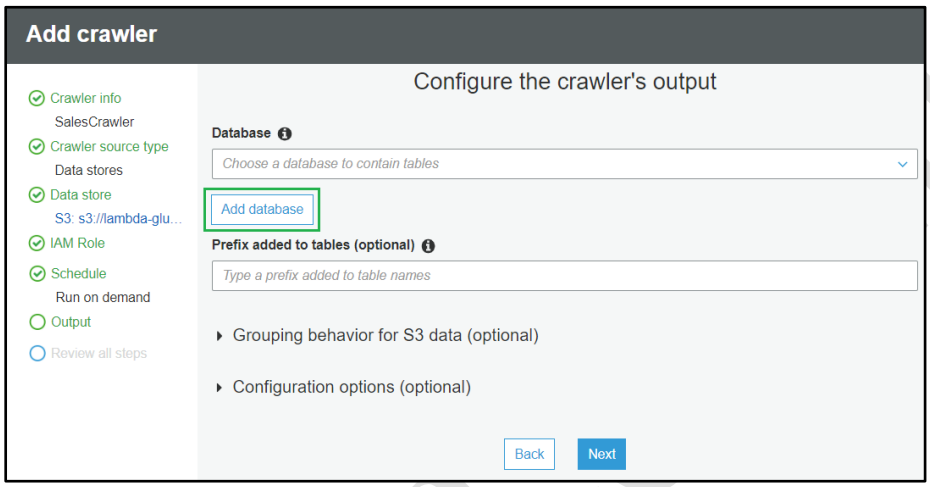
6) Choose the role which was created in earlier section – "**sales-glue-role**"



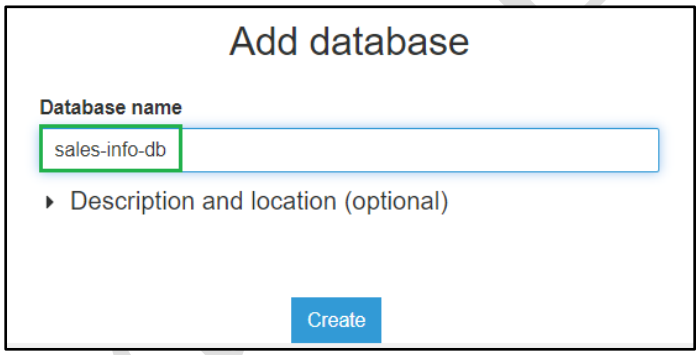
7) Set frequency to run on demand. As per the architecture lambda invokes when required.



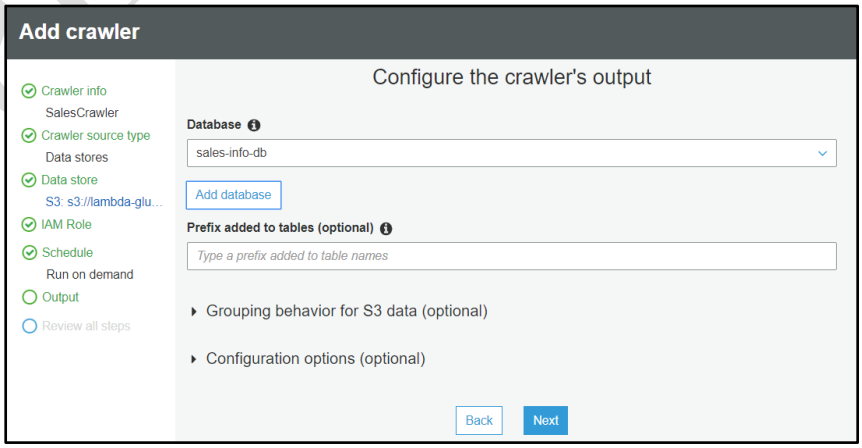
8) Click next and Add database



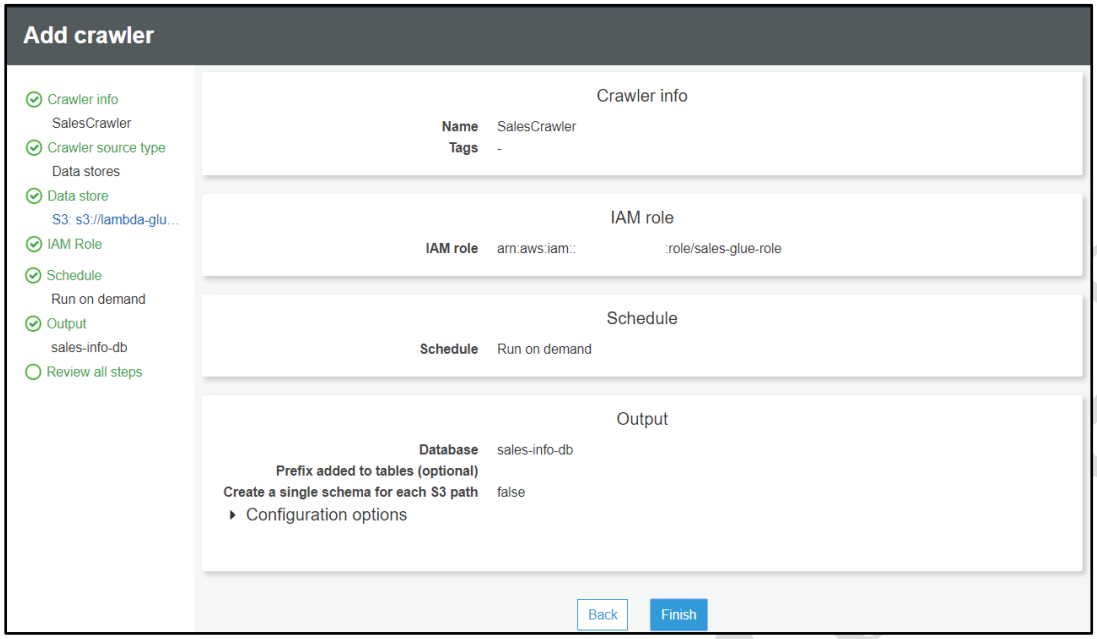
9) It opens up a pop-up, provide database name (Not the one created for RDS)



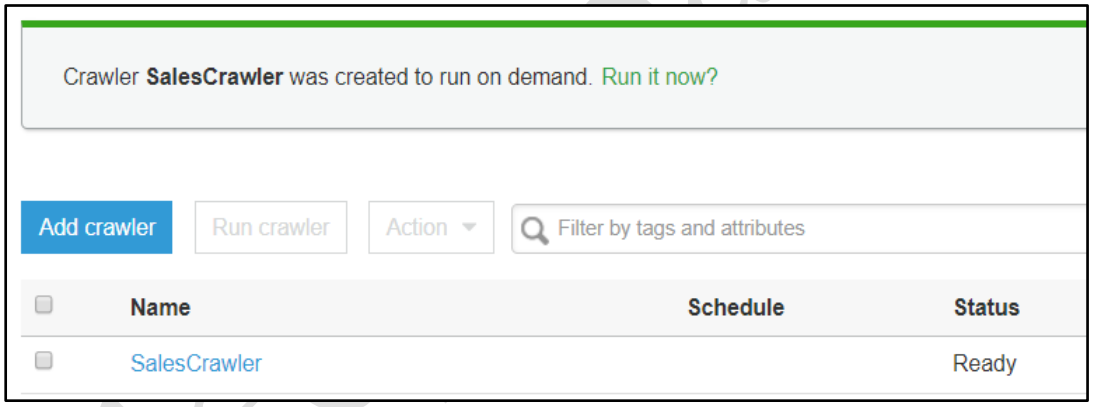
10) Click next



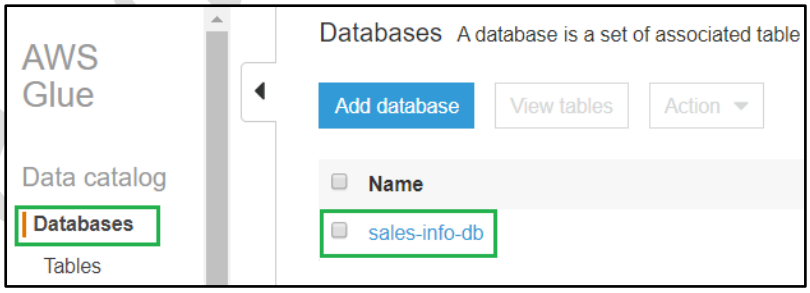
11) Review details and Click Finish.



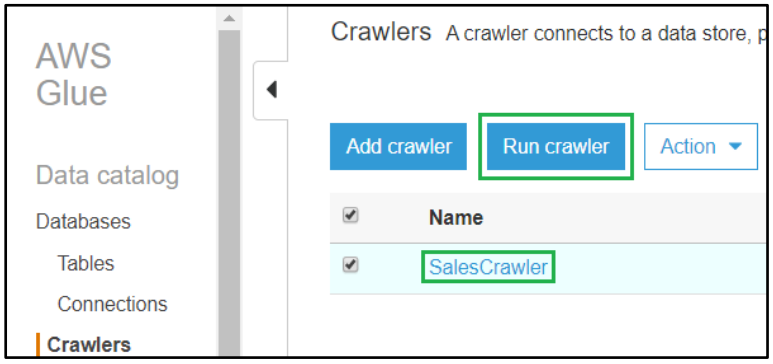
12) Sales Crawler created as follows.



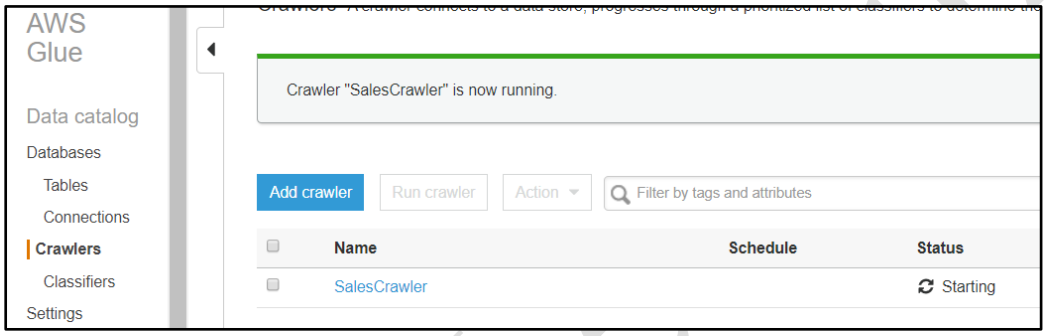
13) Database which is created in Step 9 could be seen on the Databases tab as shown below.



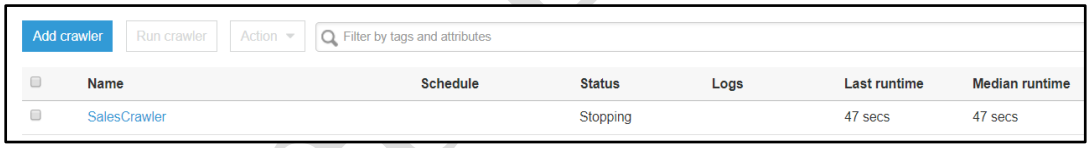
14) To create the table for sales-info-db, either run the created crawler or create table manually. In this demo, lets run the crawler.



15) Crawler starts running

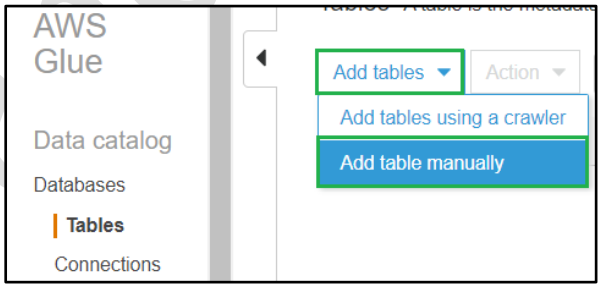


16) Crawler ran for 47 Secs and Stopped.



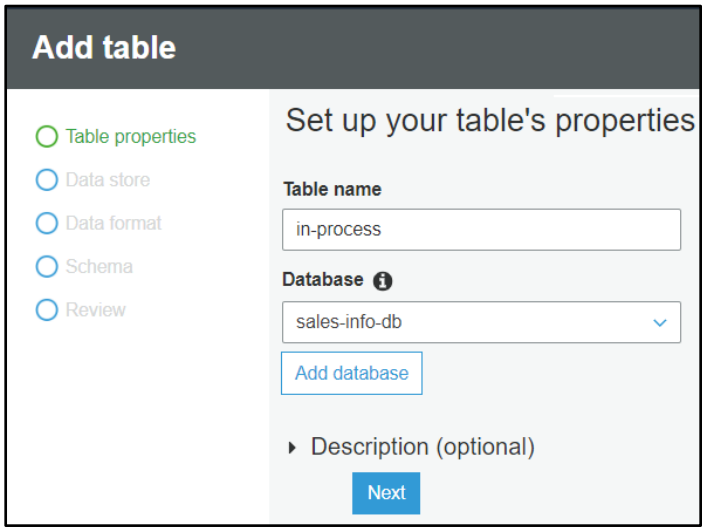
11.3. Create Table

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

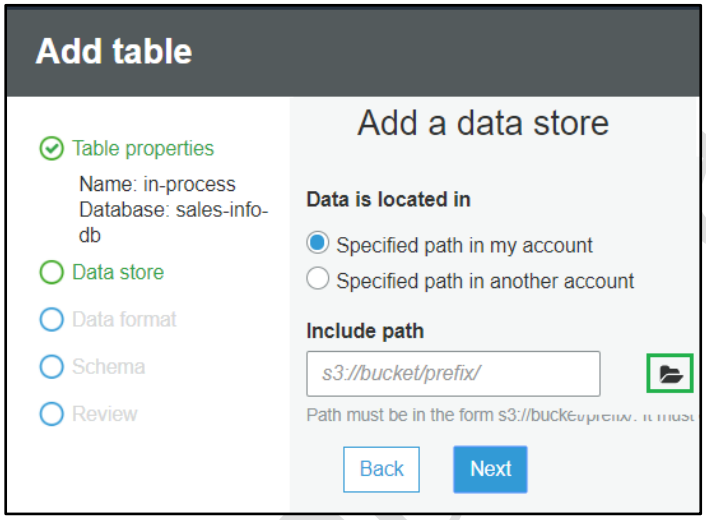


18) Setup table properties

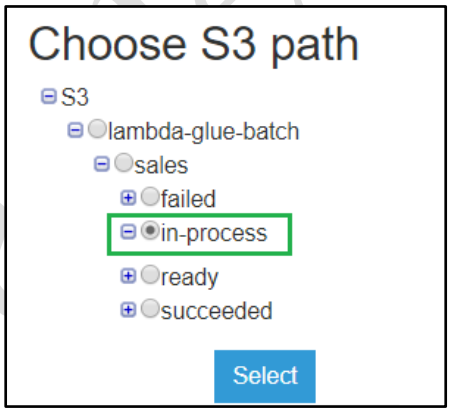
*Note: Table name is provided as **in-process** as crawler will read the data from in-process folder of s3 bucket.*

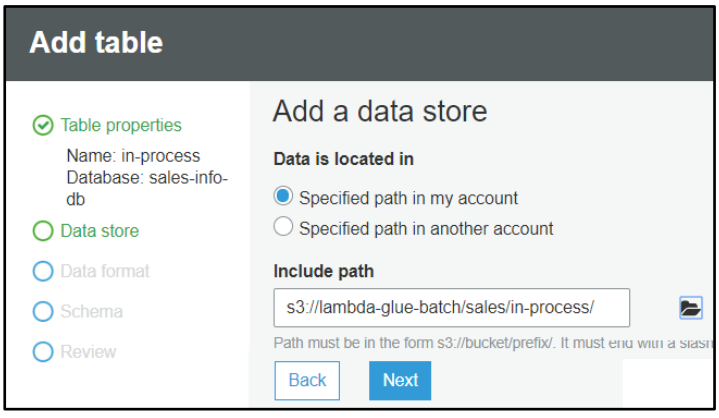


19) Provide S3 directory to crawl, click on folder, it opens a pop up

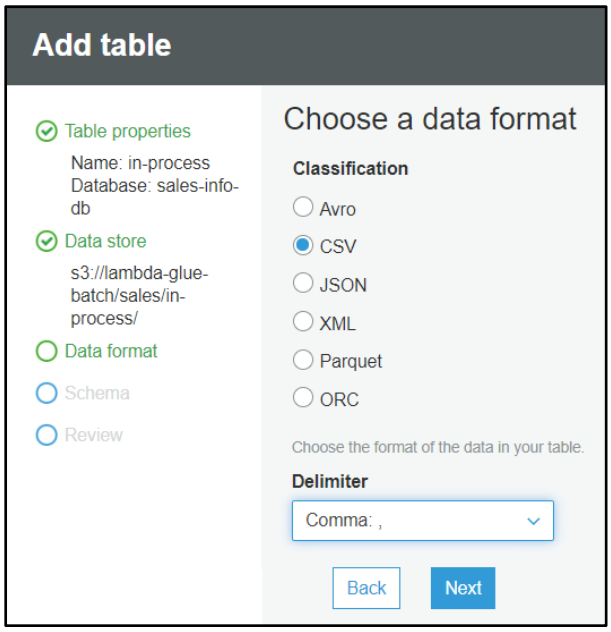


20) Select the directory to crawl

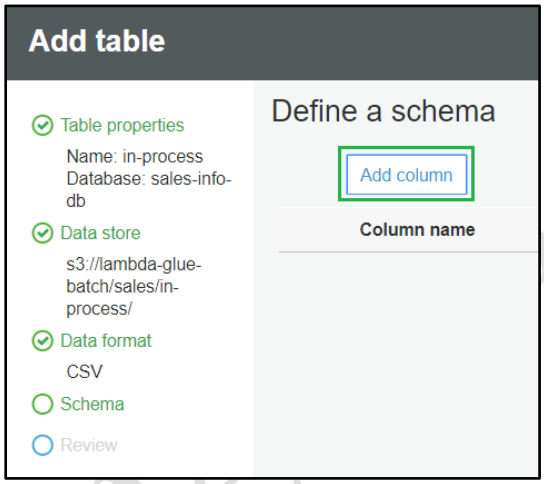


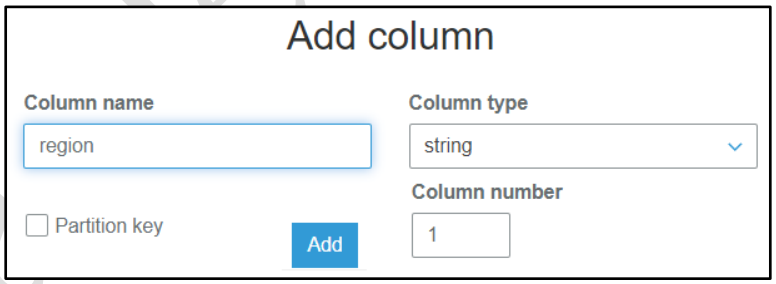


21) Choose data format

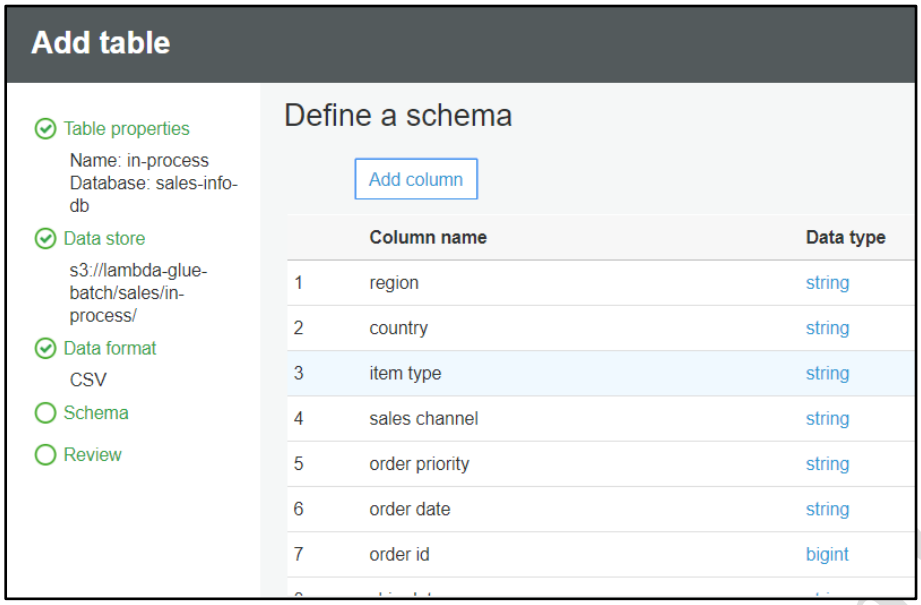


22) Define schema, click on Add column.

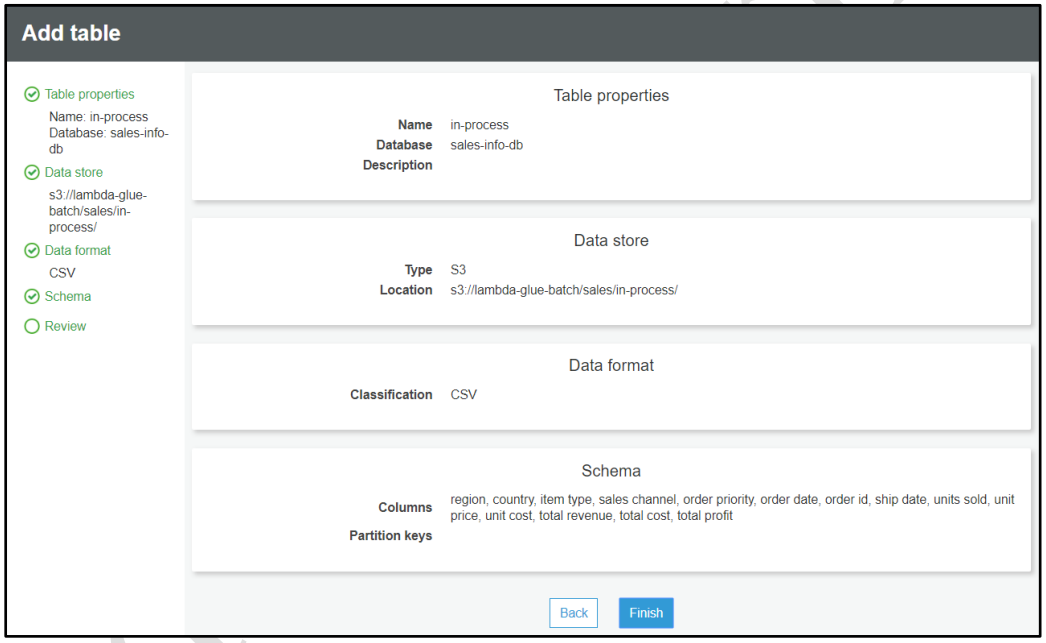




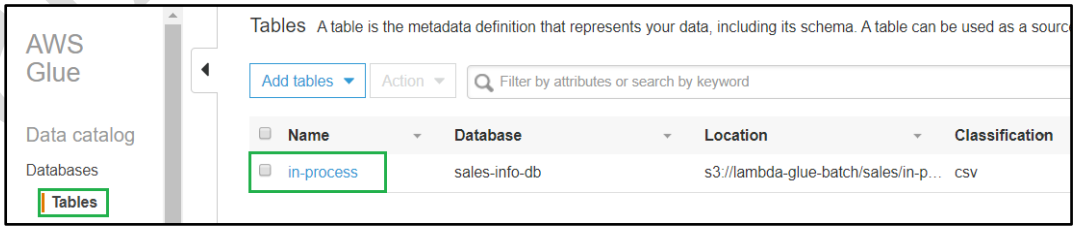
Similarly, add all the columns.



23) Click next → Review → Finish

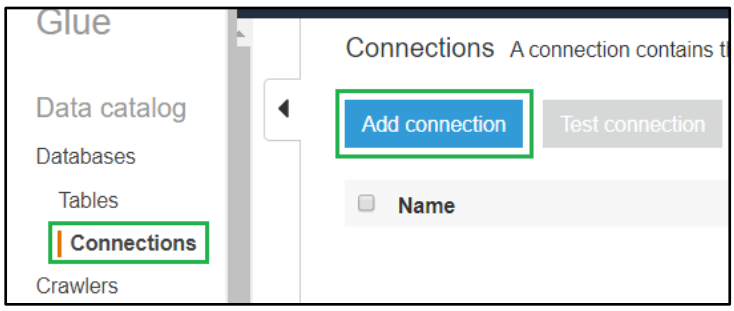


24) Table is created.

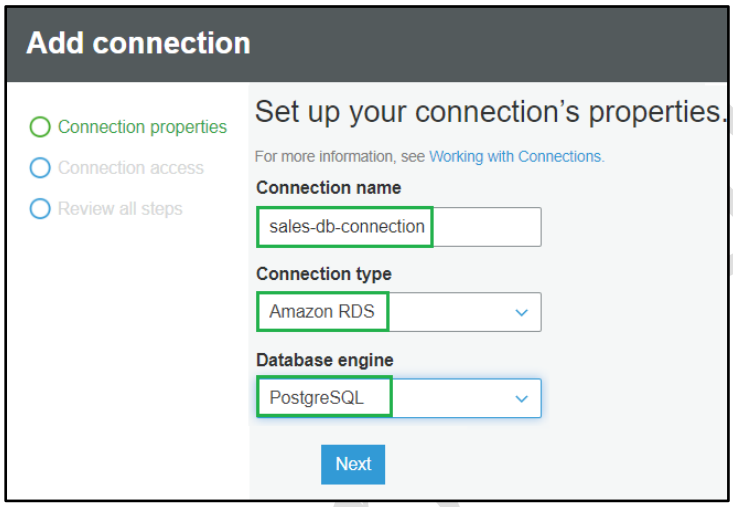


11.4. Create Connection

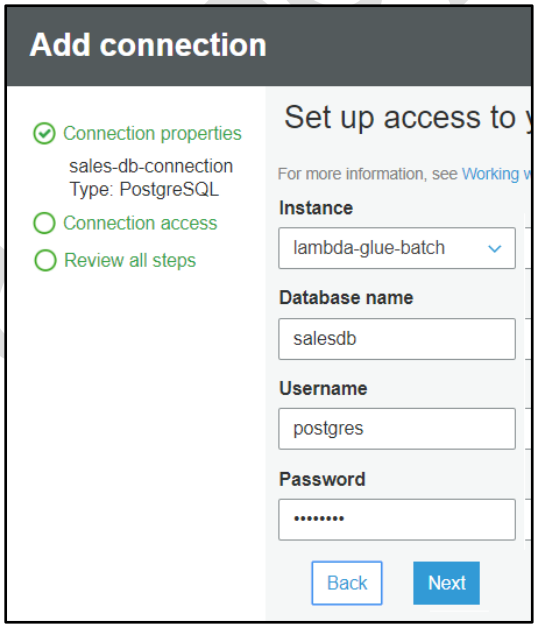
25) Go to connections on left tab and click Add connection.

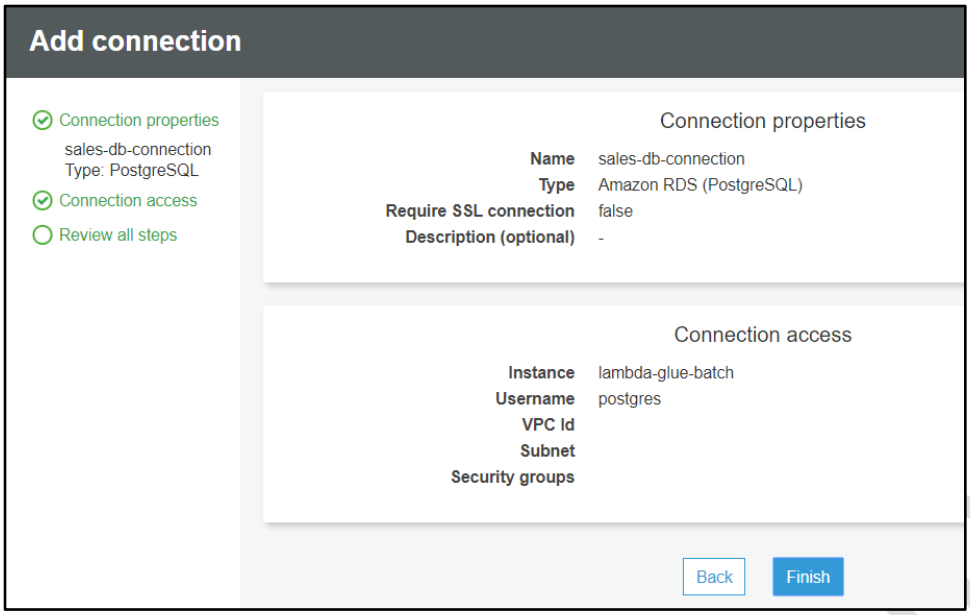


26) Setup connection details

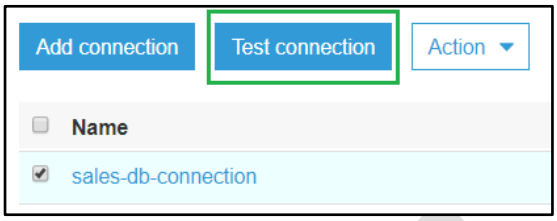


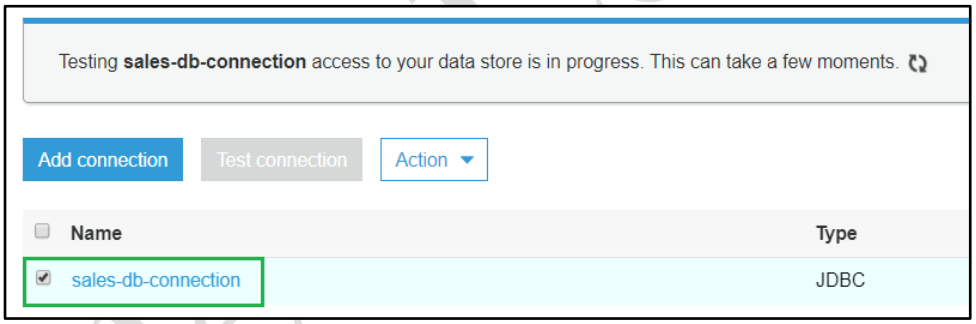
Provide instance, db name, user name and password.

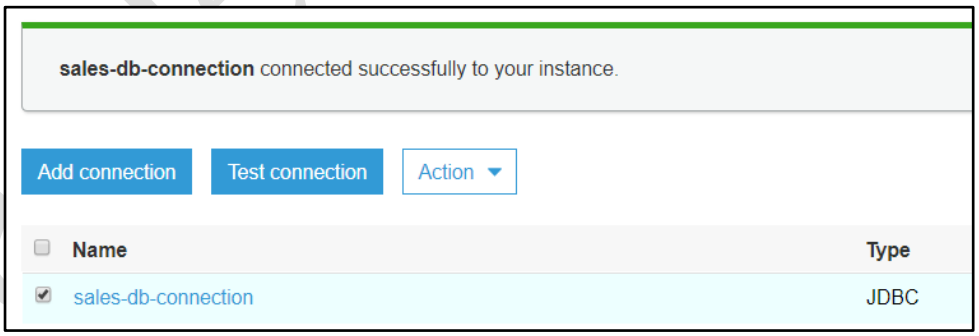




27) Now test by selecting created connection and click Test connection

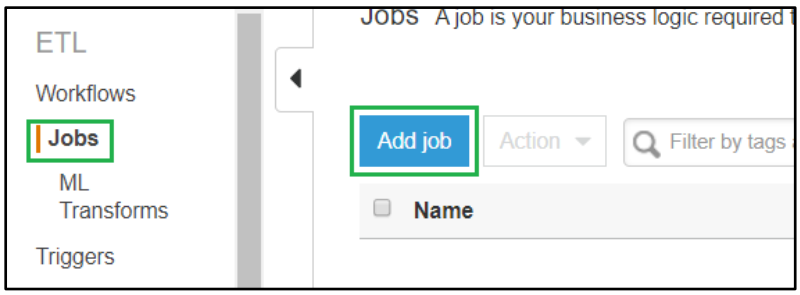




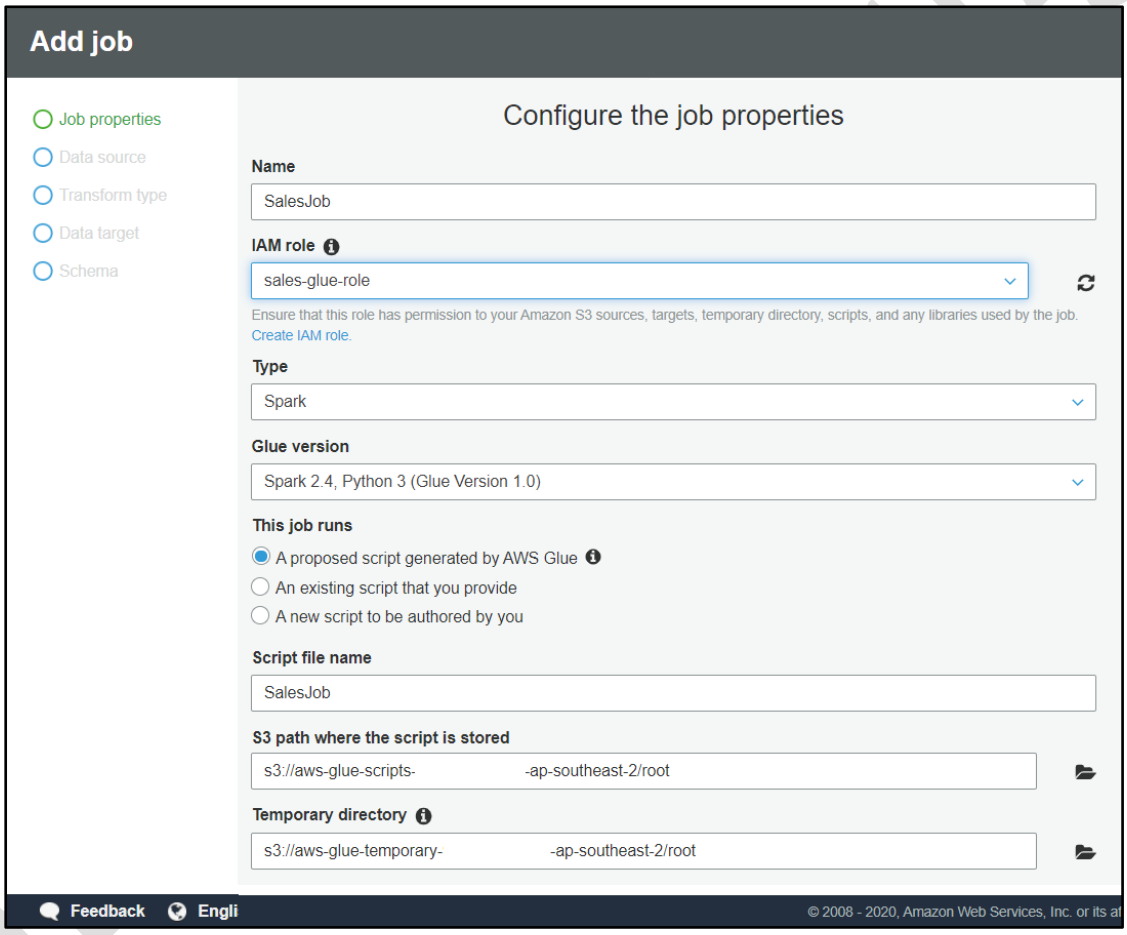


11.5. Create Job

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

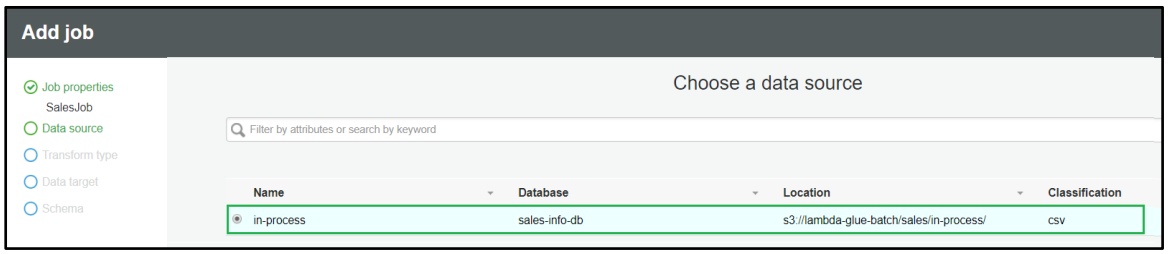


29) Provide Name and IAM Role, leave the rest to default values.

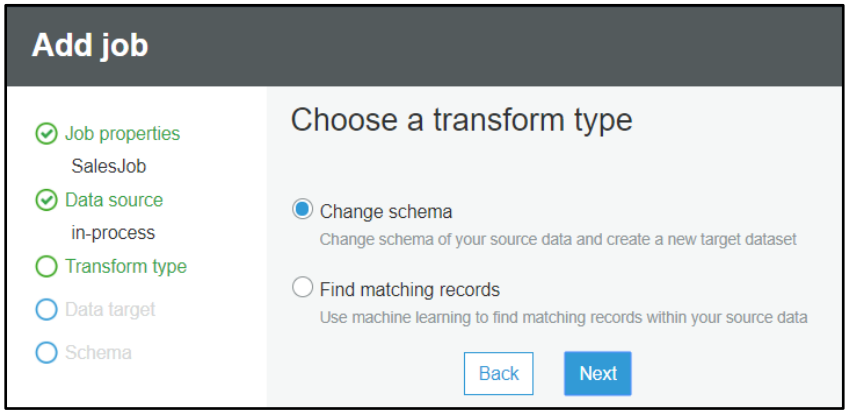


From the above screen shot it is clear that, Glue will generate Spark based code and stores it on s3 bucket as mentioned in above screen shot.

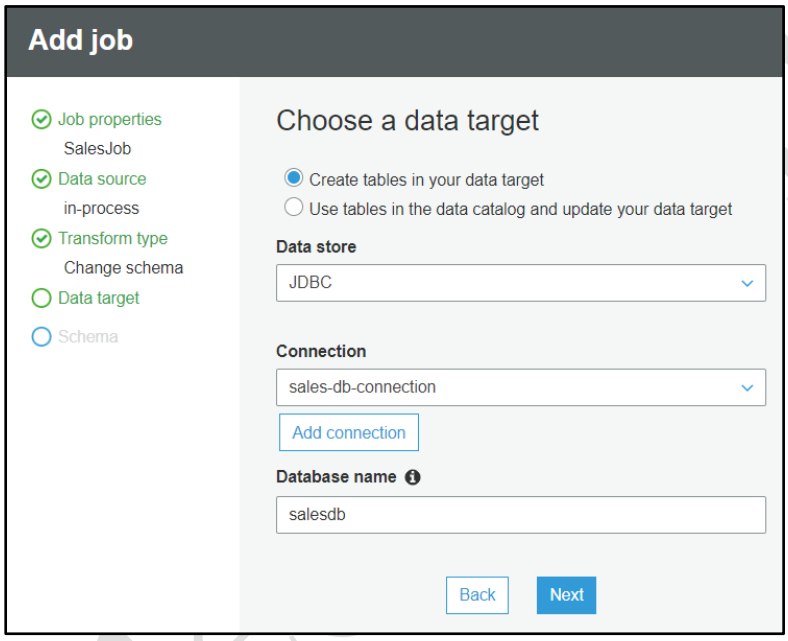
30) Choose Data Source and click next



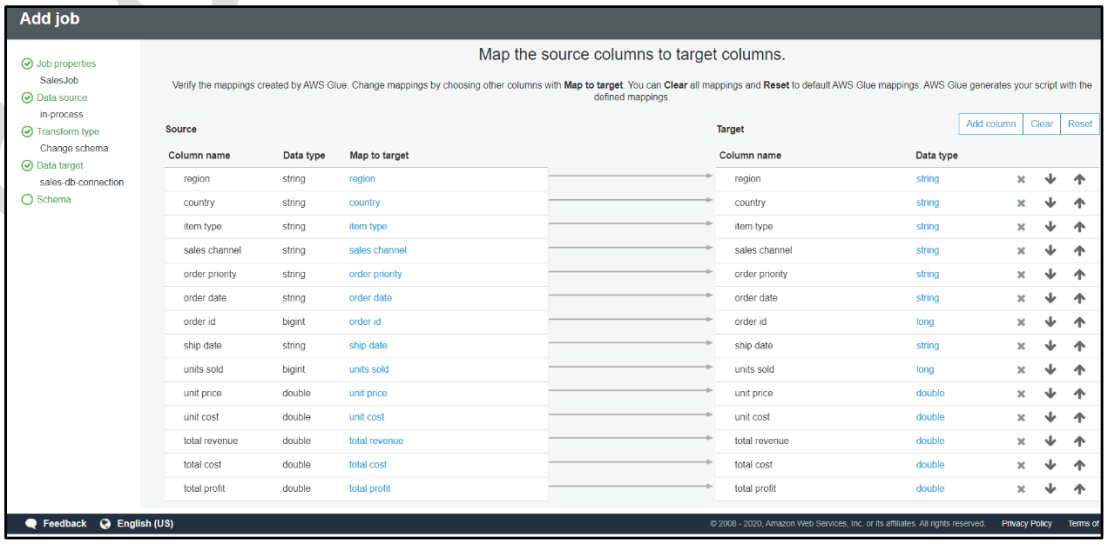
31) Leave Transform Type to default values



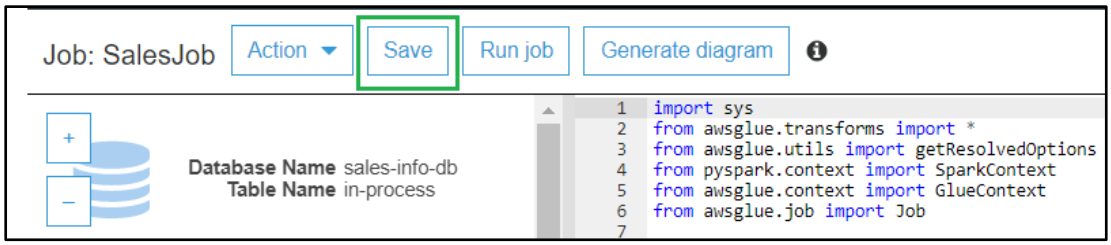
32) Select the target database



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



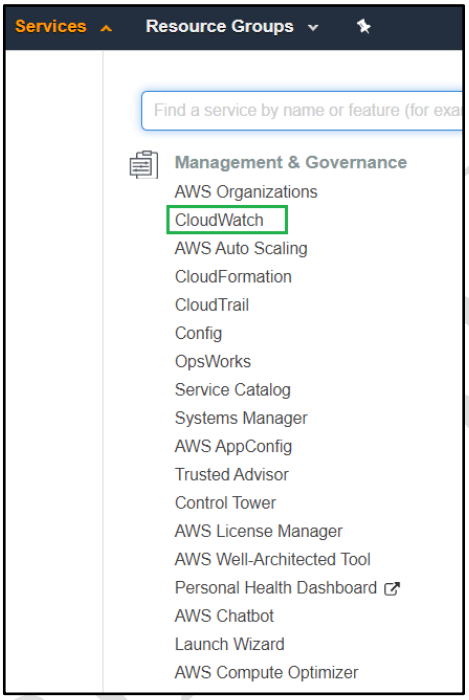
34) Save the script



35) Job added

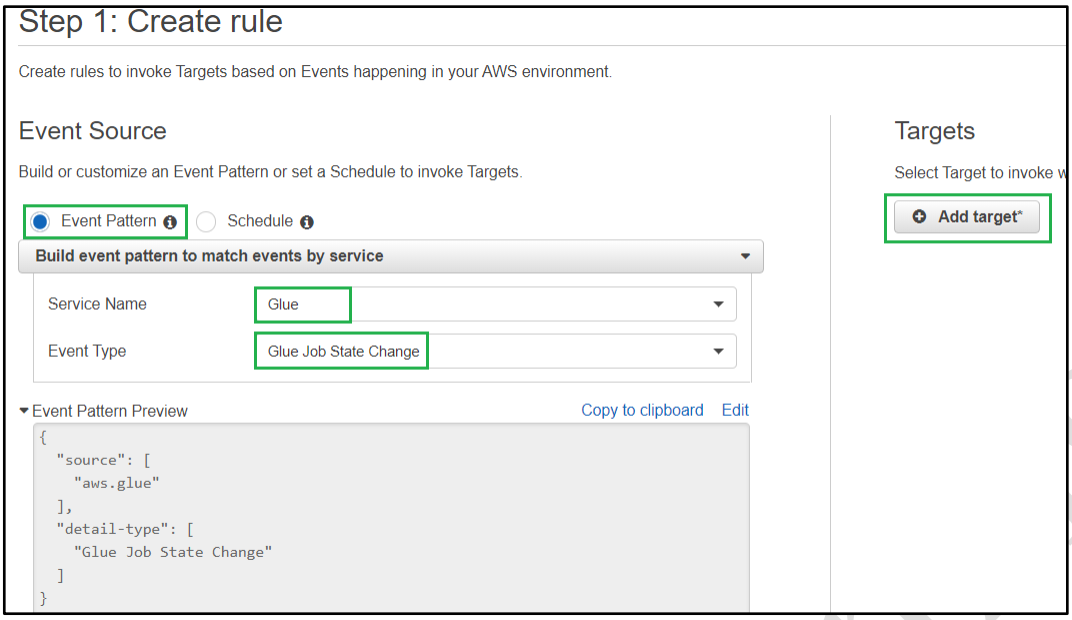
12. Setup Cloud Watch Event Rule

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

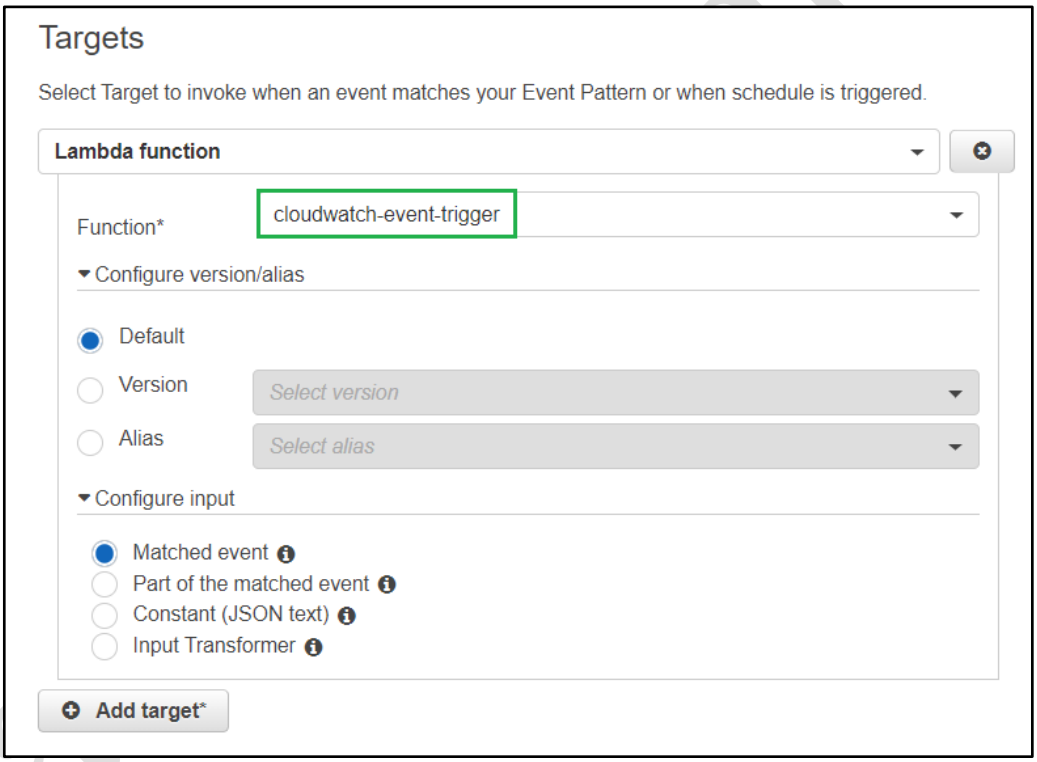


2) Click on Events → Rules → Create Rule

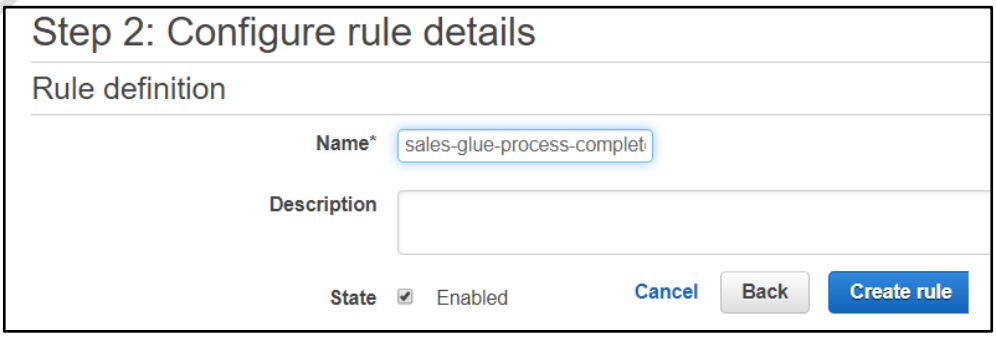
3) Select Event Pattern, Service Name and Event Type and then click Add Target



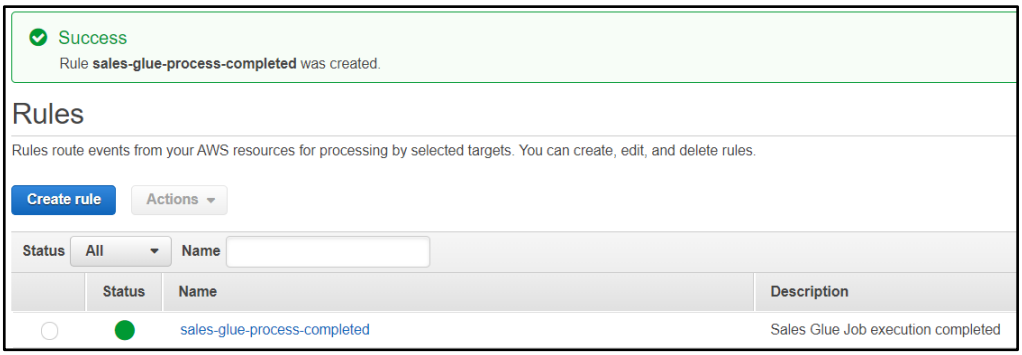
4) Configure Target



5) Provide name – "**sales-glue-process-completed**" and click create rule.



6) Rule created



13. Build Lambdas

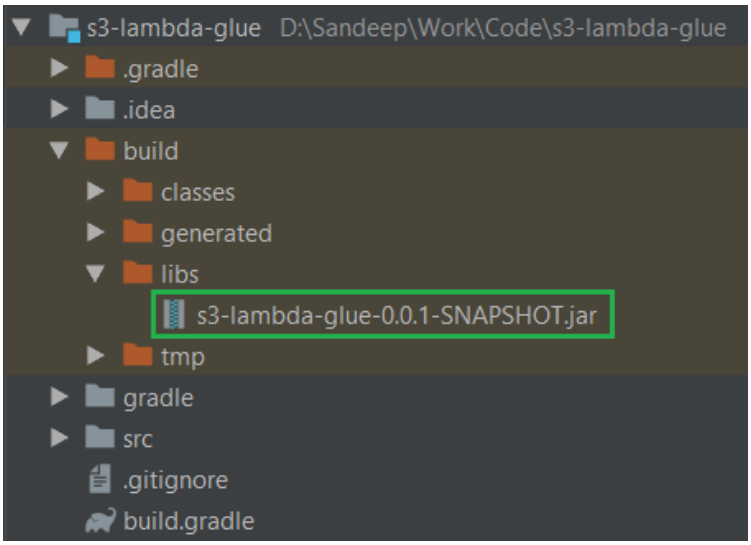
1) Build below 3 code repositories.
    a) s3-lambda-glue
    b) cloudwatch-lambda-s3
    c) lambda-lambda
2) Clean & Build the code once from the terminal

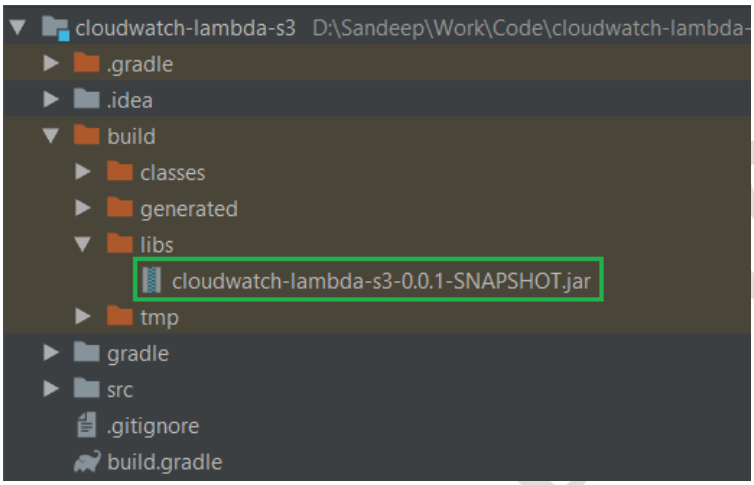
```
D:\Sandeep\Work\Code\s3-lambda-glue>gradle clean build

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

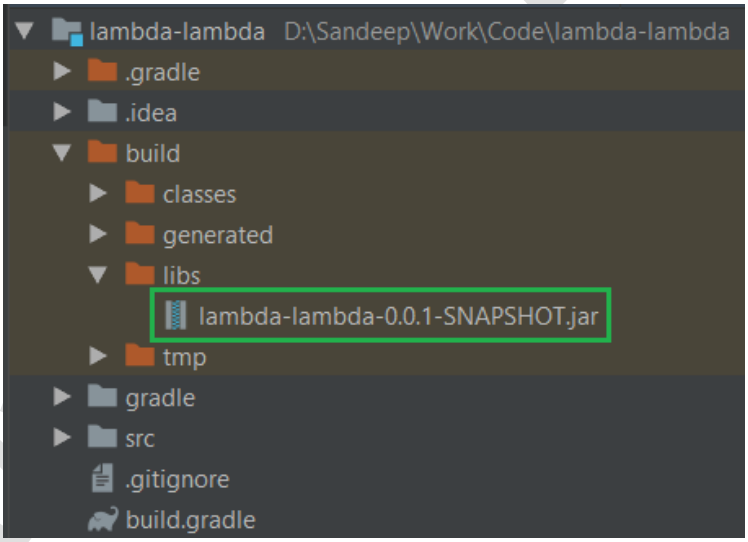
```
D:\Sandeep\Work\Code\cloudwatch-lambda-s3>gradle clean build

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

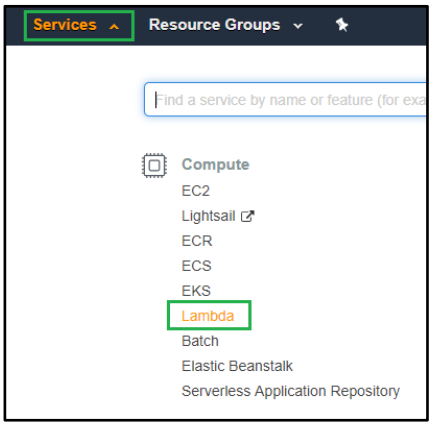




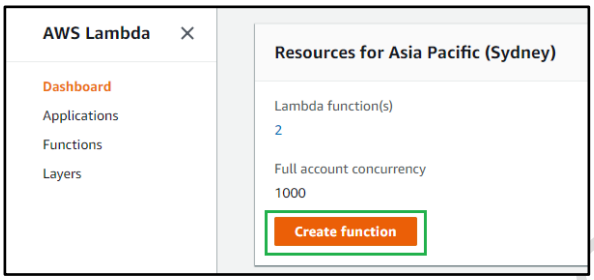


14. Deploying Lambda

1) Go to Services → Compute → Lambda

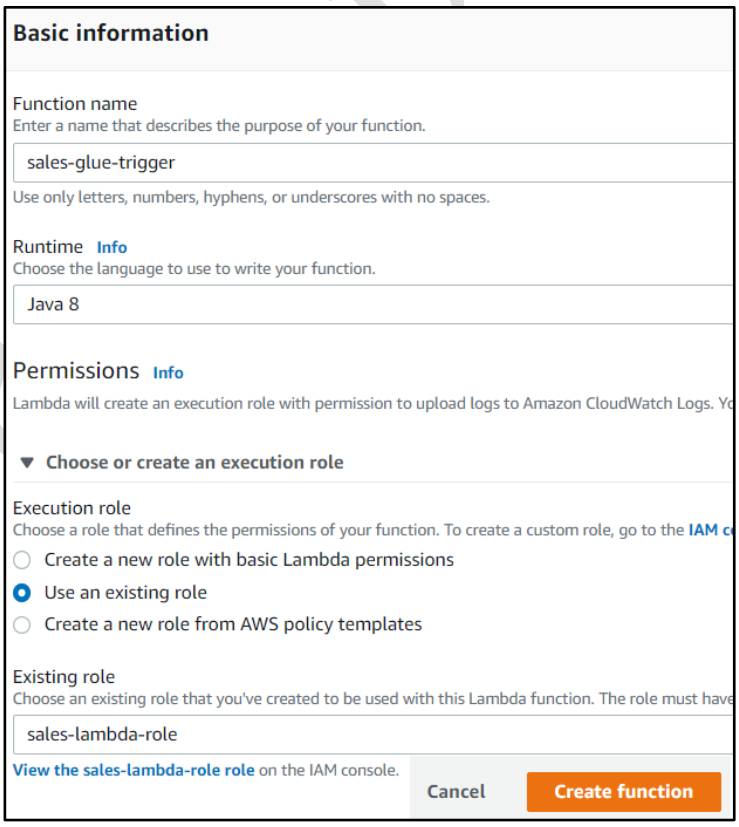


2) Click create function.

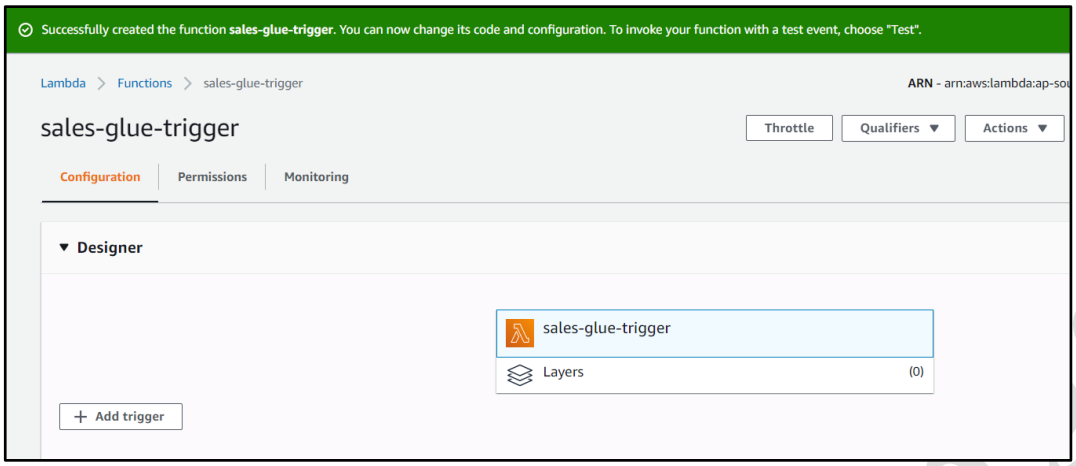


14.1 sales-glue-trigger

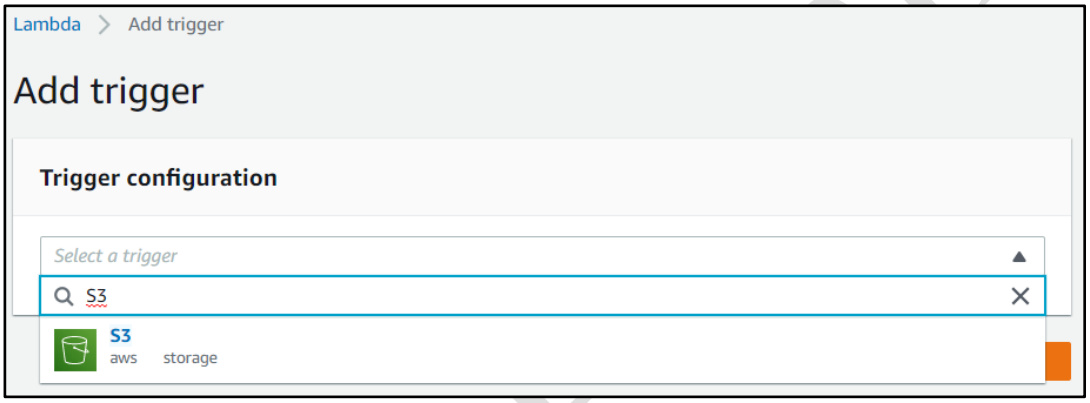
3) Create new function



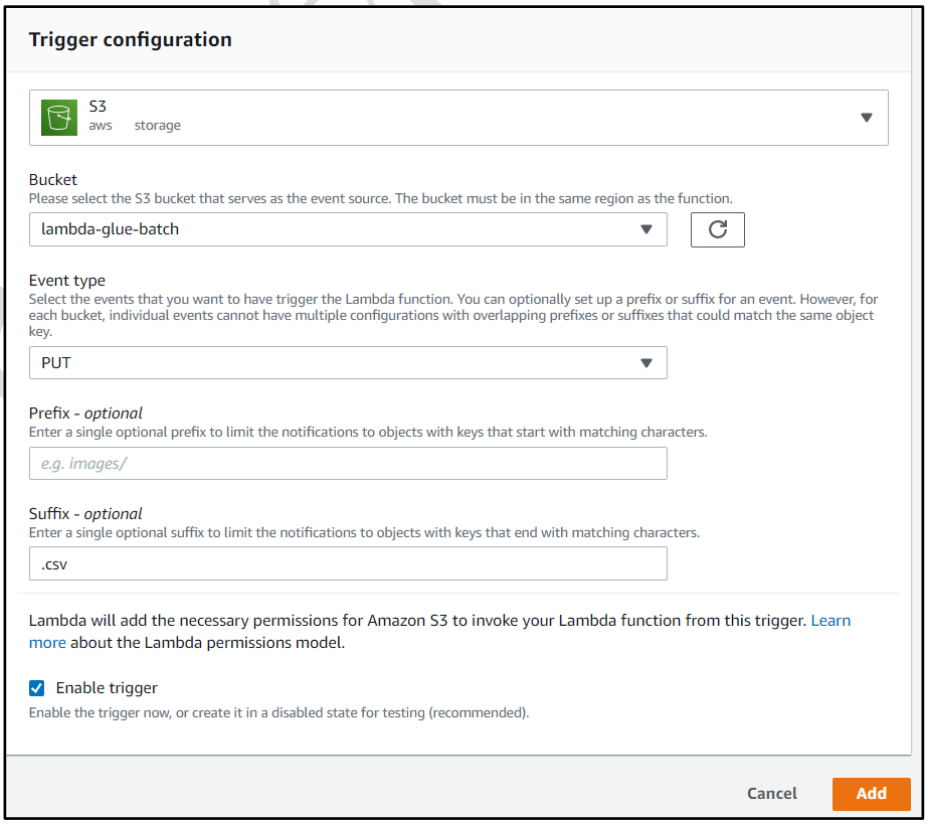
4) Lambda is created as shown below.



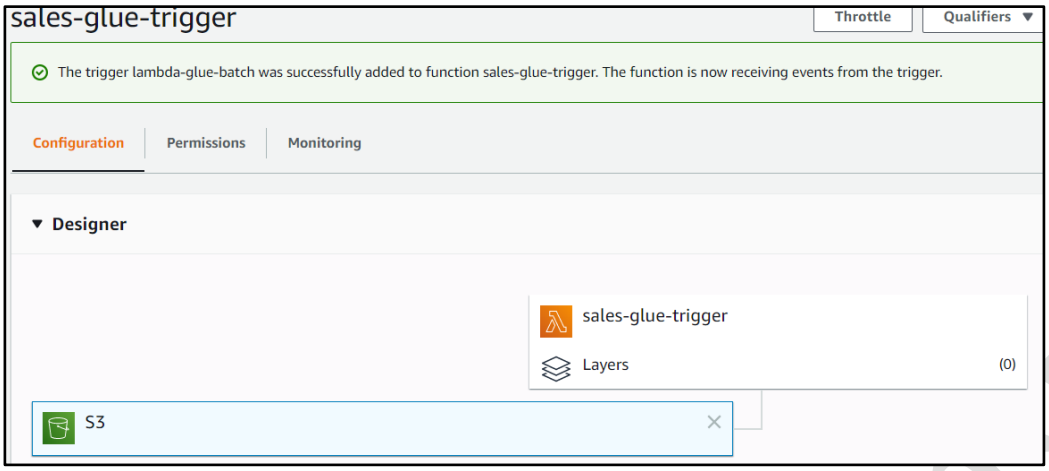
5) Click on add the trigger & Select S3



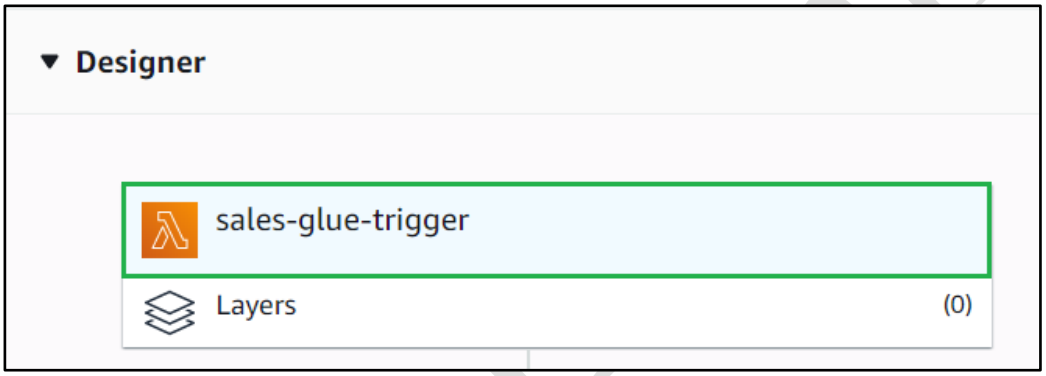
6) Configure trigger details, Add event type as **PUT** (when file is PUT in bucket, it triggers Lambda) for the files suffixed with **.csv** extension. then click Add.



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



8) Click on Lambda in the Designer pane.

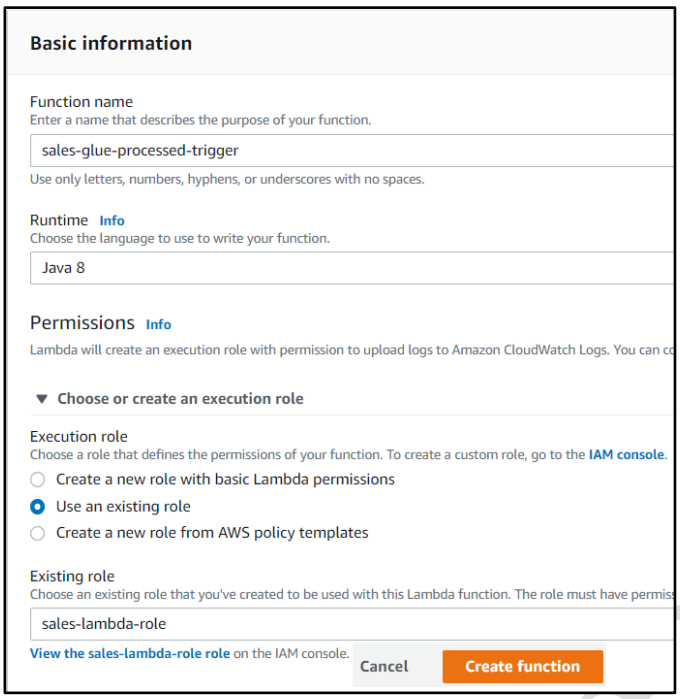


9) In the Function code Pane –
upload jar - s3-lambda-glue-0.0.1-SNAPSHOT.jar
provide the Handler info –
com.sandeep.learning.s3lambdaglue.handler.NotificationHandler::handleRequest
click save.

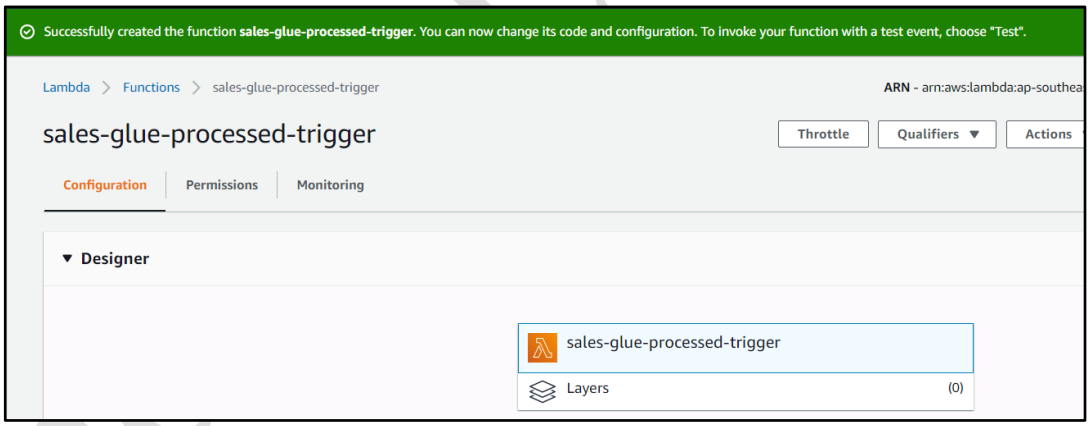
10) Once uploaded and saved, Lambda is ready for testing.

14.2 sales-glue-processed-trigger

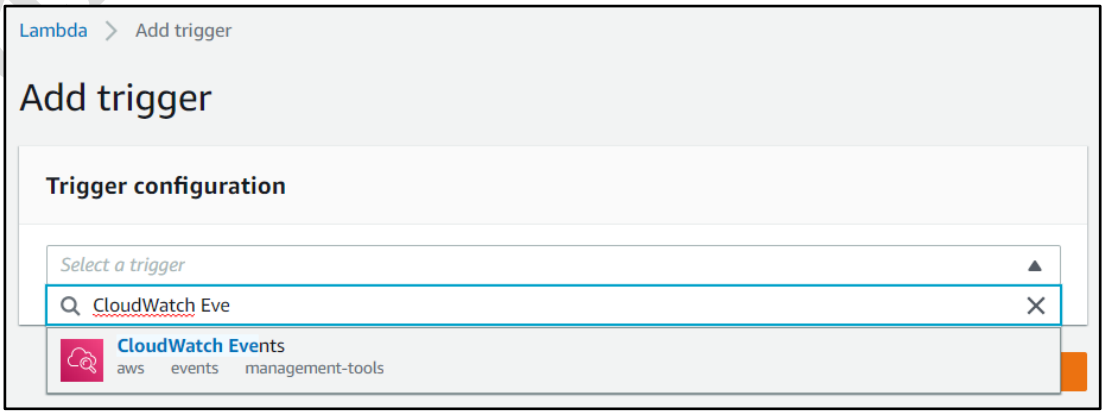
11) Create new function



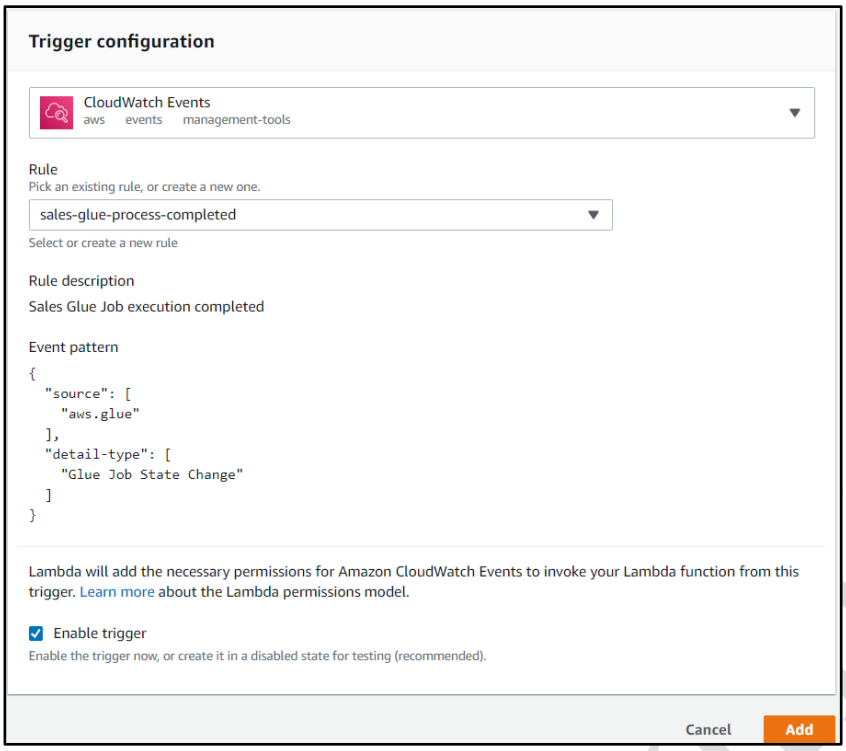
12) Lambda is created as shown below.



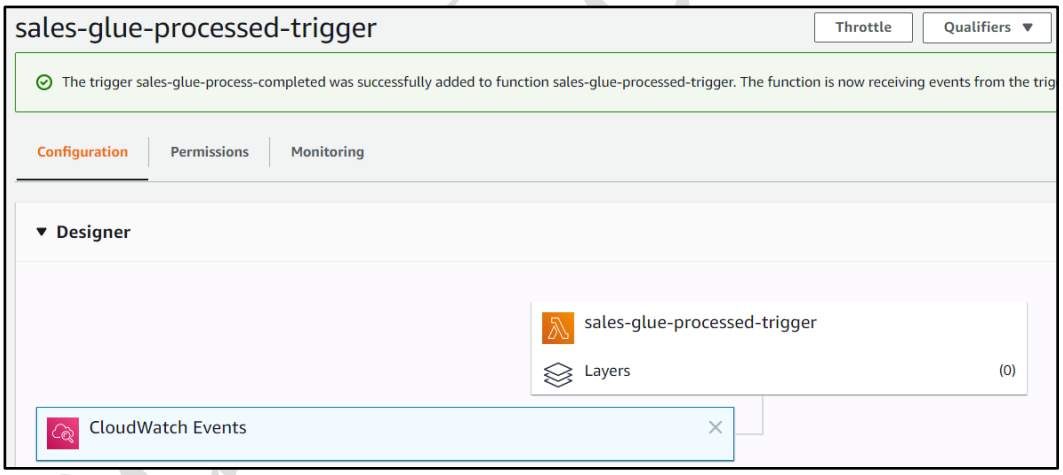
13) Click on add the trigger & Select Cloud Watch Events



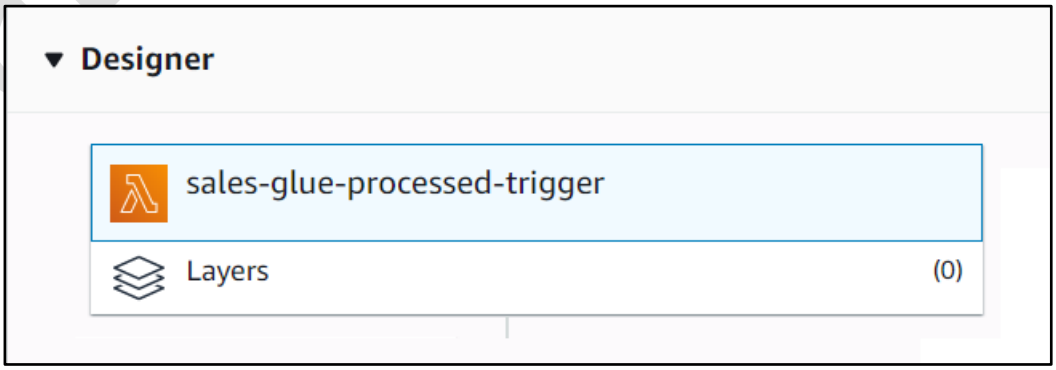
14) Configure trigger details, by selecting the rule sales-glue-process-completed created earlier.



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 –

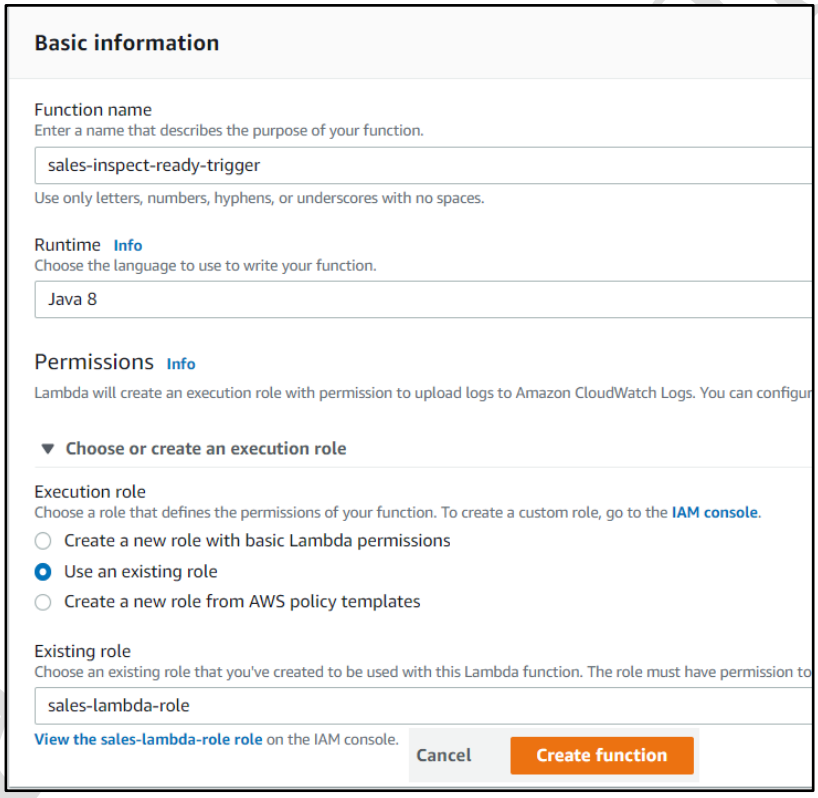
com.sandeep.learning.cloudwatchlambdas3.handler.CloudWatchEventHandler::handleRequest

click save.

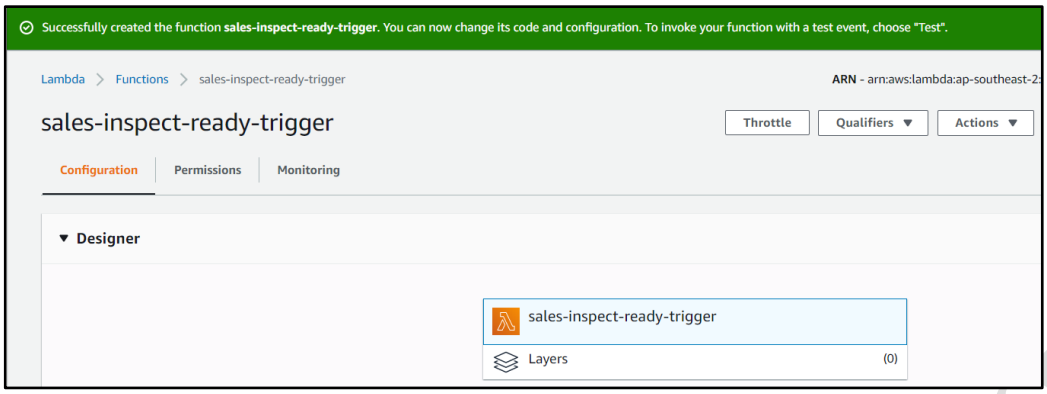
18) Once uploaded and saved, Lambda is ready for testing.

14.3 sales-inspect-ready-trigger

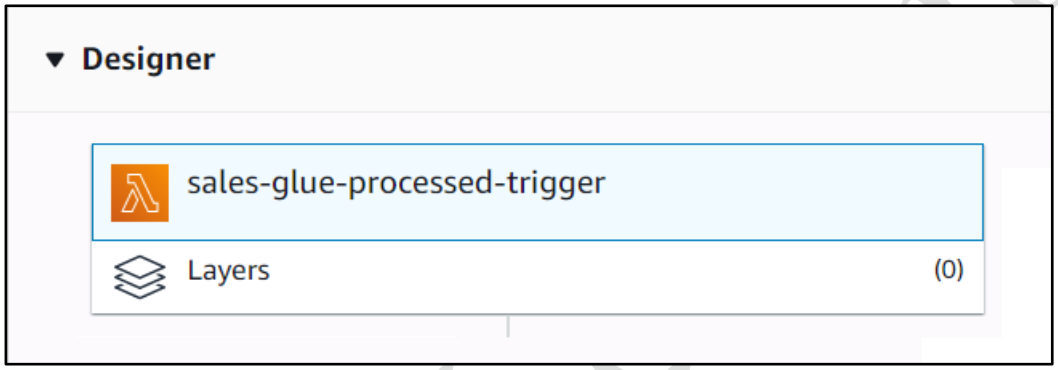
19) Create new function



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.

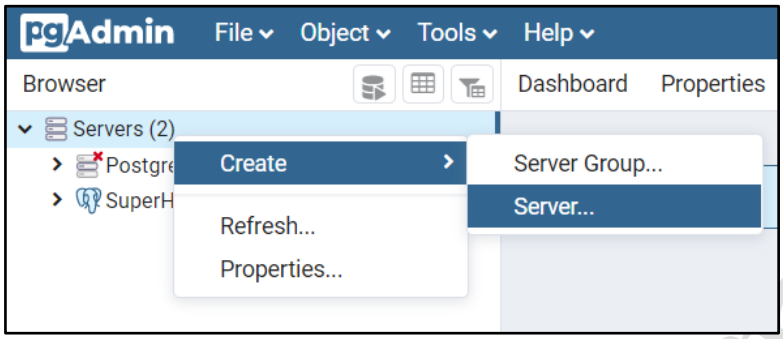
23) Once uploaded and saved, Lambda is ready for testing.

15. Testing & Verification

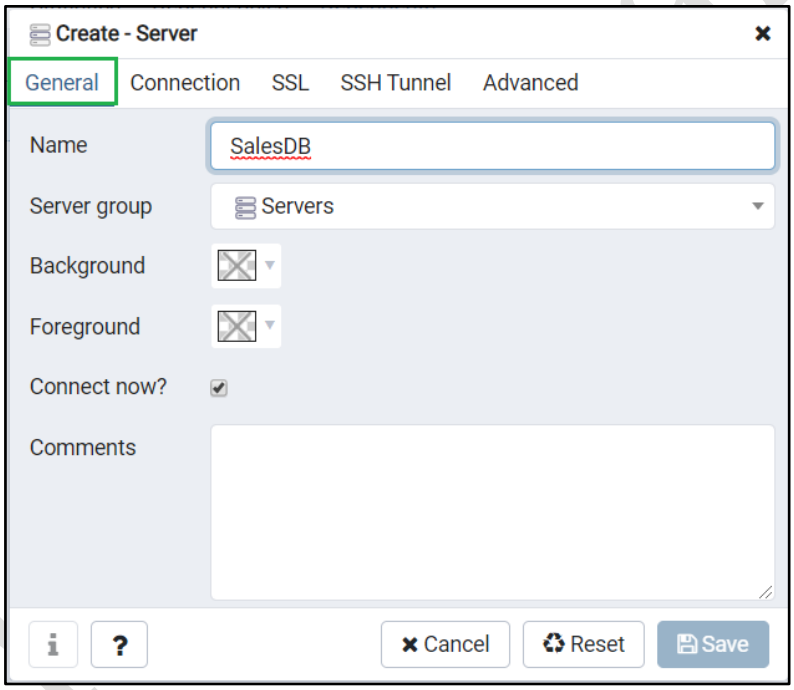
15.1 Setting up pgAdmin Tool

Note: pgAdmin tool can be installed from here.

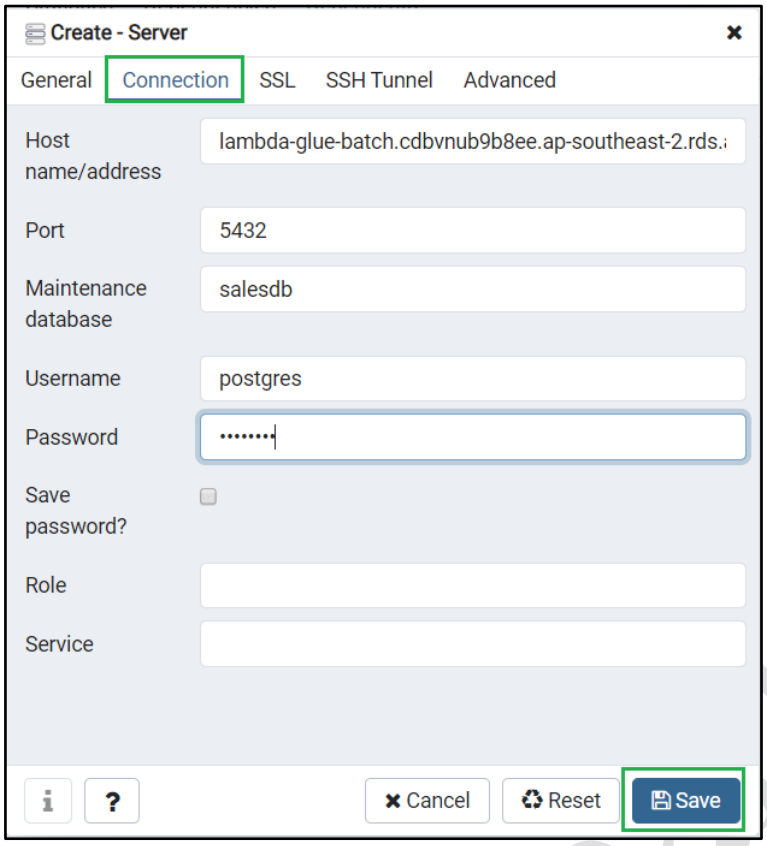
1) Connection to database from pgAdmin tool
2) Right click on Servers → Create → Server



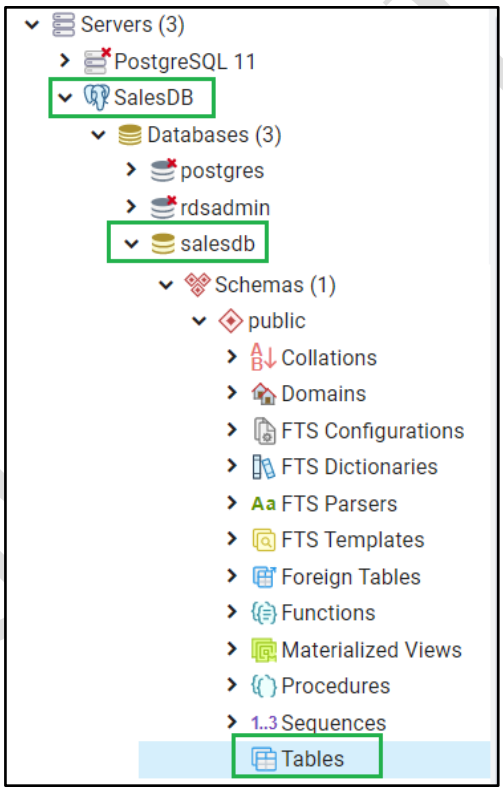
3) Provide below details



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



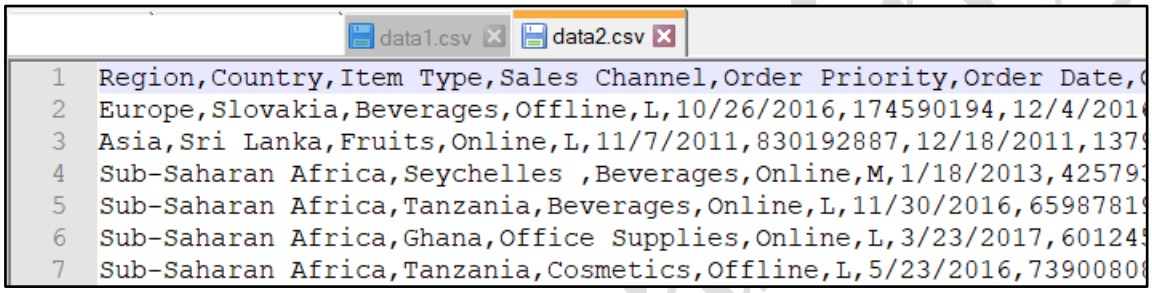
From the above screen shot you can see that there are no tables available here.

15.2 Testing Scenario

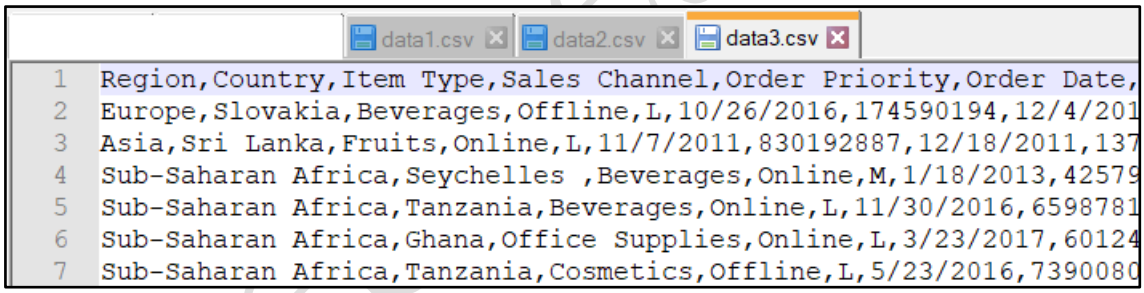
6) Now upload 2 files (data1.csv, data2.csv) to S3 Bucket ready folder in one shot and once the files are moved to in-process folder upload one more file (data3.csv).

data1.csv – has 5 records as shown below.

data2.csv – has 7 records as shown below.



data3.csv – has 7 records as shown below.



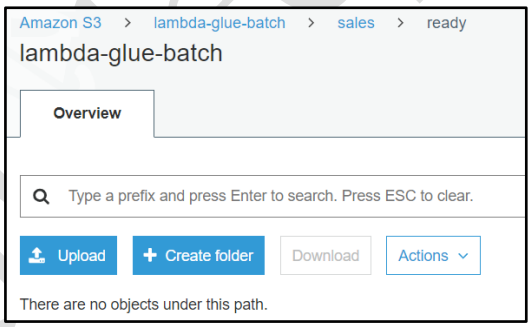
15.3 Expected Behaviour vs Actual Behaviour

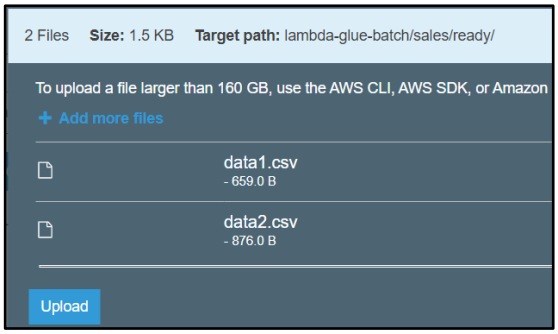
S. No	Expected Behaviour	Actual Behaviour
1	**sales-glue-trigger** lambda is invoked upon files upload. Moves data1.csv and data2.csv to in-process folder, triggers glue job and updates the dynamo db jobs table with job id, files and job status.	Aligns with expected behaviour. Refer point 9 and point 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, 2nd 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 1st 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 1st 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 2nd 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 2nd 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.

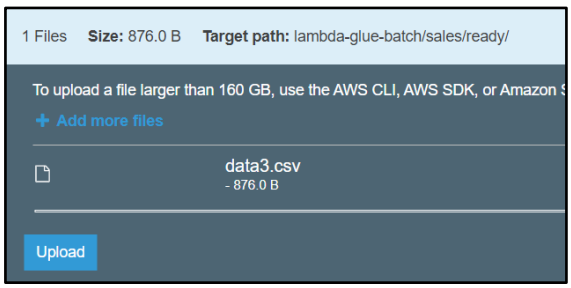
15.4 Testing In-Progress

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

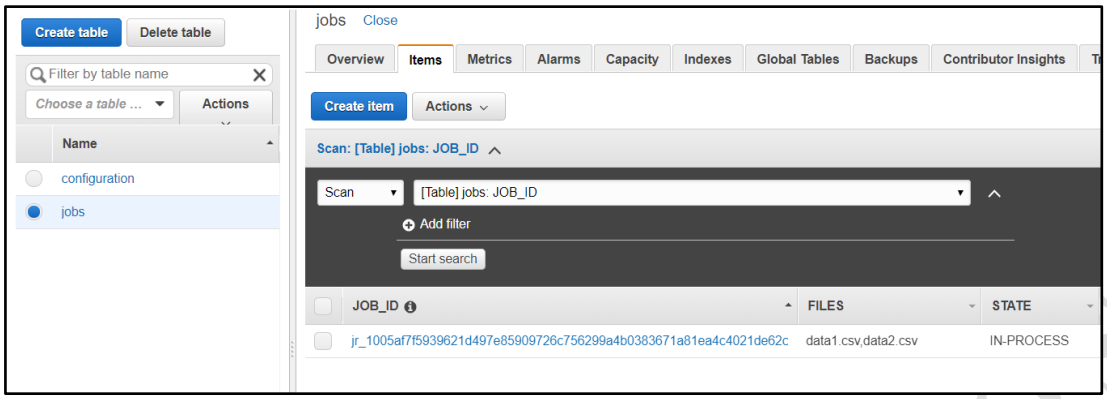




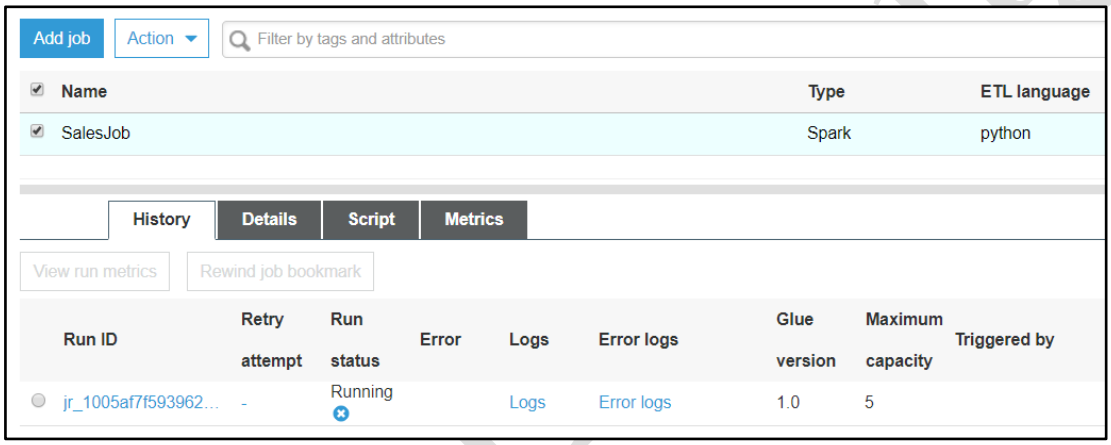
8) Files moved immediately in-process directory so uploading data3.csv



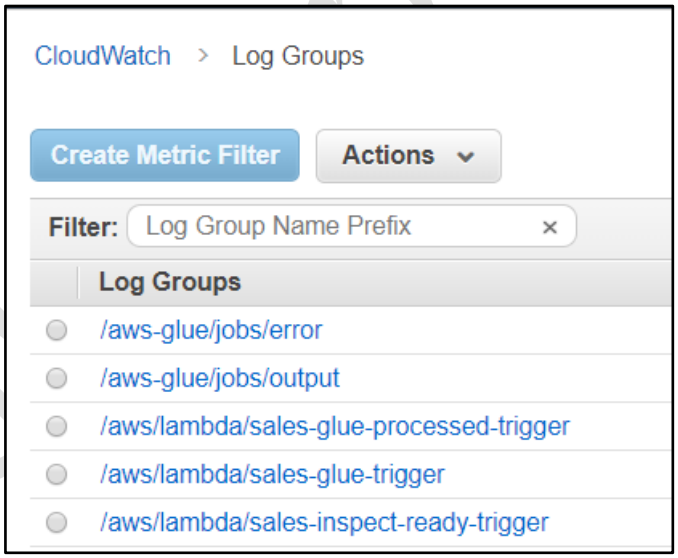
9) Dynamo DB Jobs table status



10) Glue Job state

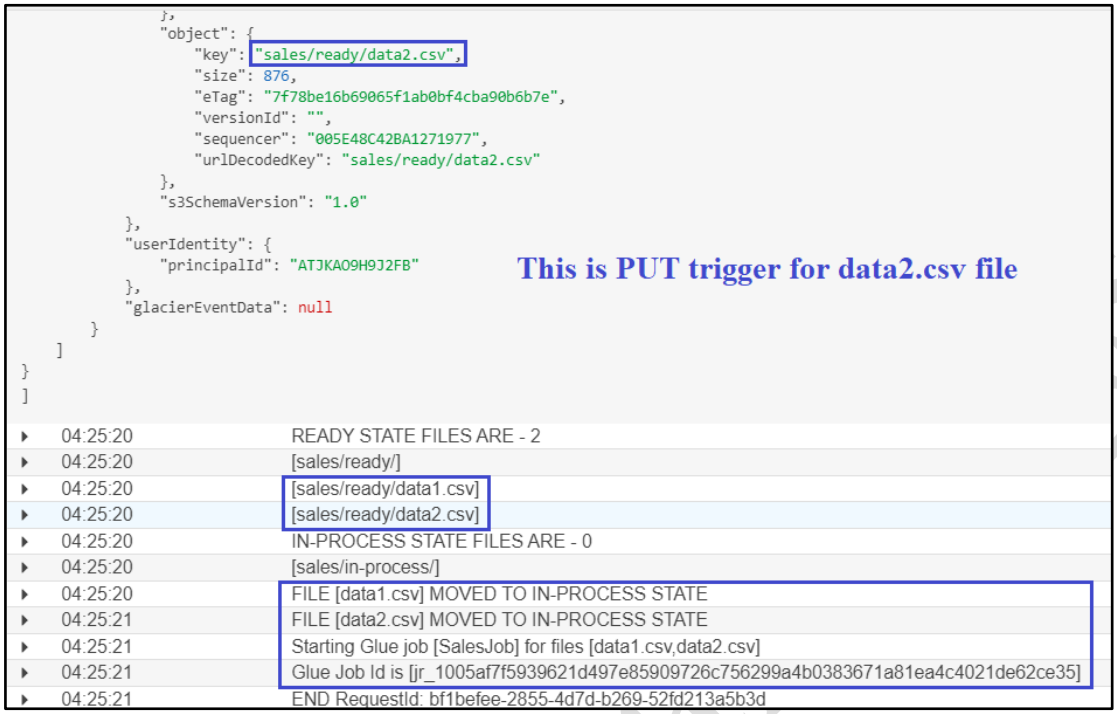


11) Go to CloudWatch → Log Groups to check logs for corresponding components.

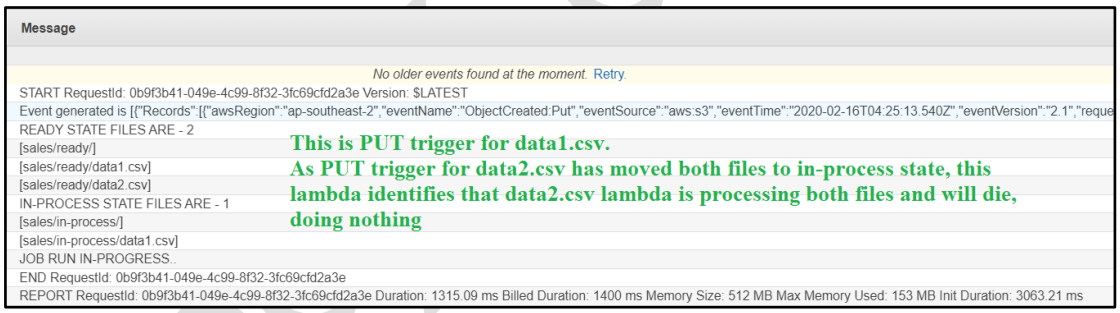


12) Lambda Logs for **sales-glue-trigger**

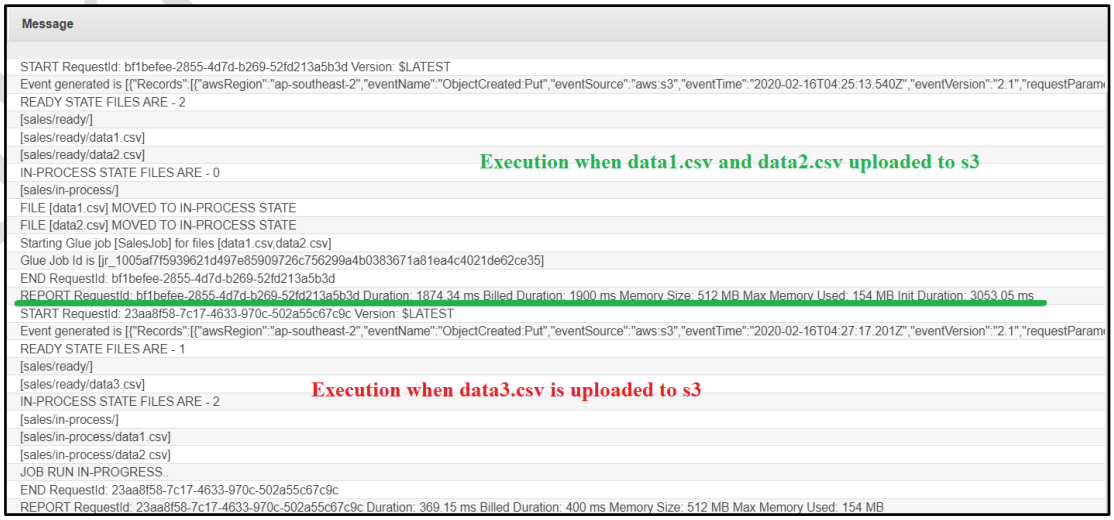
Below log is for data2.csv file which actually processed both files.



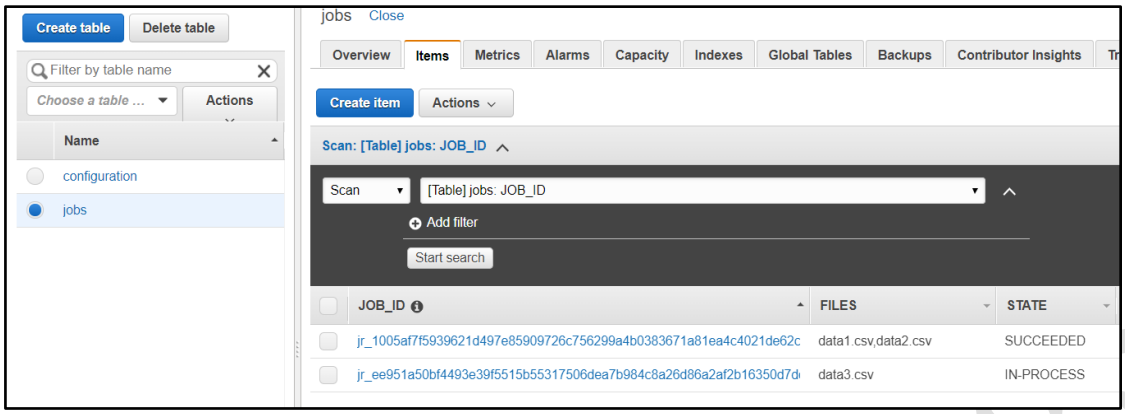
Below is the log for data1.csv file, which identified that other lambda is processing both files.



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

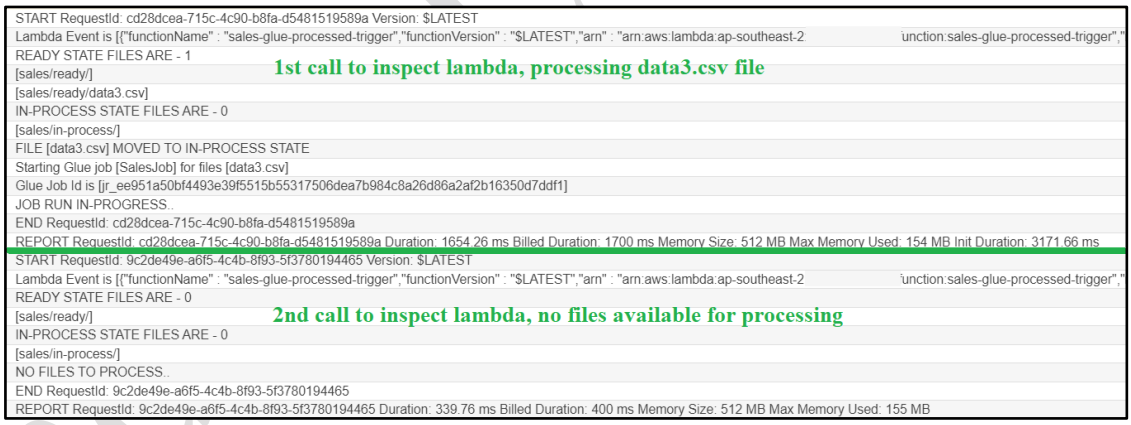


14) Dynamo DB Job state changed to SUCCEEDED & data3.csv is picked up for processing.

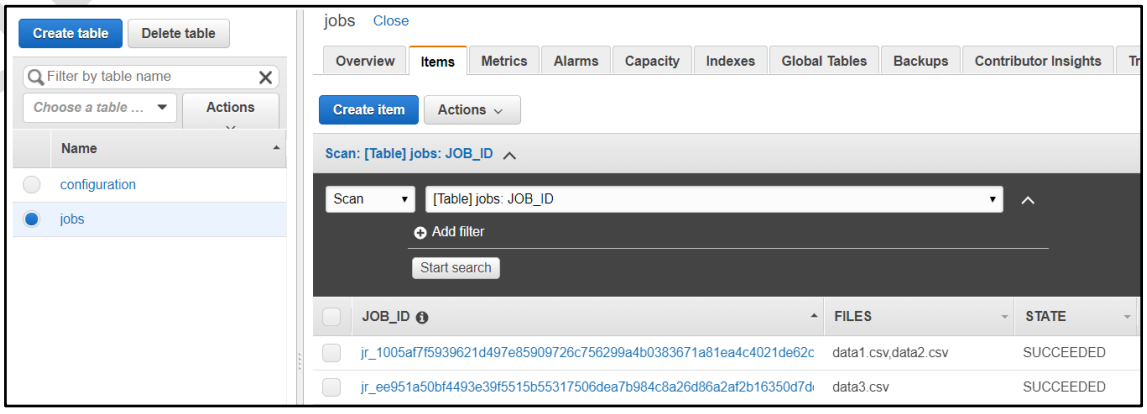


15) Let's check logs for sales-glue-processed-trigger lambda

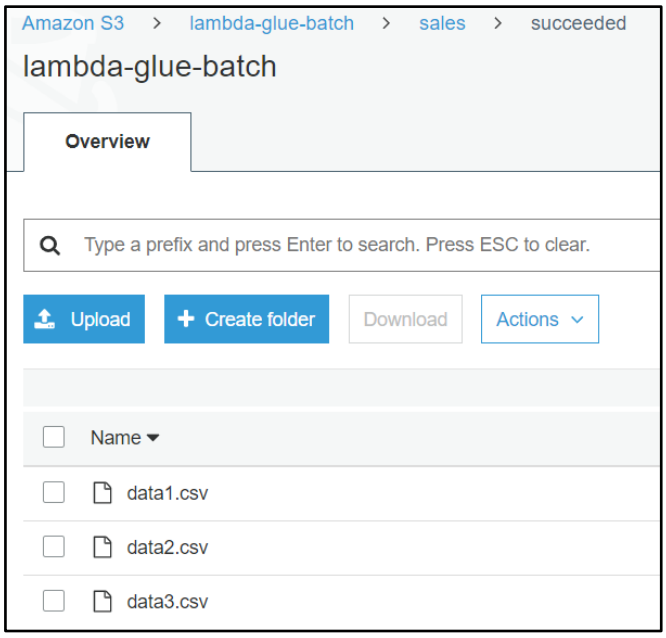
16) Let's check logs for sales-inspect-ready-trigger lambda



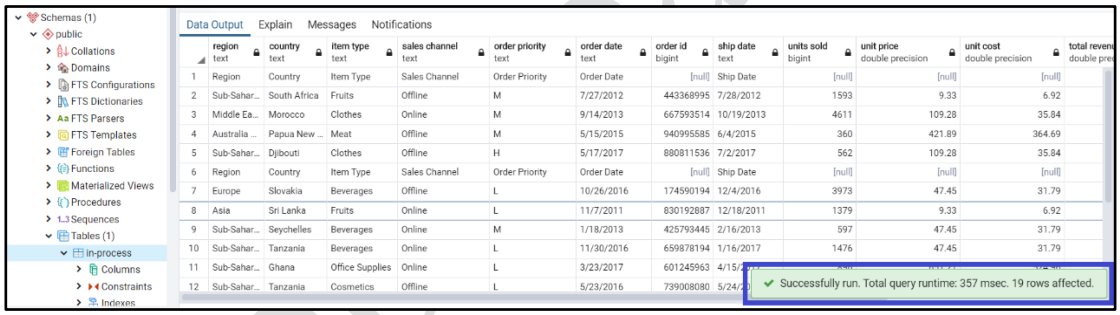
17) Dynamo DB Job state changed to SUCCEEDED for data3.csv file.



18) 3 Files found in SUCCEEDED directory.



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



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