S3 to Postgres Documentation

Table of Contents

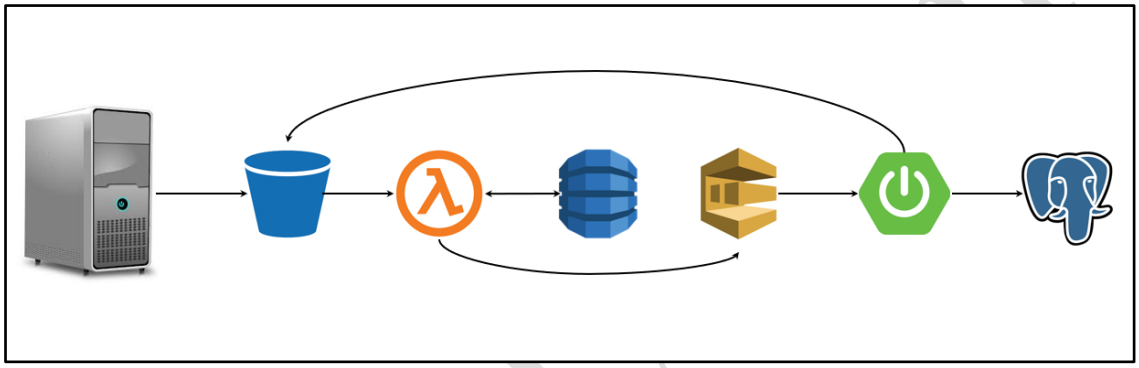
1. Use case

To migrate the data from on prem servers to AWS.

2. Project Description

S3 to Postgres is about copying the data from S3 bucket which is in CSV format to AWS RDS Postgres database. This solution uses streaming the data to database which makes this solution to perform efficiently even to load millions of records.

3. Architecture



4. Flow

1) Legacy system shall generate a CSV file which aligns with the Postgres DB table structure.
2) Generated CSV file will be placed in the S3 bucket by the legacy system.
3) Upon placing the file, S3 bucket triggers lambda notification.
4) Lambda reads the SQS queue configuration from Dynamo DB.
5) Lambda shall push the message to SQS.
6) A Spring boot service listens to SQS queue consumes the message.
7) Service shall stream the data from S3 bucket file to Postgres DB.

5. Steps

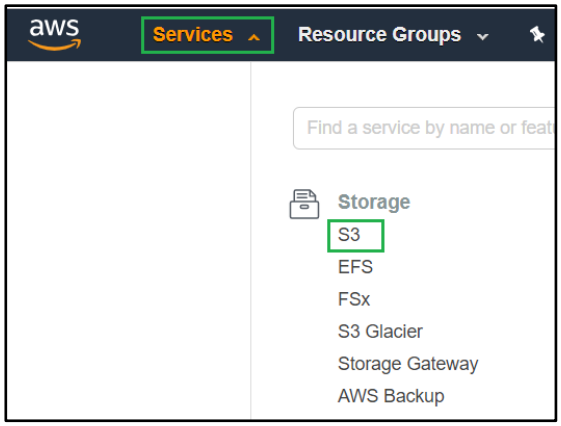
To run the s3-to-postgres application, follow below steps.

1) Create S3 bucket
2) Create SQS queue
3) Create EC2 instance
4) Create RDS Postgres database
5) Build the application
6) Lambda configuration in DynamoDB
7) Deploying Lambda
8) Run application

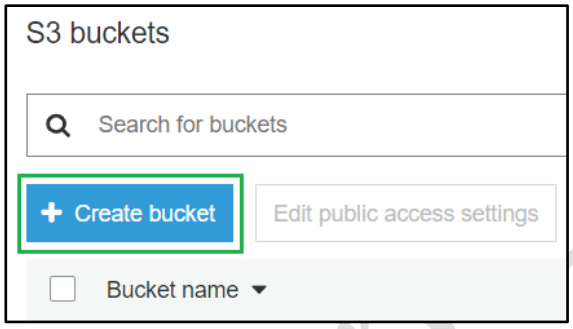
9) Testing

6. Create S3 bucket

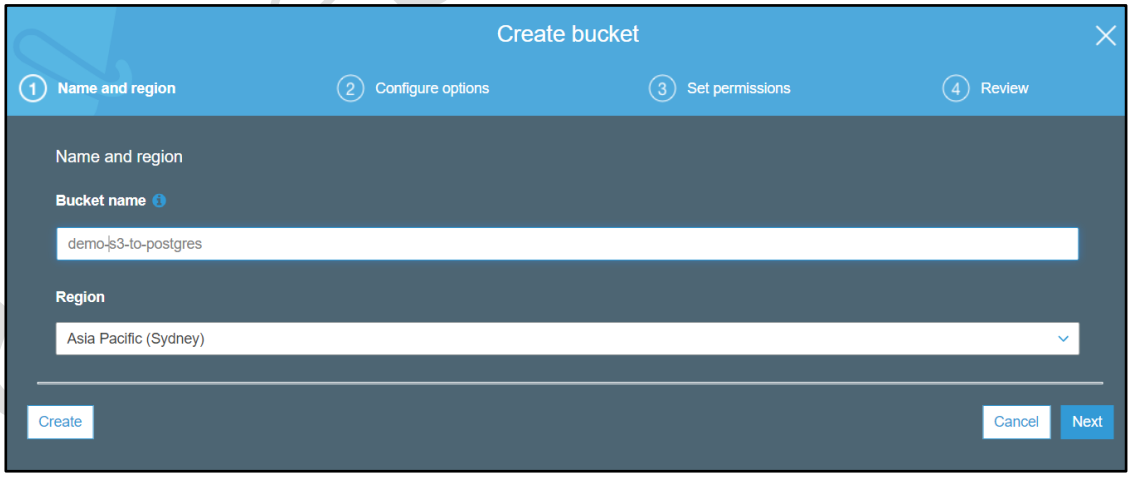
1) Click on Services → Storage → S3



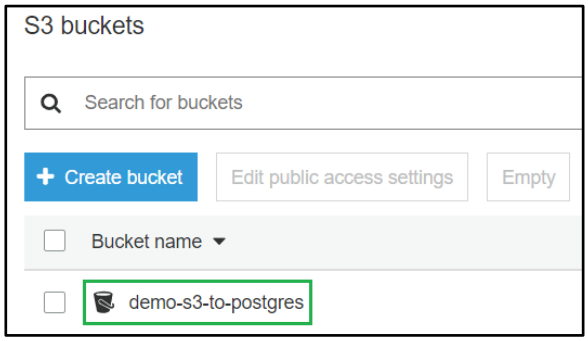
2) Click on create bucket



3) Provide a bucket name

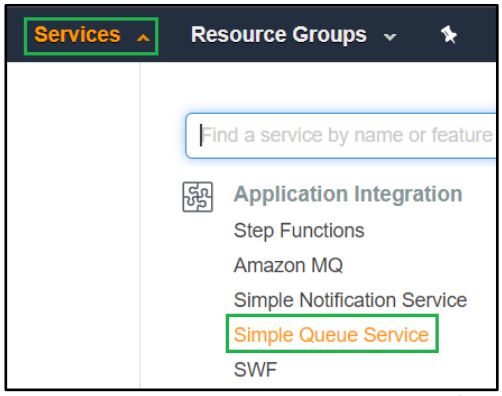


4) Bucket gets created as shown below.

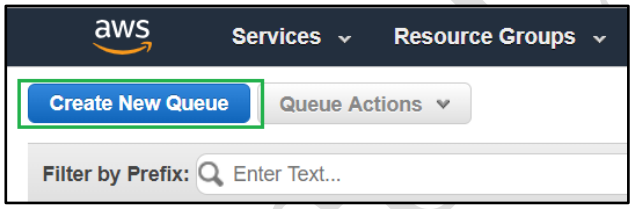


7. Create SQS queue

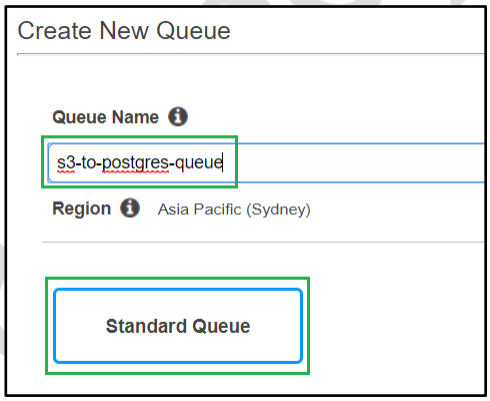
1) Click on Services → Application Integration → Simple Queue Service



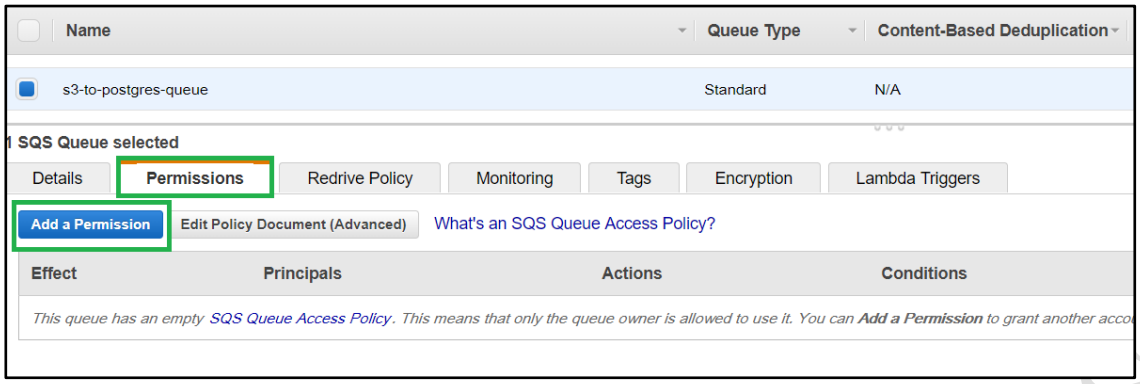
2) Click on create new queue



3) Create a Standard queue by providing name and click Quick-Create Queue.

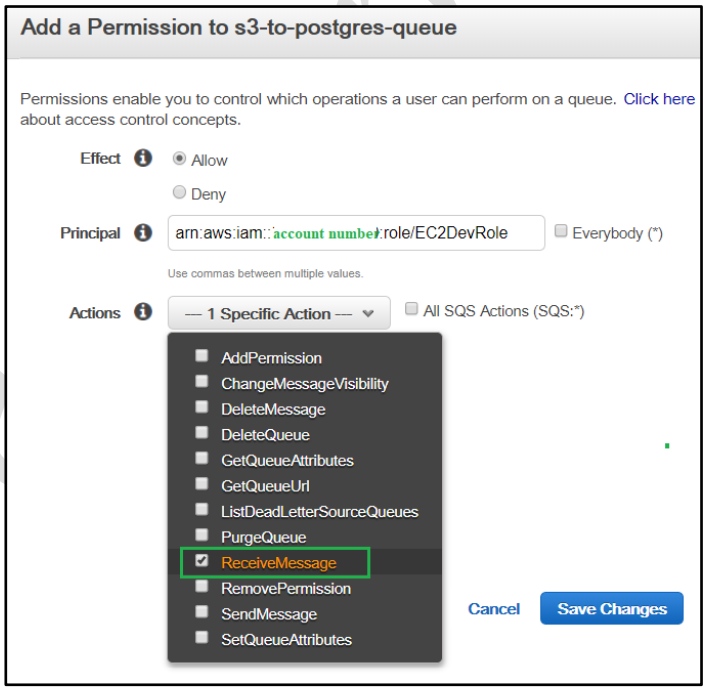


4) Provide permissions to the queue to post messages from Lambda and to consume messages from EC2 by navigating to the Permissions tab.

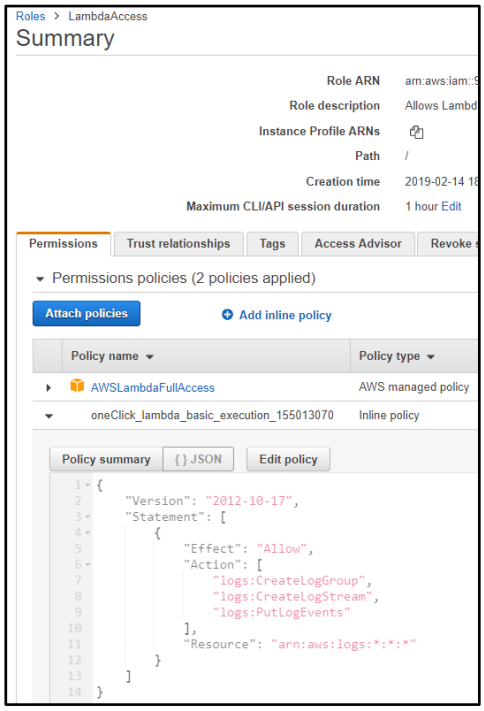


5) Select the IAM role which has access to SQS.

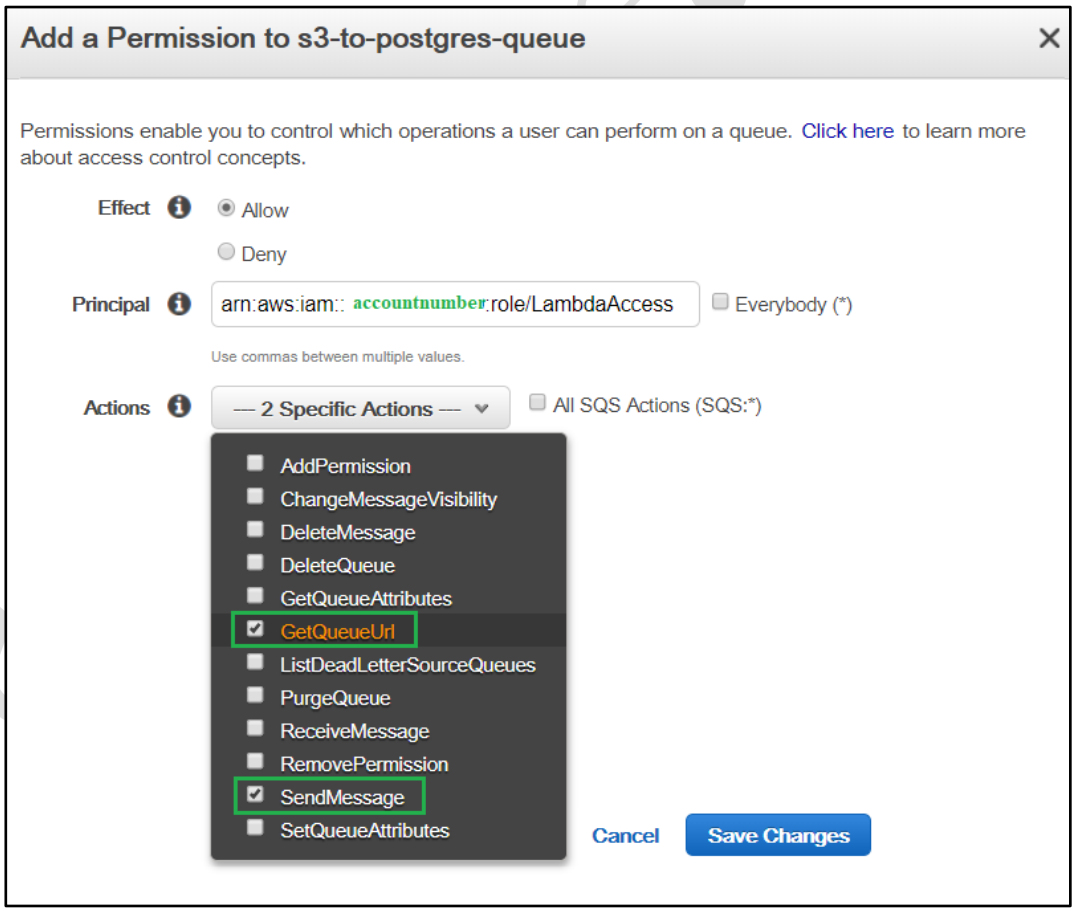
6) Allow EC2DevRole (Which will be associated with EC2 instance) to receive messages from the queue. Associate Receive action as shown below.



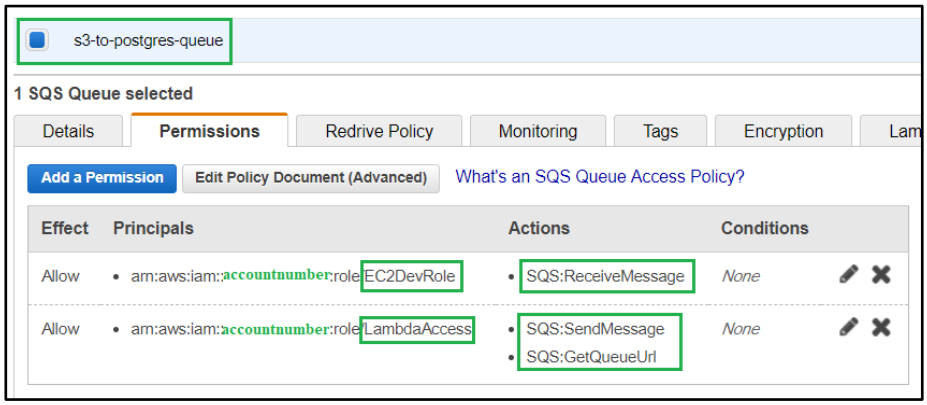
7) Make sure that LambdaAccessRole is attached with below policies.



8) Allow LambdaAccessRole (Which will be associated with Lambda function) to Get Queue URL and Send Messages to the queue. Associate actions as shown below.

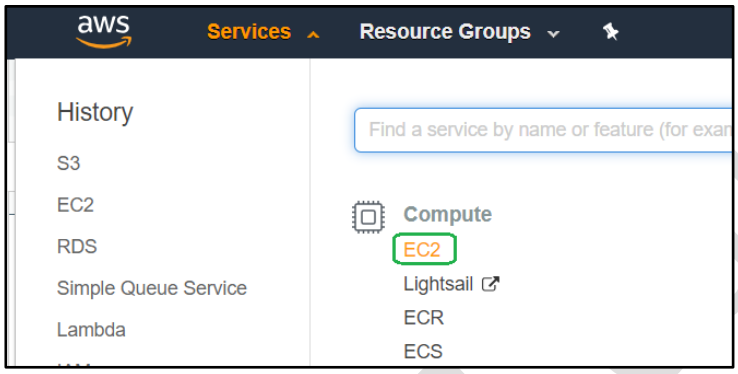


9) Overall permissions on the SQS queue should look like something below.



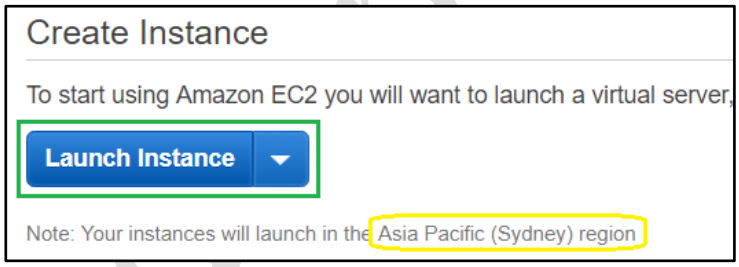
8. Create EC2 Instance

1) Login to your AWS account and click on the EC2 services.

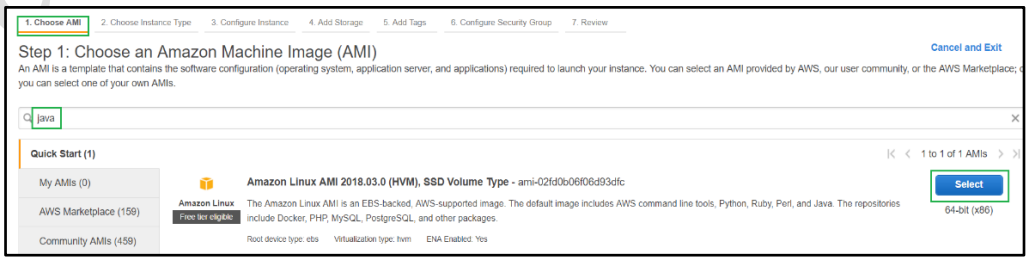


Make sure that you select region where the service is to be deployed.

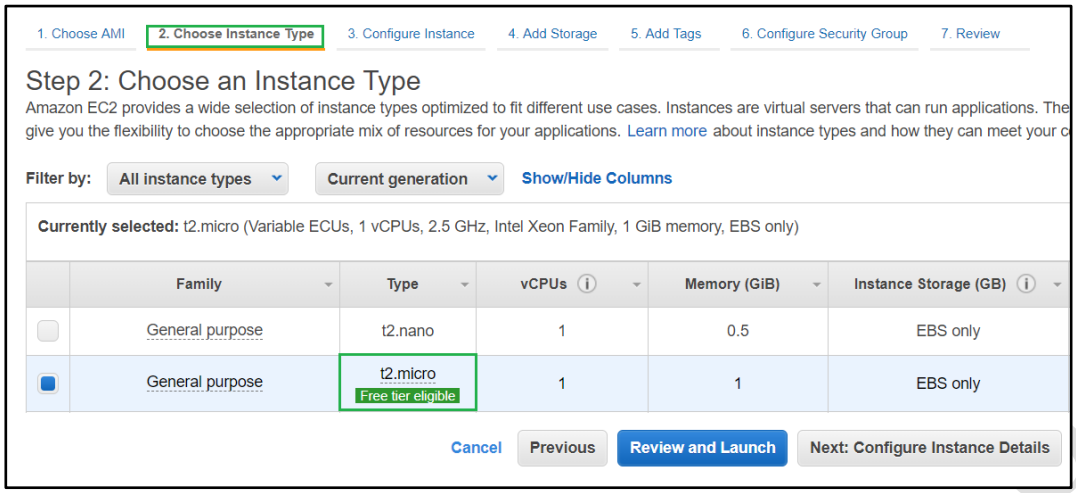
2) Click on Launch instance



3) Launch Linux AMI which has Java installed in it. As it is demo let us select free tier eligible AMI. Search for Java to search an AMI.

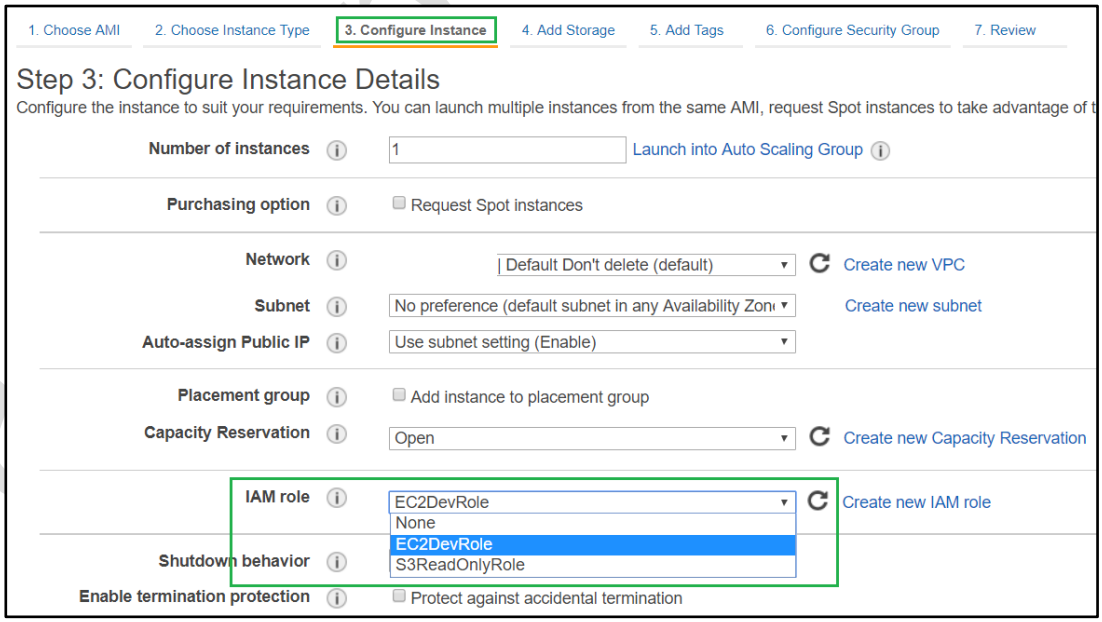


4) Select t2.micro instance to instance.

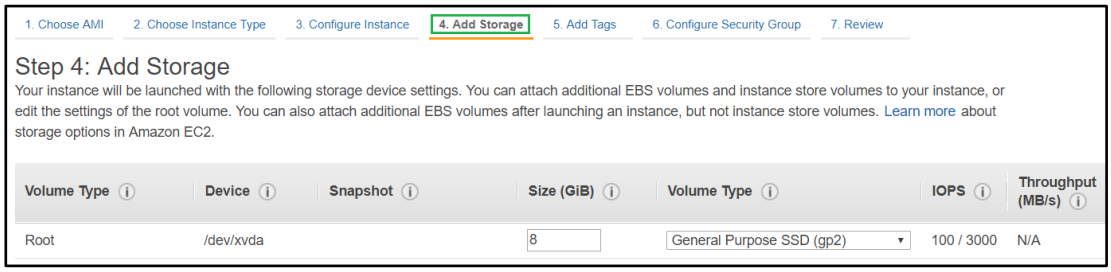


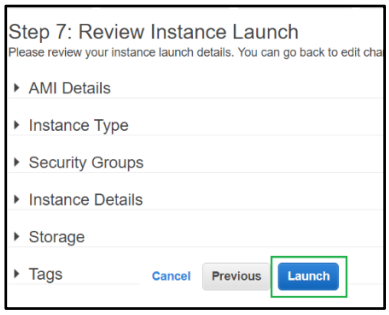
5) Select IAM role which has access to RDS, SQS and S3.

Ideally IAM role should have limited access to the AWS resources.

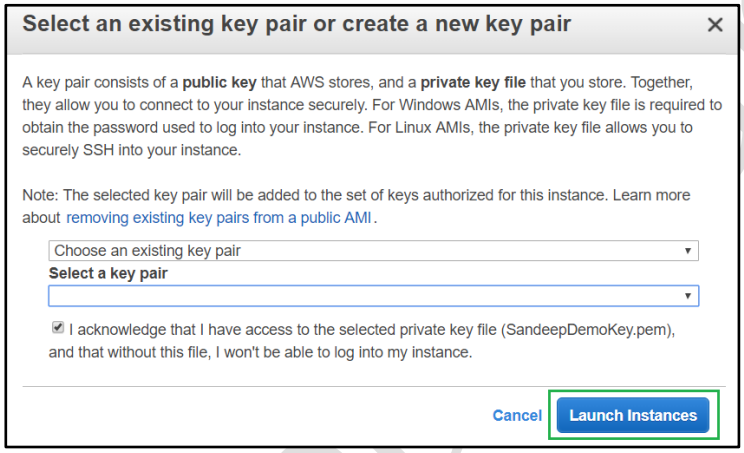


6) Add default storage, proceed to review and launch.

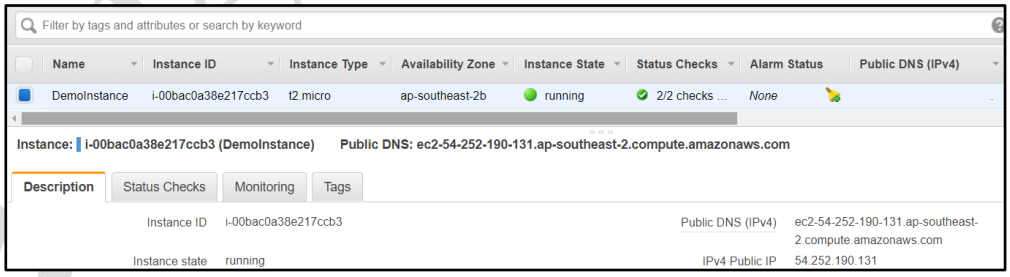




7) Before launching the instance, create a new key pair or use the existing key pair.



8) EC2 instance shall be up and running as shown below.



9) Now login to the EC2 instance using terminal or putty and update the packages.

```
ec2-user@ip-172-31-8-85:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
15 package(s) needed for security, out of 22 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-8-85 ~]$ sudo yum update
```

10) Now check for the java version.

```
ec2-user@ip-172-31-8-85:~
[ec2-user@ip-172-31-8-85 ~]$ java -version
java version "1.7.0_211"
OpenJDK Runtime Environment (amzn-2.6.17.1.79.amzn1-x86_64 u211-b02)
OpenJDK 64-Bit Server VM (build 24.211-b02, mixed mode)
[ec2-user@ip-172-31-8-85 ~]$
```

11) Uninstall Java 1.7 version as shown below.

```
ec2-user@ip-172-31-8-85:~
OpenJDK 64-Bit Server VM (build 24.211-b02, mixed mode)
[ec2-user@ip-172-31-8-85 ~]$ clear
[ec2-user@ip-172-31-8-85 ~]$ sudo yum remove java-1.7.0-openjdk
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
---> Package java-1.7.0-openjdk.x86_64 1:1.7.0.211-2.6.17.1.79.amzn1 will be erase
--> Processing Dependency: jre >= 1.6.0 for package: aws-apitools-mon-1.0.20.0-1.0
--> Processing Dependency: jre >= 1.6.0 for package: aws-apitools-elb-1.0.35.0-1.0
--> Processing Dependency: jre >= 1.6.0 for package: aws-apitools-common-1.1.0-1.9
--> Processing Dependency: jre >= 1.6.0 for package: aws-apitools-ec2-1.7.3.0-1.0.
--> Processing Dependency: jre >= 1.6.0 for package: aws-apitools-as-1.0.61.6-1.0.
```

12) Install Java 1.8.

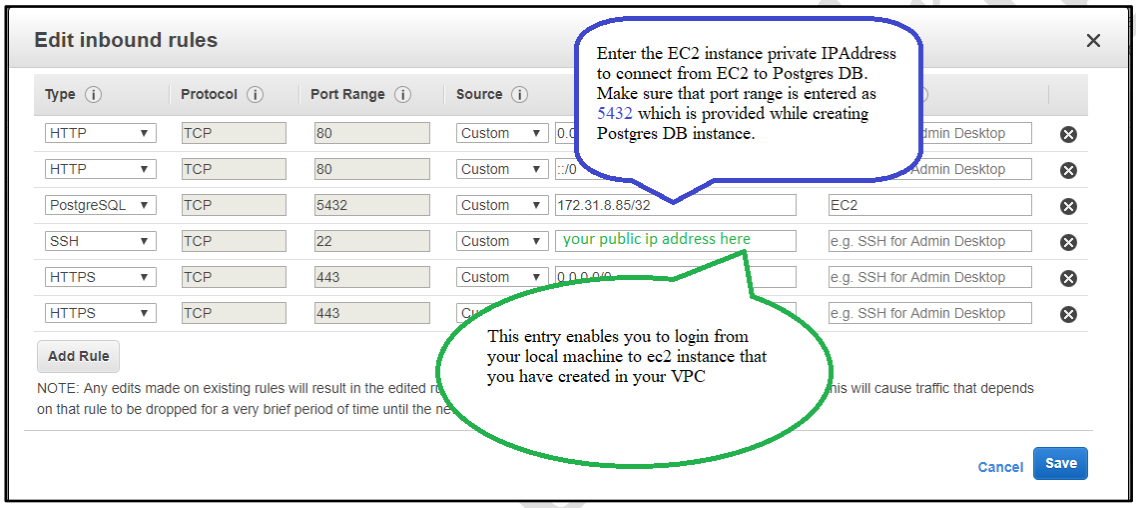
```
ec2-user@ip-172-31-8-85:~
[ec2-user@ip-172-31-8-85 ~]$
[ec2-user@ip-172-31-8-85 ~]$
[ec2-user@ip-172-31-8-85 ~]$
[ec2-user@ip-172-31-8-85 ~]$
[ec2-user@ip-172-31-8-85 ~]$
[ec2-user@ip-172-31-8-85 ~]$ sudo yum install java-1.8.0
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
---> Package java-1.8.0-openjdk.x86_64 1:1.8.0.201.b09-0.43.amzn1 will be
--> Processing Dependency: java-1.8.0-openjdk-headless(x86-64) = 1:1.8.0.2
--> Processing Dependency: libjvm.so(SUNWprivate_1.1)(64bit) for package:
--> Processing Dependency: libjava.so(SUNWprivate_1.1)(64bit) for package:
--> Processing Dependency: libjvm.so()(64bit) for package: 1:java-1.8.0-op
--> Processing Dependency: libjava.so()(64bit) for package: 1:java-1.8.0-o
--> Running transaction check
```

13) Now check for the java version to ensure that 1.8 is installed.

```
ec2-user@ip-172-31-8-85:~
[ec2-user@ip-172-31-8-85 ~]$
[ec2-user@ip-172-31-8-85 ~]$
[ec2-user@ip-172-31-8-85 ~]$ java -version
openjdk version "1.8.0_201"
OpenJDK Runtime Environment (build 1.8.0_201-b09)
OpenJDK 64-Bit Server VM (build 25.201-b09, mixed mode)
[ec2-user@ip-172-31-8-85 ~]$
```

14) Make sure that security group inbound rules have entry to connect to Postgres DB. Go to Services → EC2 → Network & Security → Security groups

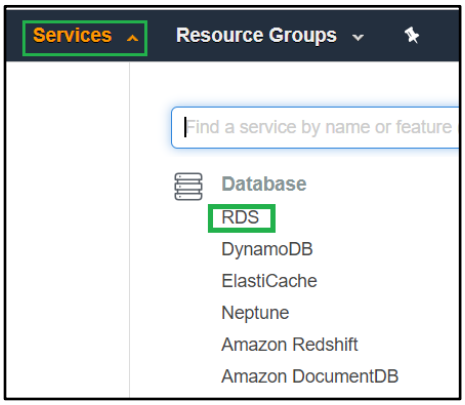
15) Select the Security group associated to EC2 and go to Inbound rules → Click Edit



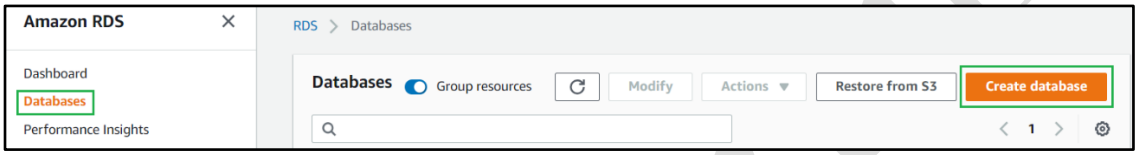
Now EC2 instance is ready for running application.

9. Create RDS Postgres database

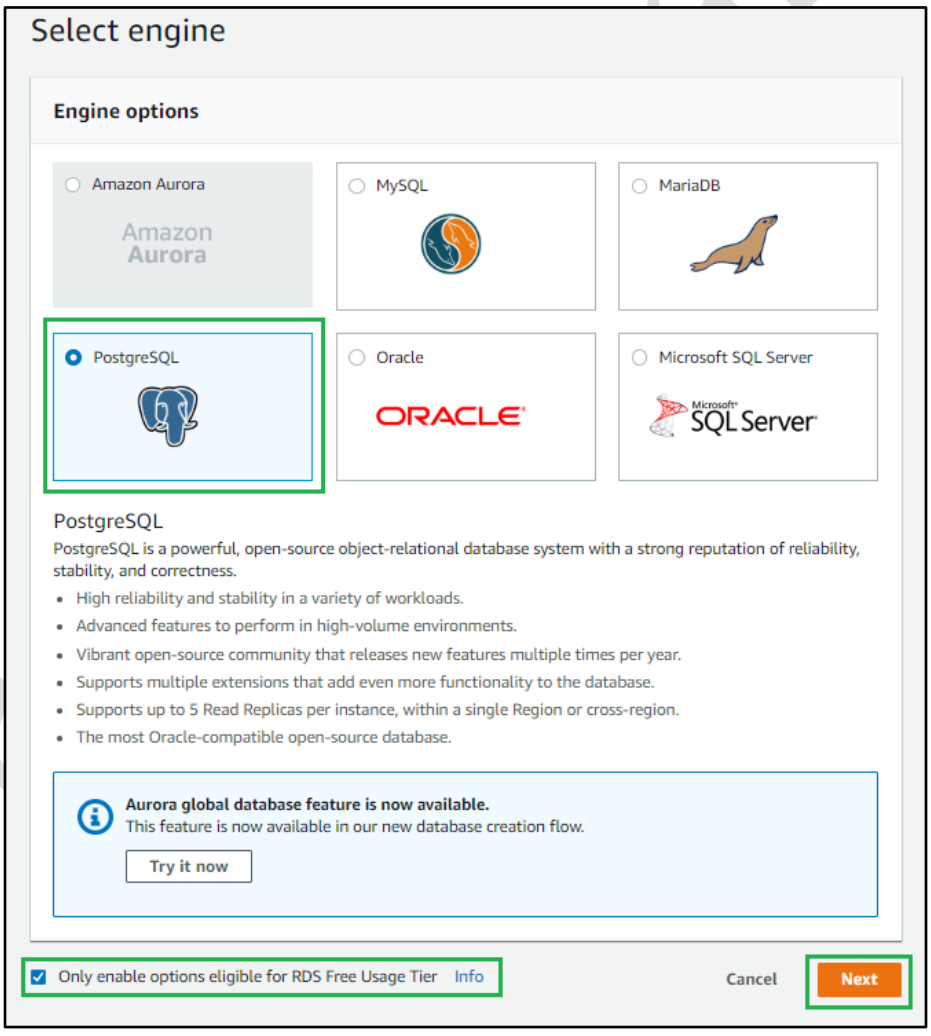
1) Click on Services → Database → RDS



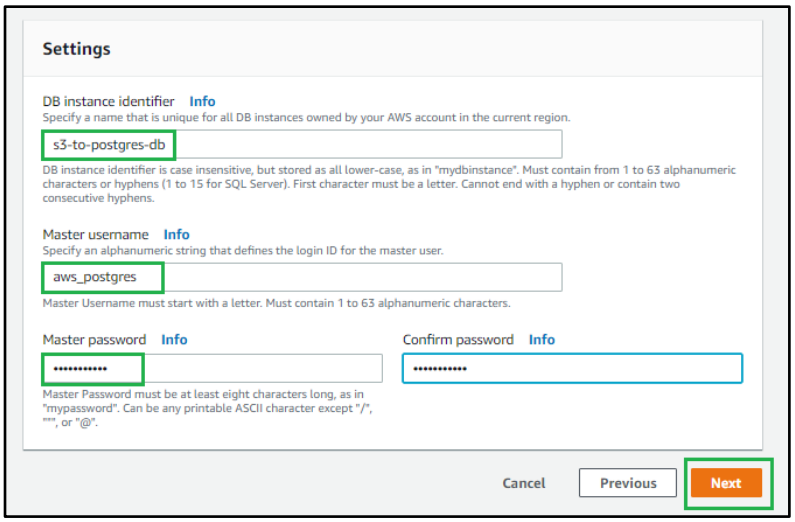
2) Go to databases and create database



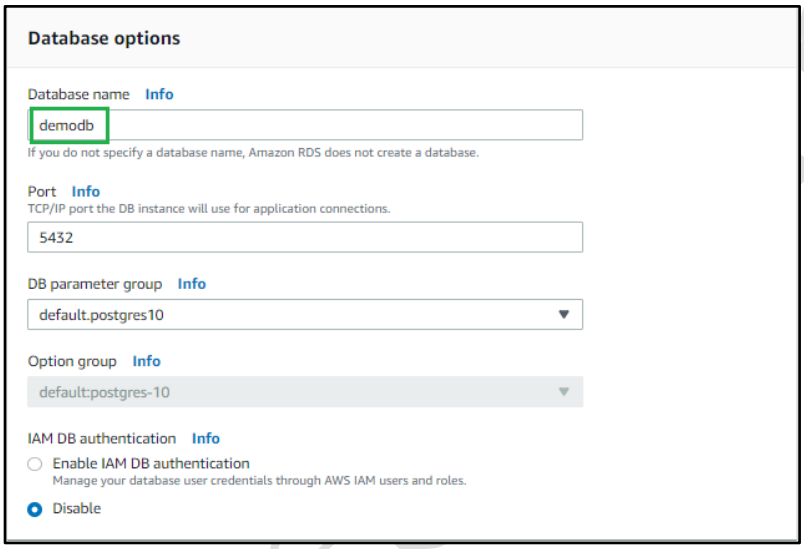
3) Select PostgreSQL engine, tick free tier usage and click next



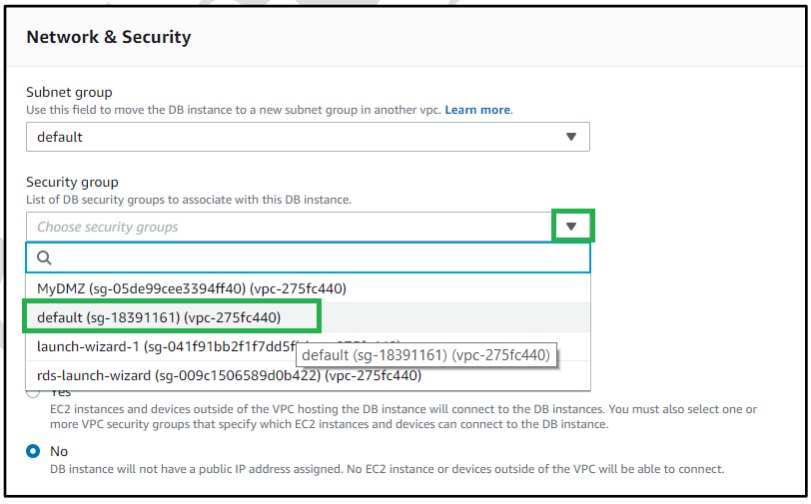
4) Provide highlighted details and note them in a notepad, which can be used for configuration.



5) Provide Database name in Database options section and leave rest to default values.



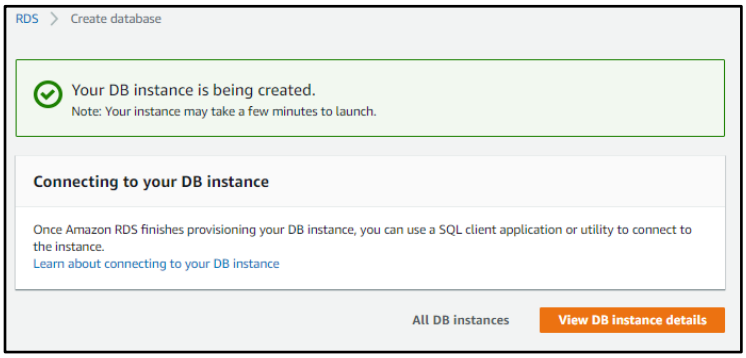
6) In the Network and security section select the security group.



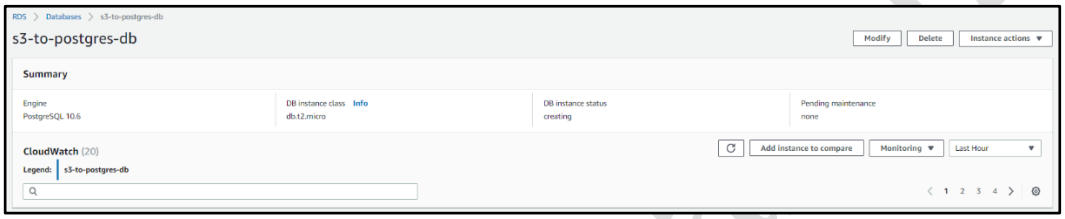
7) Leave below sections to default values and Create database.

- Encryption
- Backup
- Monitoring
- Performance Insights

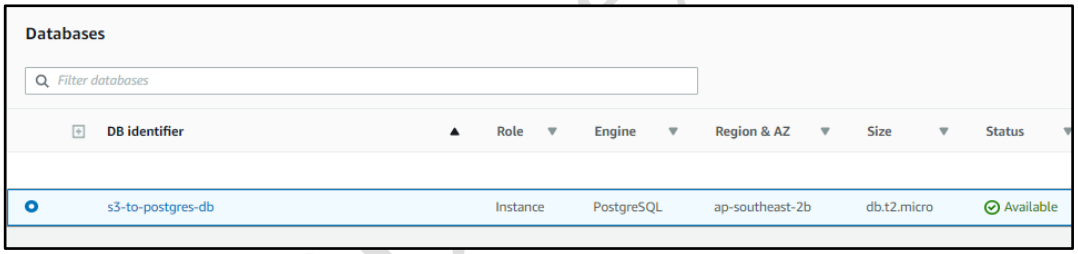
- Log exports
- Maintenance
- Deletion protection



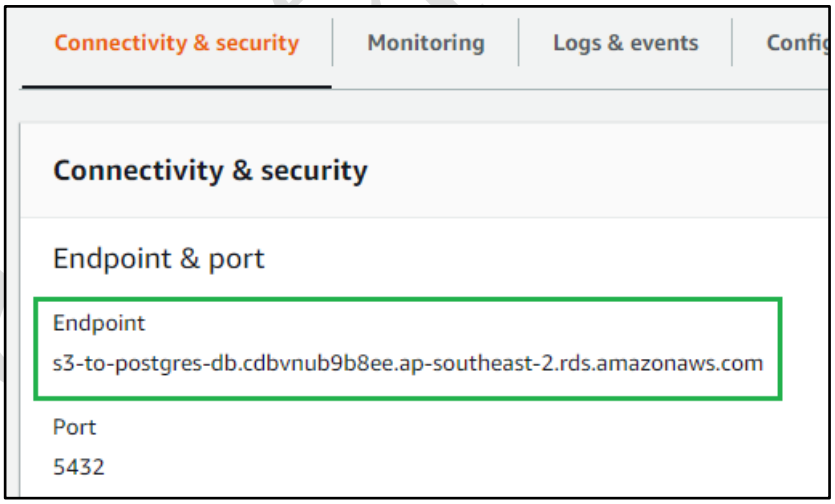
8) It takes few minutes to create the database and shows status as **creating**



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



10) Check for the database endpoint.



11) Now login to the EC2 instance and execute below command to check if EC2 connects to DB created.

psql -h <<DB Endpoint>> -U <<Master username>> -d <<Database name>>

```
ec2-user@ip-172-31-8-85:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Wed May  1 03:09:30 2019 from 113.163.220.203.dial.dynamic.acc01-gur
w-wag.comindico.com.au

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-8-85 ~]$ psql -h s3-to-postgres-db.cdbvnub9b8ee.ap-southeast-2.rds.amazonaws.com -U aws_postgres -d demodb
-bash: psql: command not found
```

12) To connect to postgres db install postgresql client as shown below.

```
ec2-user@ip-172-31-8-85:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Wed May  1 03:09:30 2019 from 113.163.220.203.dial.dynamic.acc01-gur
w-wag.comindico.com.au

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-8-85 ~]$ psql -h s3-to-postgres-db.cdbvnub9b8ee.ap-southeast-2.rds.amazonaws.com -U aws_postgres -d demodb
-bash: psql: command not found
[ec2-user@ip-172-31-8-85 ~]$ sudo yum install postgresql client
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main
amzn-updates
No package client available.
Resolving Dependencies
--> Running transaction check
```

13) Now login to the EC2 instance and execute below command to check if EC2 connects to DB created.

psql -h <<DB Endpoint>> -U <<Master username>> -d <<Database name>>

```
ec2-user@ip-172-31-8-85:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Wed May  1 04:21:25 2019 from 113.163.220.203.dial.dynamic.acc01-gurw-wag.comindico.com.au

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-8-85 ~]$ psql -h s3-to-postgres-db.cdbvnub9b8ee.ap-southeast-2.rds.amazonaws.com -U aws_postgres -d demodb
Password for user aws_postgres:
psql (9.2.24, server 10.6)
WARNING: psql version 9.2, server version 10.0.
         Some psql features might not work.
SSL connection (cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256)
Type "help" for help.

demodb=>
```

14) Check if any tables are found.

```
ec2-user@ip-172-31-8-85:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Wed May  1 04:21:25 2019 from 113.163.220.203.dial.dynamic.acc01-gurw-wag.comindico.com.au

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-8-85 ~]$ psql -h s3-to-postgres-db.cdbvnub9b8ee.ap-southeast-2.rds.amazonaws.com -U aws_postgres -d demodb
Password for user aws_postgres:
psql (9.2.24, server 10.6)
WARNING: psql version 9.2, server version 10.0.
         Some psql features might not work.
SSL connection (cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256)
Type "help" for help.

demodb=> \dt
No relations found.
```

15) Create table as shown below.

```
demodb=> \dt
No relations found.
demodb=> CREATE TABLE employees (
demodb(> employee_id VARCHAR(10),
demodb(> first_name VARCHAR(20),
demodb(> last_name VARCHAR(20),
demodb(> title VARCHAR(20),
demodb(> email VARCHAR(50)
demodb(> );
CREATE TABLE
demodb=>
```

16) Check if table is created.

```
demodb=> \dt
             List of relations
 Schema |   Name    | Type  |    Owner
--------+-----------+-------+--------------
 public | employees | table | aws_postgres
(1 row)

demodb=> select * from employees;
 employee_id | first_name | last_name | title | email
-------------+------------+-----------+-------+-------
(0 rows)

demodb=>
```

17) Exit from the Postgres DB

```
demodb=> \q
[ec2-user@ip-172-31-8-85 ~]$
```

10. Build the application

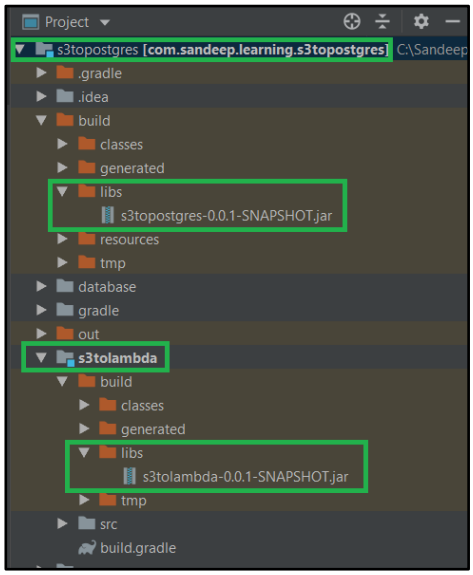
1) S3 to Postgres project has a lambda module. When built it generates two jar files.
    a) s3topostgres-0.0.1-SNAPSHOT.jar
    b) s3tolambda-0.0.1-SNAPSHOT.jar
2) Clean the code once from the terminal

```
Terminal:  Local  +
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Sandeep\Work\IJWorkspace\s3topostgres>gradle clean

BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 up-to-date
C:\Sandeep\Work\IJWorkspace\s3topostgres>
```

3) Build the code.

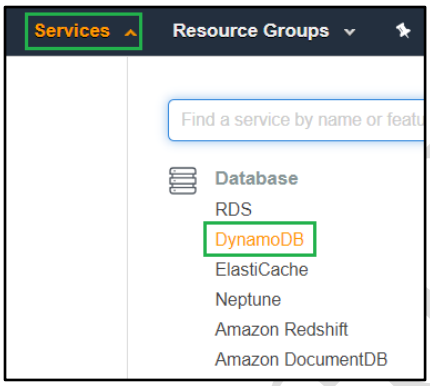
```
Terminal:  Local  +
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Sandeep\Work\IJWorkspace\s3topostgres>gradle build

BUILD SUCCESSFUL in 15s
5 actionable tasks: 5 executed
C:\Sandeep\Work\IJWorkspace\s3topostgres>
```

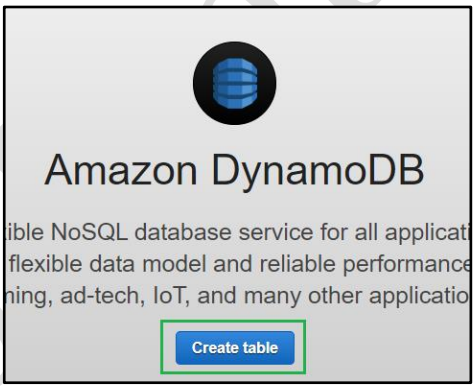
4) Check if jars are built.



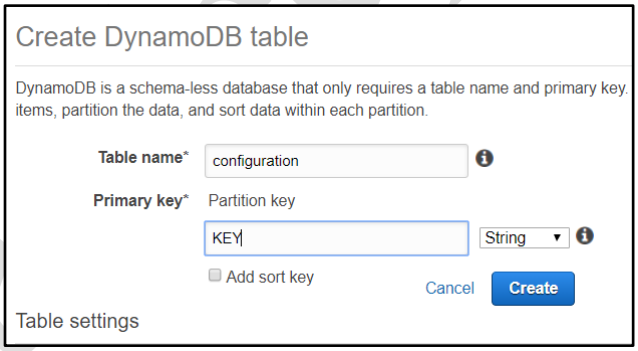
11. Lambda configuration in Dynamo DB

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

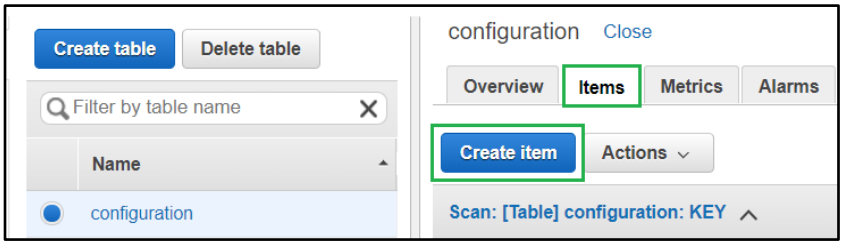


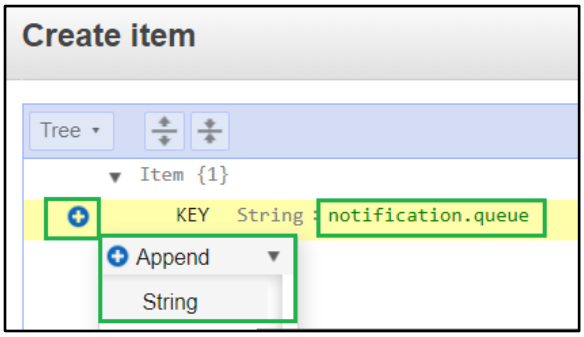


2) Provide the Table Name and Primary Key Column with Data Type.

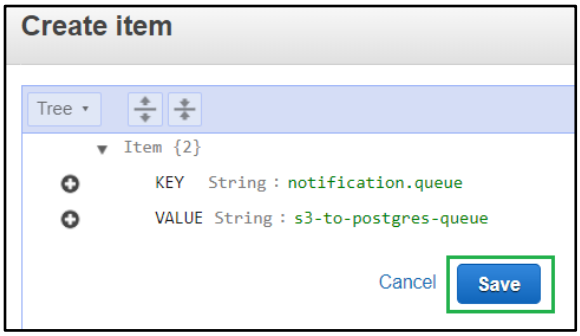


3) Table is created, now create item.



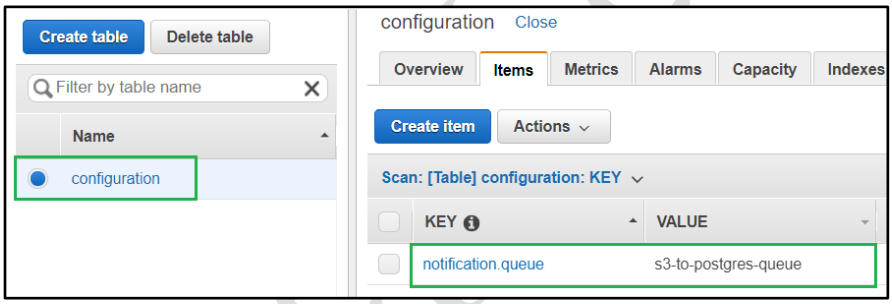


4) Once item is created, Save it.



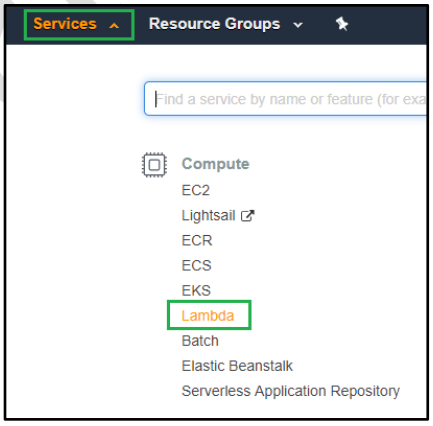
5) Upon saving, it would look like below.

KEY: **notification.queue**

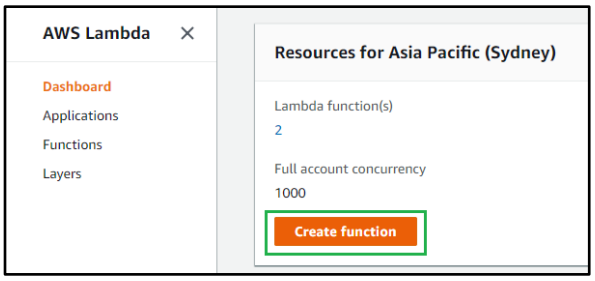


12. Deploying Lambda

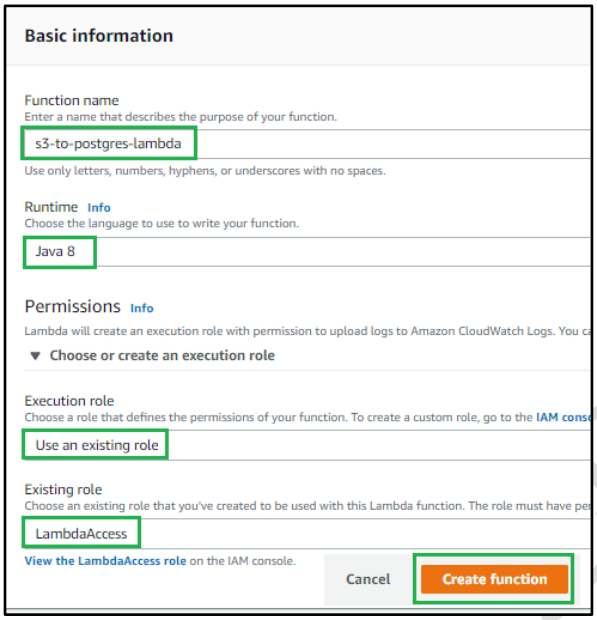
1) Go to Services → Compute → Lambda



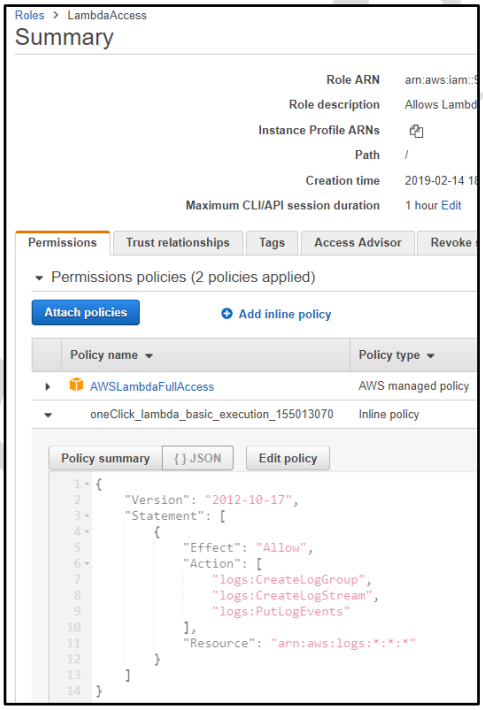
2) Click create function.



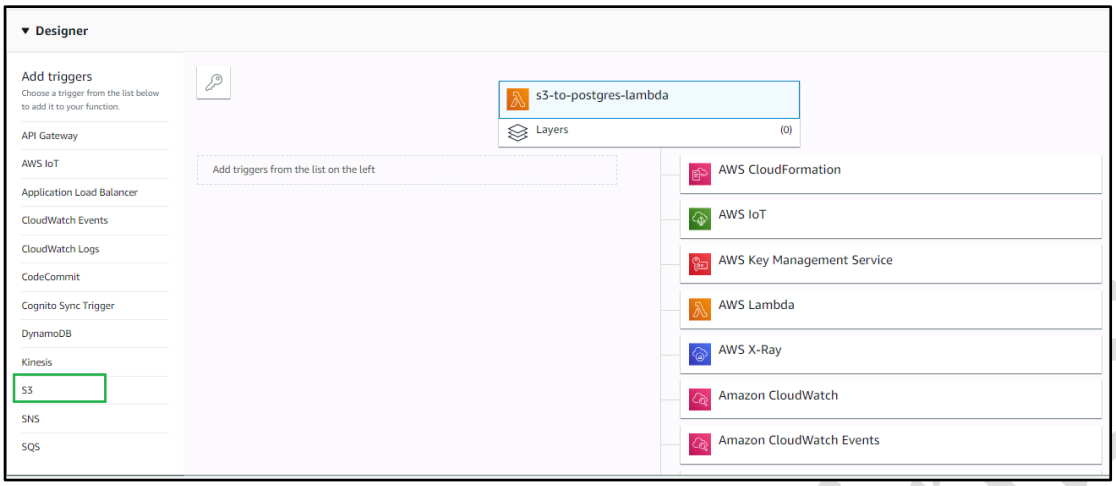
3) Select Author from scratch and provide the basic information.



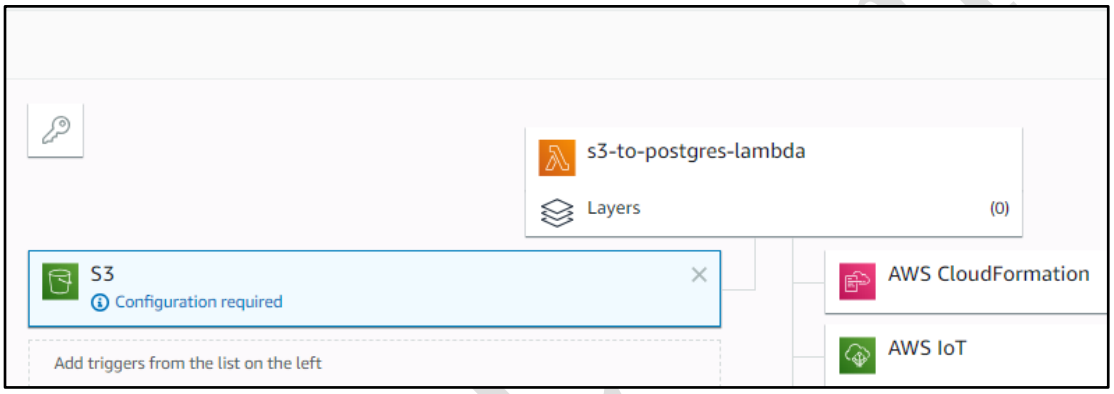
4) Make sure that Lambda Access role is attached with below policies.



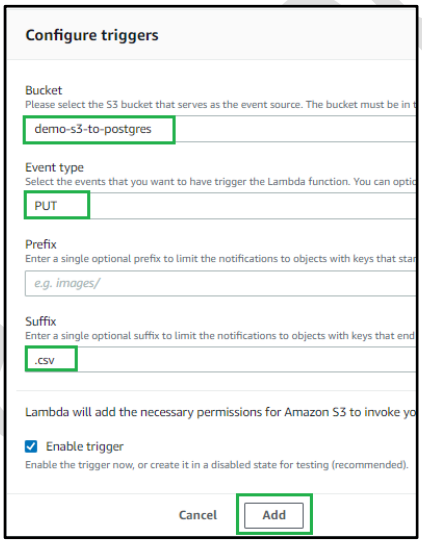
5) Select S3 from the left pane to add the trigger.



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

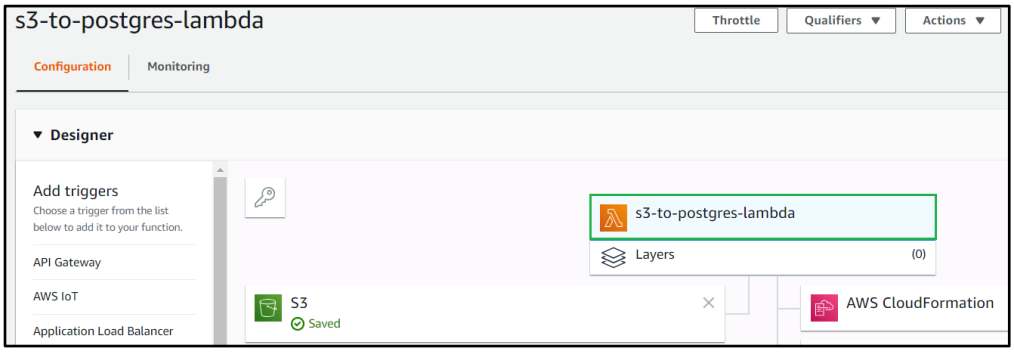


7) Proceed with trigger configuration as shown. Select the **bucket name** that is created in Create Bucket section, Add event type as **PUT** (when file is PUT in bucket, it triggers Lambda) for the files suffixed with **.csv** extension.



8) Click save button on the top.

9) Now select the lambda as shown below.

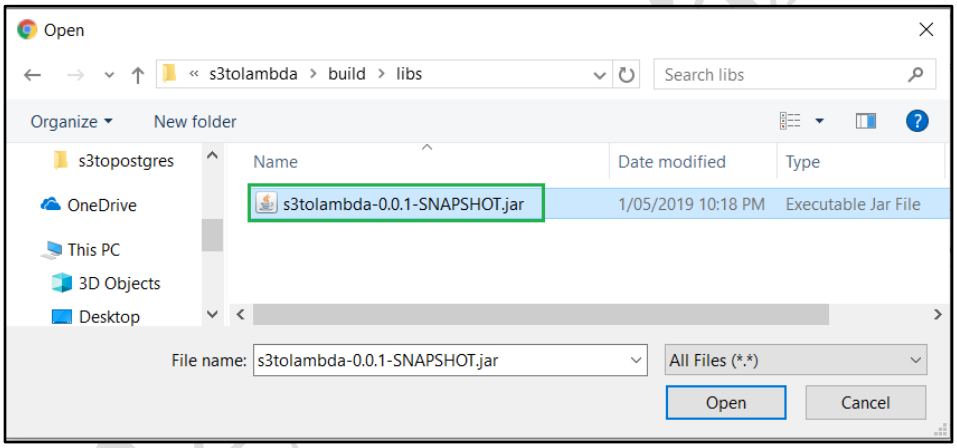


10) In the function code section, select the below shown configuration and click on upload.

For the Handler, provide below value –

com.sandeep.learning.s3tolambda.handler.NotificationHandler::handleRequest

11) Select the jar from local machine and upload.



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

13. Run Application

1) Login to EC2 instance and create a directory as shown below.

2) Upload the generated s3topostgres-0.0.1-SNAPSHOT.jar file to the directory created above.

```
[ec2-user@ip-172-31-8-85 s3-to-postgres]$ pwd
/home/ec2-user/s3-to-postgres
[ec2-user@ip-172-31-8-85 s3-to-postgres]$ ls
s3topostgres-0.0.1-SNAPSHOT.jar
[ec2-user@ip-172-31-8-85 s3-to-postgres]$
```

3) Copy the application.yml file to the below shown location.

ec2-user@ip-172-31-8-85:~/s3-to-postgres

```
[ec2-user@ip-172-31-8-85 s3-to-postgres]$
[ec2-user@ip-172-31-8-85 s3-to-postgres]$
[ec2-user@ip-172-31-8-85 s3-to-postgres]$ ls
application.yml  s3topostgres-0.0.1-SNAPSHOT.jar
[ec2-user@ip-172-31-8-85 s3-to-postgres]$
```

4) Modify below highlighted configuration in application.yml.

- datasource:url:

  jdbc:postgresql:// <<DB Endpoint>>:<<DBPort>>/<<DatabaseName>>
- queue: As created in SQS
- logging:file: Where the log file should be generated.

```
[ec2-user@ip-172-31-8-85 s3-to-postgres]$ cat application.yml
spring:
  profile:
    active: dev
  application:
    name: s3-to-postgres
  datasource:
    url: "jdbc:postgresql://s3-to-postgres-db.cdbvnub9b8ee.ap-southeast-2.rds.amazonaws.com:5432/demodb"
    username: "aws_postgres"                                    #Postgres DB Username
    password: "postgres123"                                    #Postgres DB Password
    hikari:
      maximumPoolSize: 10                                  #Postgres DB Connection Pool size

server:
  port: 7070
  servlet:
    context-path: /s3-to-postgres

s3topostgres:
  sqs:
    notification:
      queue: s3-to-postgres-queue

logging:
  file: /home/ec2-user/s3-to-postgres/s3-to-postgres.log
  level:
    com.sandeep.learning.s3topostgres: DEBUG
    org.springframework.web: ERROR
  pattern:
    console: "%d{yyyy-MM-dd HH:mm:ss} - %msg%n"
    file: "%d{yyyy-MM-dd HH:mm:ss} [thread] %-5level %logger{36} - %msg%n"
```

5) Create a start.sh file with the below content.

java -jar s3topostgres-0.0.1-SNAPSHOT.jar --spring.config.location=./application.yml &

```
[ec2-user@ip-172-31-8-85 s3-to-postgres]$ cat start.sh
java -jar s3topostgres-0.0.1-SNAPSHOT.jar --spring.config.location=./application.yml &
[ec2-user@ip-172-31-8-85 s3-to-postgres]$
```

6) Overall directory should have below list of look like below.

```
[ec2-user@ip-172-31-8-85 s3-to-postgres]$ ls
application.yml  s3topostgres-0.0.1-SNAPSHOT.jar  start.sh
[ec2-user@ip-172-31-8-85 s3-to-postgres]$
```

7) Now application is ready for testing.

8) Run ./start.sh file

```
[ec2-user@ip-172-31-8-85 s3-to-postgres]$ ./start.sh
[ec2-user@ip-172-31-8-85 s3-to-postgres]$
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.4.RELEASE)

2019-05-03 10:29:04 - Starting S3topostgresApplication on ip-17
3-to-postgres)
2019-05-03 10:29:04 - Running with Spring Boot v2.1.4.RELEASE,
2019-05-03 10:29:04 - No active profile set, falling back to de
```

14. Testing

1) Login to the EC2 and connect to PostgreSQL by using below command.

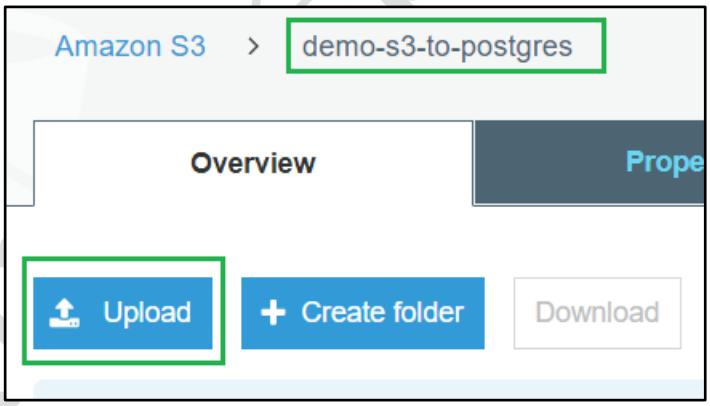
   psql -h <DBEndpoint> -U <Master Username> -d <Database name>

```
[ec2-user@ip-172-31-8-85 s3-to-postgres]$
[ec2-user@ip-172-31-8-85 s3-to-postgres]$ psql -h s3-to-postgres-db.cdbvnub9b8ee.ap-southeast-2.rds.amazonaws.com -U aws_postgres -d demodb
Password for user aws_postgres:
psql (9.2.24, server 10.6)
WARNING: psql version 9.2, server version 10.0.
         Some psql features might not work.
SSL connection (cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256)
Type "help" for help.

demodb=>
```

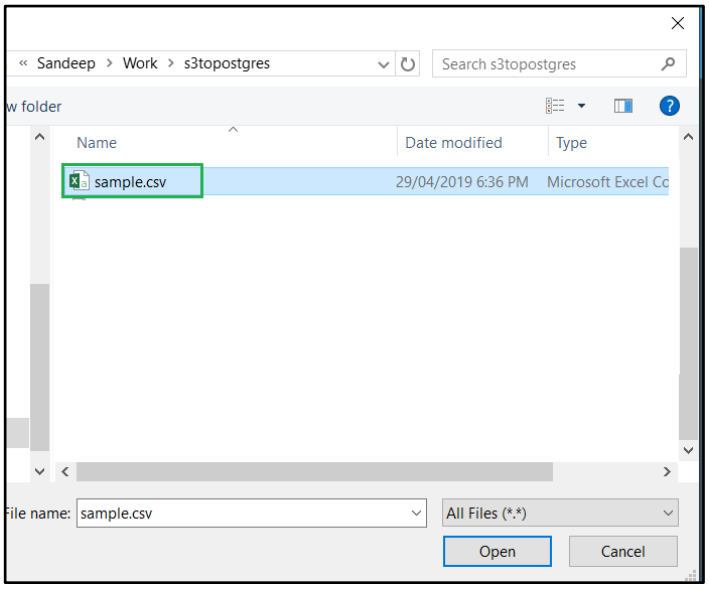
2) Check if the table has any records.

```
demodb=> select * from employees;
 employee_id | first_name | last_name | title | email
-------------+------------+-----------+-------+-------
(0 rows)

demodb=>
```

3) Now Go to S3 bucket and click upload

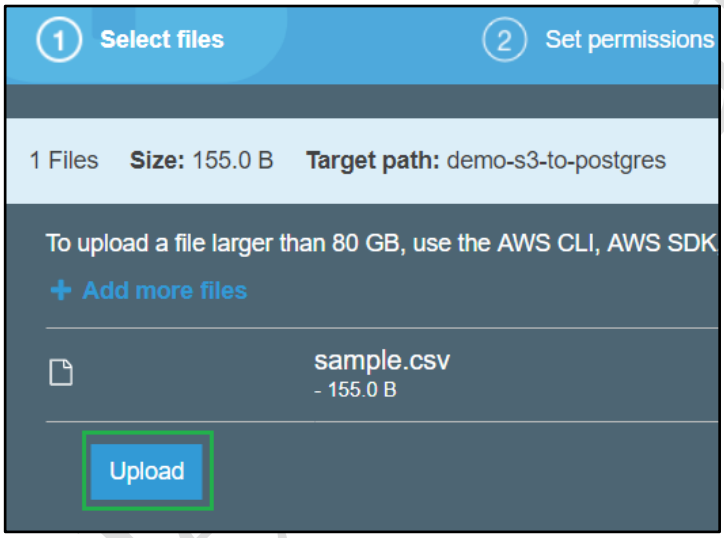


4) Upload sample.csv

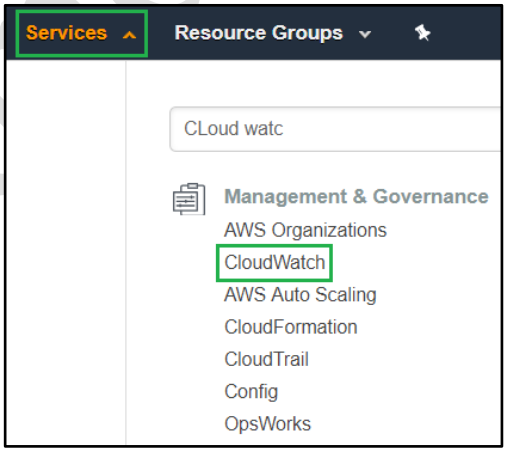


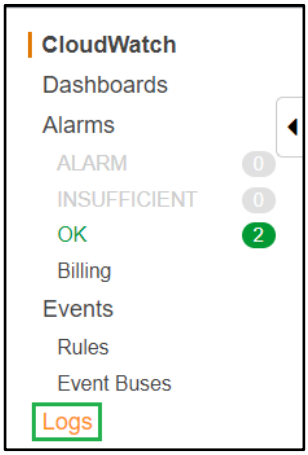
Note: sample data (sample.csv) is available in the resources folder of the project.

5) Click upload.

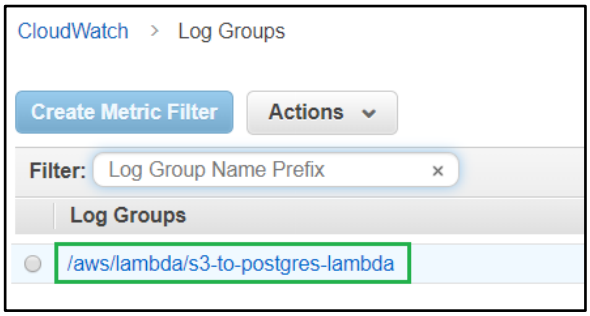


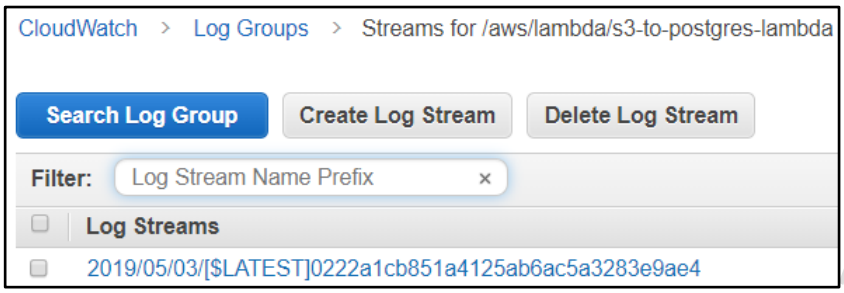
6) Go to Services → Management & Governance → Cloud Watch → Logs



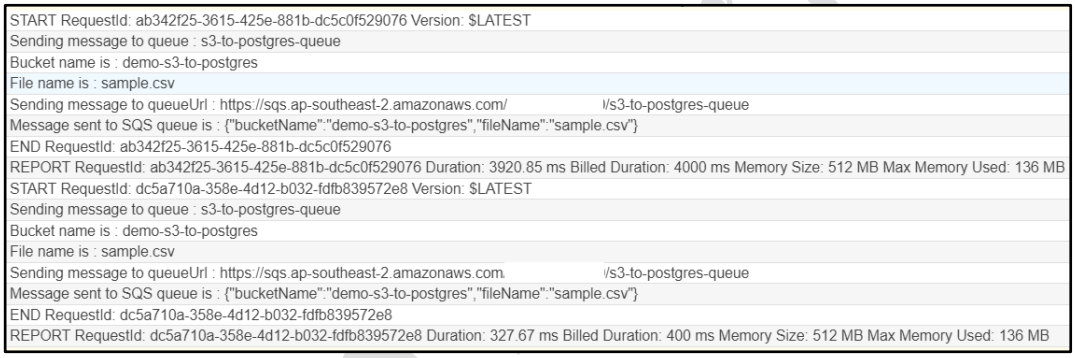


7) Click on generated logs

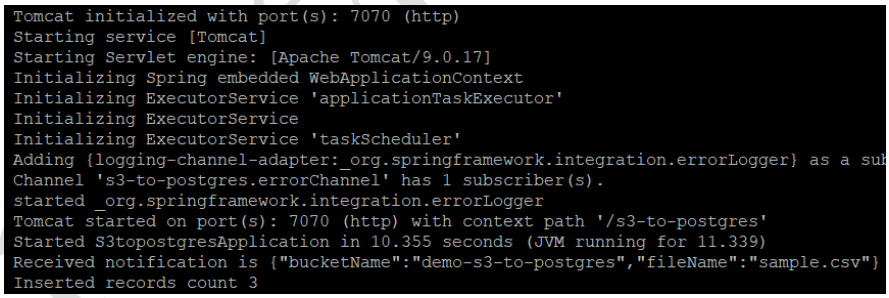




8) Click on this to see the Lambda logs.



9) Now check the logs on EC2.



10) Check the records inserted in database table.

```
demodb=> select * from employees;
 employee_id | first_name | last_name |   title    |         email
-------------+------------+-----------+------------+-----------------------
 EMP12345    | Tony       | Stark     | Iron Man   | tony.stark@mail.com
 EMP45678    | Peter      | Parker    | Spider Man | peter.parker@mail.com
 EMP21234    | Clark      | Kent      | Super Man  | clark.kent@mail.com
```