

S3 to Postgres Documentation

Table of Contents

1. Use case.....	3
2. Project Description.....	3
3. Architecture	3
4. Flow	3
5. Steps	3
6. Create S3 bucket	4
7. Create SQS queue	5
8. Create EC2 Instance	8
9. Create RDS Postgres database.....	13
10. Build the application	17
11. Lambda configuration in Dynamo DB	18
12. Deploying Lambda.....	19
13. Run Application	22
14. Testing.....	24

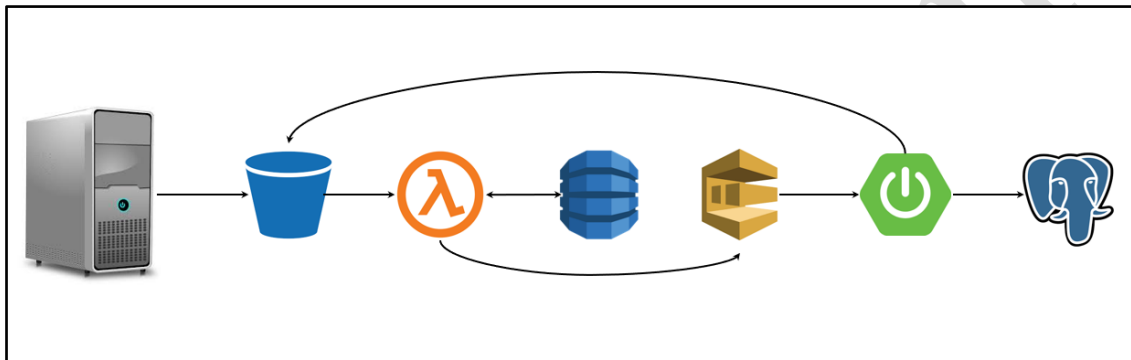
1. Use case

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

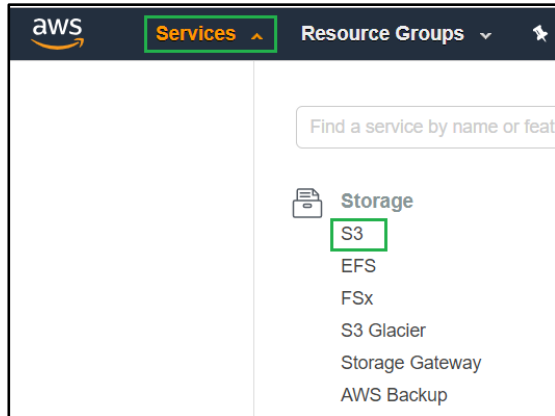
To run the s3-to-postgres application, follow below steps.

- 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](#)

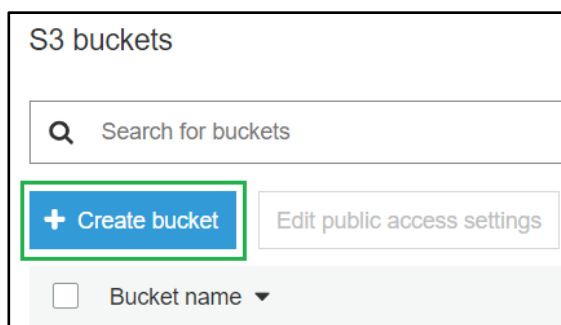
9) [Testing](#)

6. Create S3 bucket

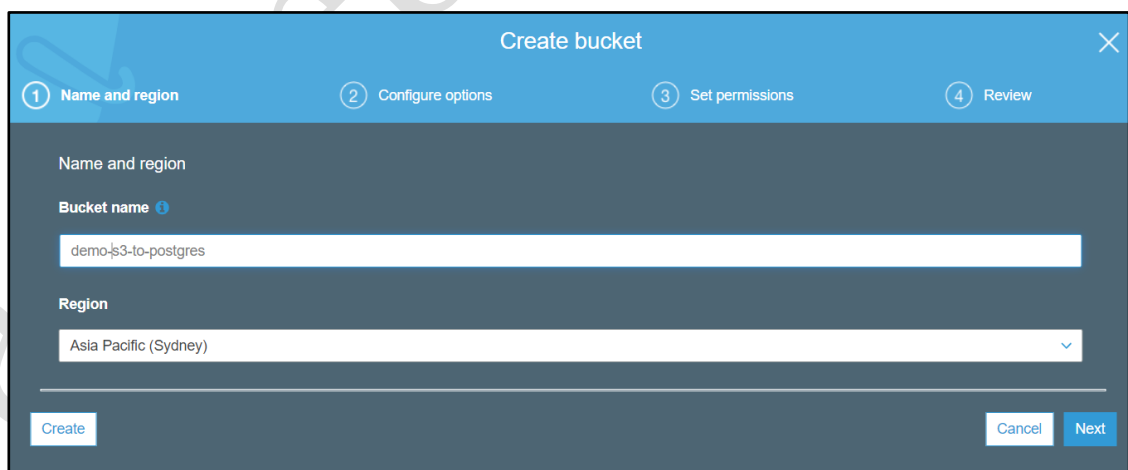
- 1) Click on Services → Storage → S3



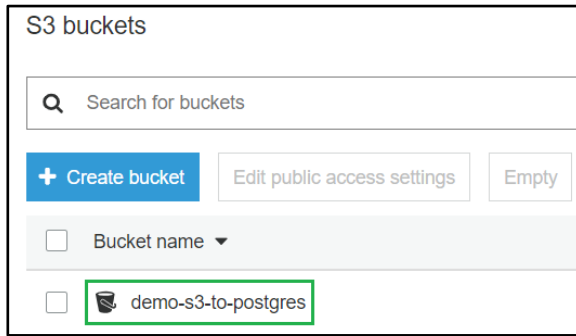
- 2) Click on create bucket



- 3) Provide a bucket name

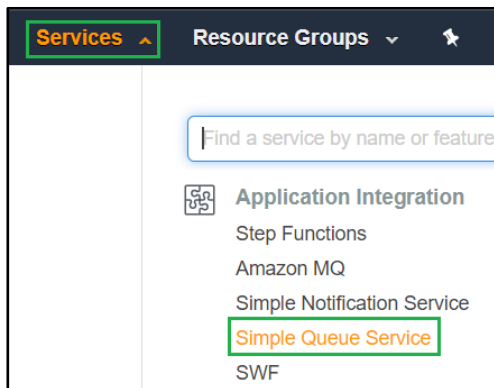


- 4) Bucket gets created as shown below.

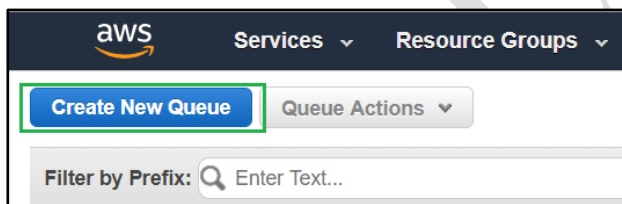


7. Create SQS queue

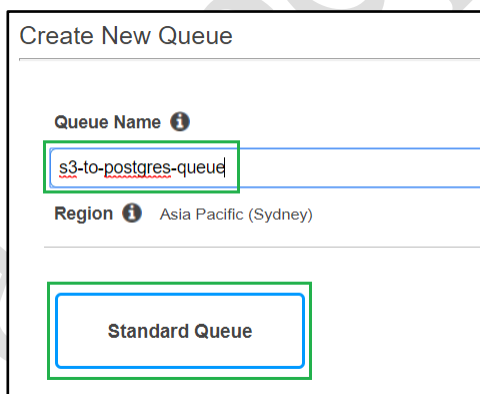
- 1) Click on Services → Application Integration → Simple Queue Service



- 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.

1 S3 Queue selected

Details **Permissions** Redrive Policy Monitoring Tags Encryption Lambda Triggers

Add a Permission Edit Policy Document (Advanced) What's an S3 Queue Access Policy?

Effect	Principals	Actions	Conditions
This queue has an empty S3 Queue Access Policy . This means that only the queue owner is allowed to use it. You can Add a Permission to grant another account access.			

- 5) Select the IAM role which has access to S3.

Roles > **EC2DevRole**

Summary

Permissions Trust relationships Tags

Permissions policies (3 policies applied)

Attach policies

Policy name
AmazonRDSFullAccess
AmazonSQSFullAccess
AmazonS3FullAccess

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

Add a Permission to s3-to-postgres-queue

Permissions enable you to control which operations a user can perform on a queue. [Click here](#) about access control concepts.

Effect **Allow** ☐ Deny

Principal **arn:aws:iam::account number:role/EC2DevRole** ☐ Everybody (*)

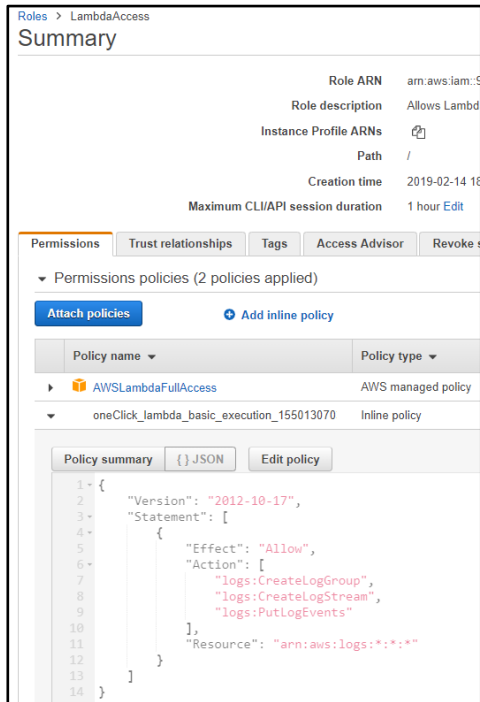
Use commas between multiple values.

Actions **1 Specific Action** ☐ All S3 Actions (SQS:*)

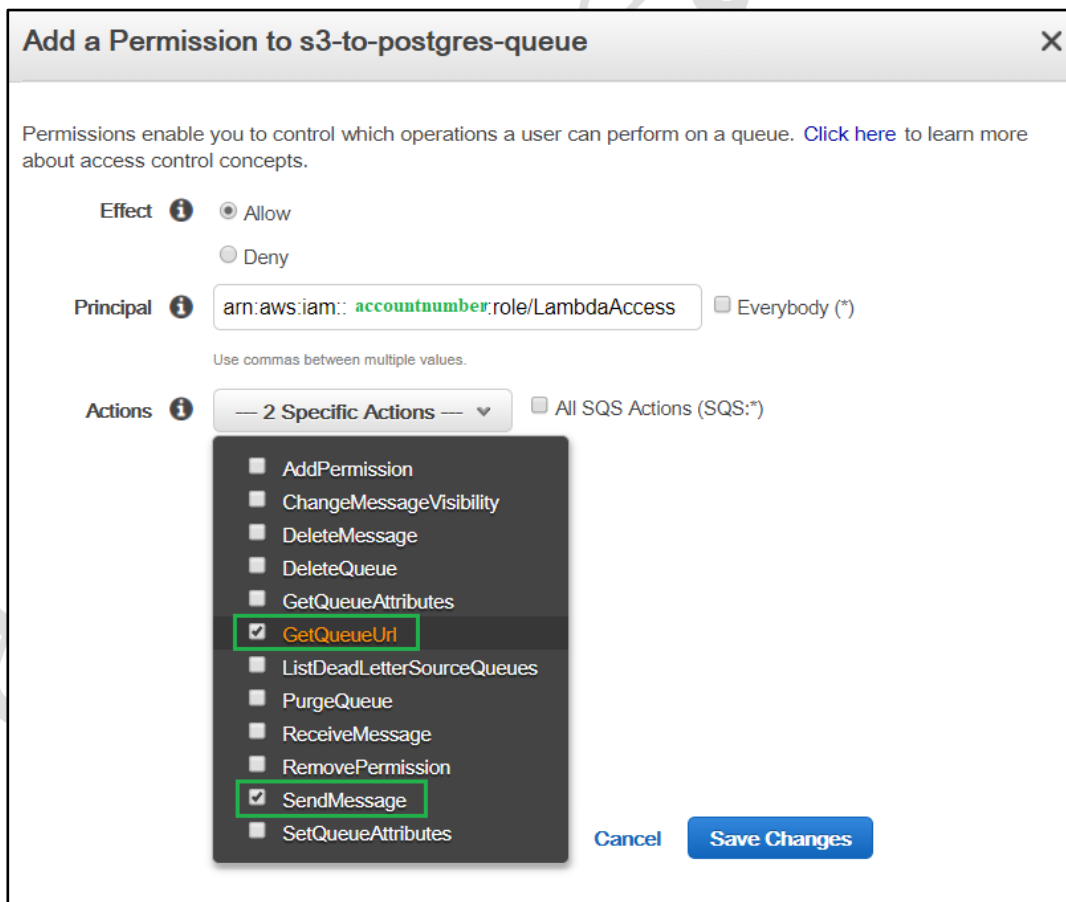
- ☐ AddPermission
- ☐ ChangeMessageVisibility
- ☐ DeleteMessage
- ☐ DeleteQueue
- ☐ GetQueueAttributes
- ☐ GetQueueUrl
- ☐ ListDeadLetterSourceQueues
- ☐ PurgeQueue
- ☒ **ReceiveMessage**
- ☐ RemovePermission
- ☐ SendMessage
- ☐ SetQueueAttributes

Cancel Save Changes

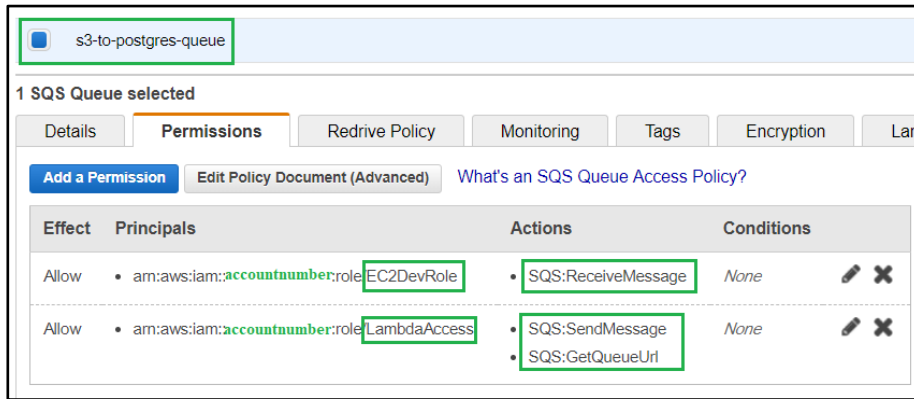
- 7) Make sure that LambdaAccessRole is attached with below policies.



- 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.

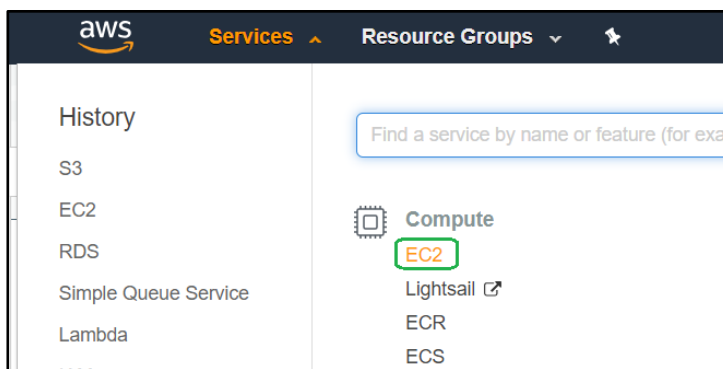


- 9) Overall permissions on the SQS queue should look like something below.



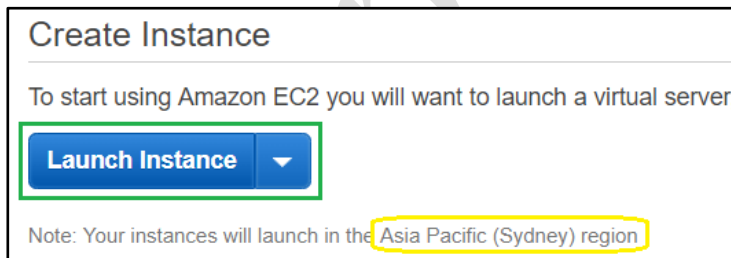
8. Create EC2 Instance

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

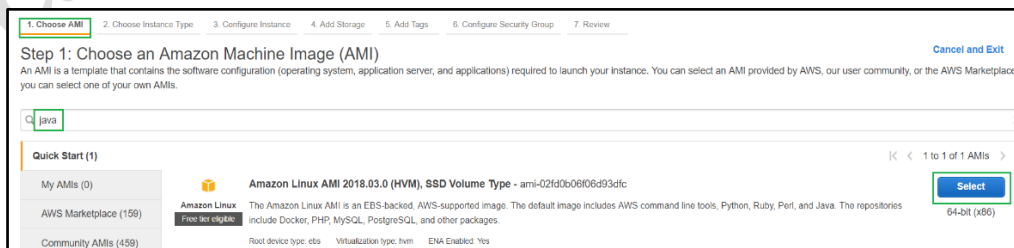


Make sure that you select region where the service is to be deployed.

- 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.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your needs.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only
<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only

Cancel Previous Review and Launch Next: Configure Instance Details

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

Roles > EC2DevRole

Summary

Permissions Trust relationships Tags

Permissions policies (3 policies applied)

Attach policies

Policy name
AmazonRDSFullAccess
AmazonSQSFullAccess
AmazonS3FullAccess

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

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of t

Number of instances 1 Launch into Auto Scaling Group

Purchasing option ☐ Request Spot instances

Network Default Don't delete (default) Create new VPC

Subnet No preference (default subnet in any Availability Zone) Create new subnet

Auto-assign Public IP Use subnet setting (Enable)

Placement group ☐ Add instance to placement group

Capacity Reservation Open Create new Capacity Reservation

IAM role EC2DevRole
None
EC2DevRole
S3ReadOnlyRole Create new IAM role

Shutdown behavior

Enable termination protection ☐ Protect against accidental termination

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

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)
Root	/dev/xvda		8	General Purpose SSD (gp2)	100 / 3000	N/A

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit details.

- AMI Details
- Instance Type
- Security Groups
- Instance Details
- Storage
- Tags

[Cancel](#) [Previous](#) [Launch](#)

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

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair

Select a key pair

☒ I acknowledge that I have access to the selected private key file (SandeepDemoKey.pem), and that without this file, I won't be able to log into my instance.

[Cancel](#) [Launch Instances](#)

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

Filter by tags and attributes or search by keyword							
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
Demolnstance	i-00bac0a38e217ccb3	t2.micro	ap-southeast-2b	running	2/2 checks ...	None	
Instance: i-00bac0a38e217ccb3 (Demolnstance) Public DNS: ec2-54-252-190-131.ap-southeast-2.compute.amazonaws.com							
<div> <div>Description</div> <div>Status Checks</div> <div>Monitoring</div> <div>Tags</div> </div>							
Instance ID: i-00bac0a38e217ccb3				Public DNS (IPv4): ec2-54-252-190-131.ap-southeast-2.compute.amazonaws.com			
Instance state: running				IPv4 Public IP: 54.252.190.131			

- 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 erased  
--> 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

Type	Protocol	Port Range	Source	Destination
HTTP	TCP	80	Custom	Admin Desktop
HTTP	TCP	80	Custom	Admin Desktop
PostgreSQL	TCP	5432	Custom	EC2
SSH	TCP	22	Custom	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom	e.g. SSH for Admin Desktop

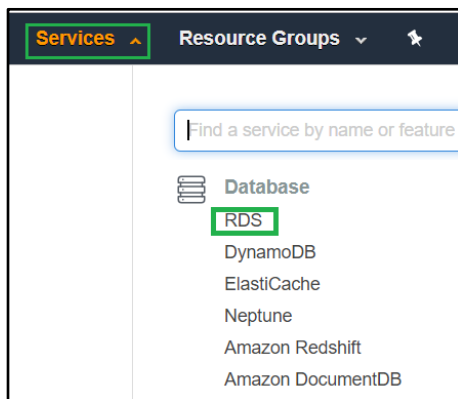
NOTE: Any edits made on existing rules will result in the edited rule being dropped for a very brief period of time until the new rule is applied.

Cancel Save

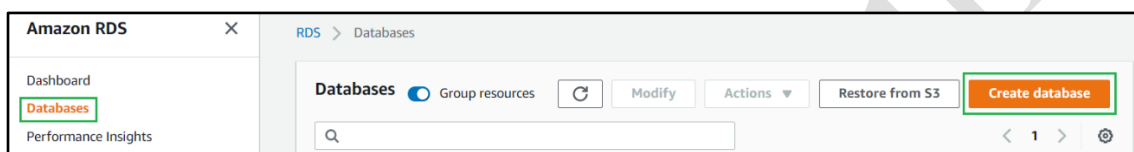
Now EC2 instance is ready for running application.

9. Create RDS Postgres database

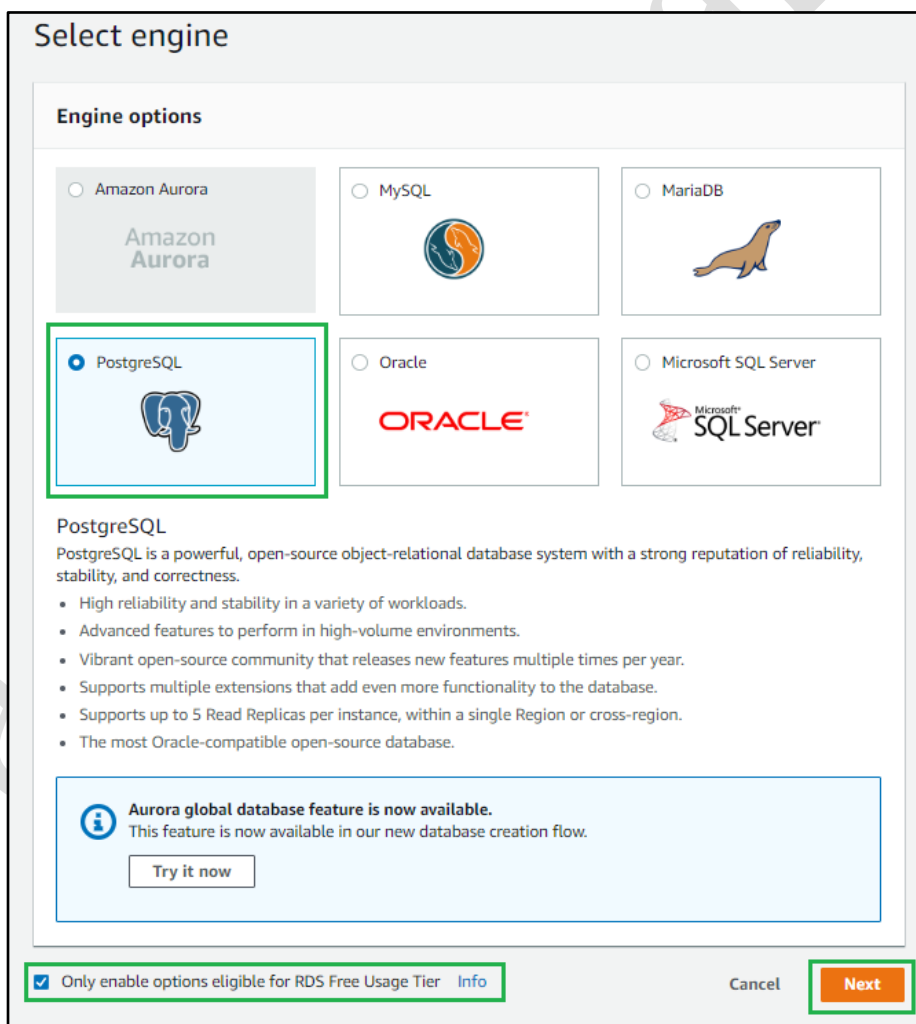
- 1) Click on Services → Database → RDS



- 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.

Settings

DB instance identifier [Info](#)
Specify a name that is unique for all DB instances owned by your AWS account in the current region.

Master username [Info](#)
Specify an alphanumeric string that defines the login ID for the master user.

Master password [Info](#) **Confirm password** [Info](#)
Master Password must be at least eight characters long, as in "mypassword". Can be any printable ASCII character except "/", "", or "@".

[Cancel](#) [Previous](#) [Next](#)

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

Database options

Database name [Info](#)

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

Port [Info](#)
TCP/IP port the DB instance will use for application connections.

DB parameter group [Info](#)

Option group [Info](#)

IAM DB authentication [Info](#)
☐ Enable IAM DB authentication
Manage your database user credentials through AWS IAM users and roles.
☒ Disable

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

Network & Security

Subnet group
Use this field to move the DB instance to a new subnet group in another vpc. [Learn more.](#)

Security group
List of DB security groups to associate with this DB instance.

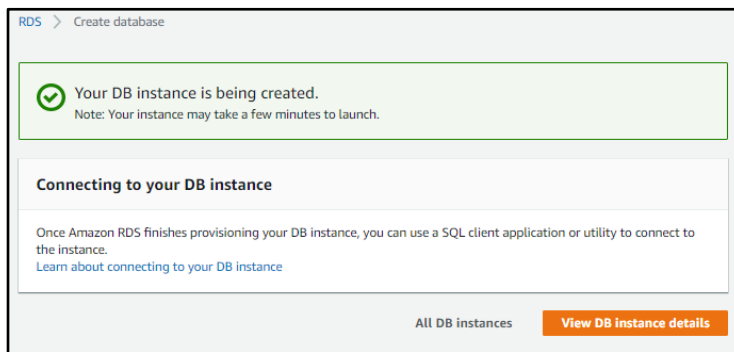
MyDMZ (sg-05de99cee3394ff40) (vpc-275fc440)
default (sg-18391161) (vpc-275fc440)
launch-wizard-1 (sg-041f91bb2f1f7dd5f) (vpc-275fc440)
rds-launch-wizard (sg-009c1506589d0b422) (vpc-275fc440)

EC2 instances and devices outside of the VPC
EC2 instances and devices outside of the VPC hosting the DB instance will connect to the DB instances. You must also select one or more VPC security groups that specify which EC2 instances and devices can connect to the DB instance.
☒ No
DB instance will not have a public IP address assigned. No EC2 instance or devices outside of the VPC will be able to connect.

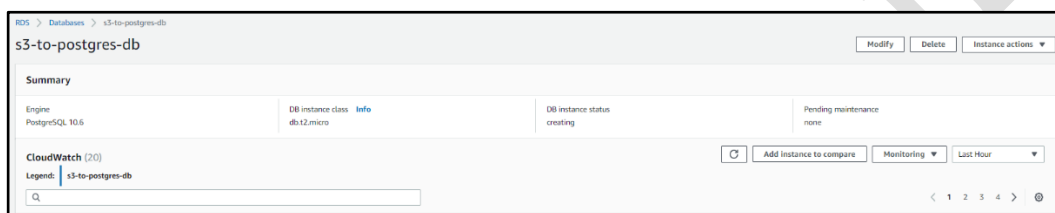
- 7) Leave below sections to default values and Create database.

- Encryption
- Backup
- Monitoring
- Performance Insights

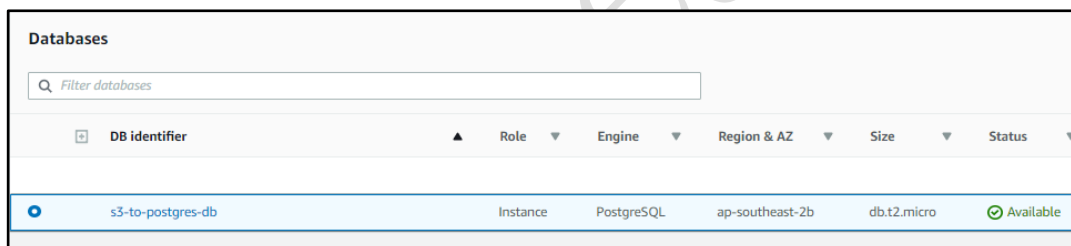
- Log exports
- Maintenance
- Deletion protection



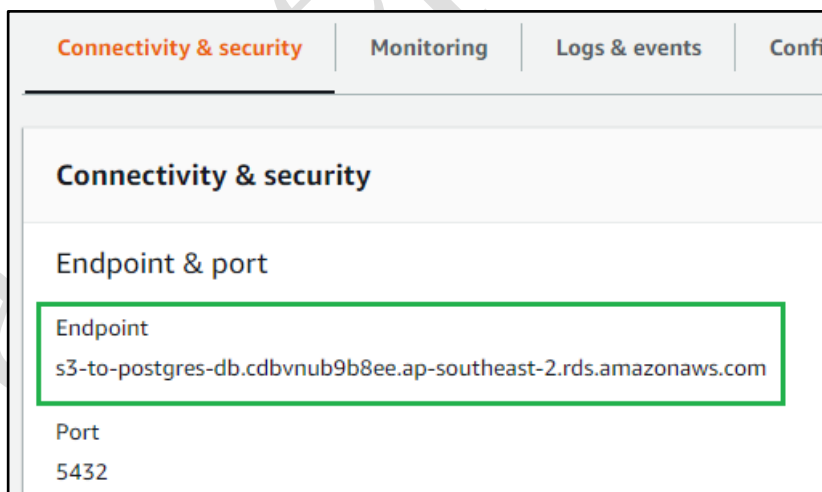
- 8) It takes few minutes to create the database and shows status as **creating**



- 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.cdbvnu9b8ee.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.cdbvnu9b8ee.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.cdbvnu9b8ee.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.cdbvnu9b8ee.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 x +
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.

```

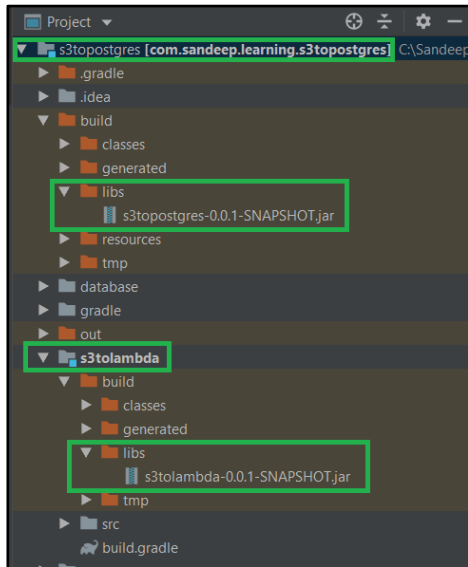
Terminal: Local x +
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

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

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

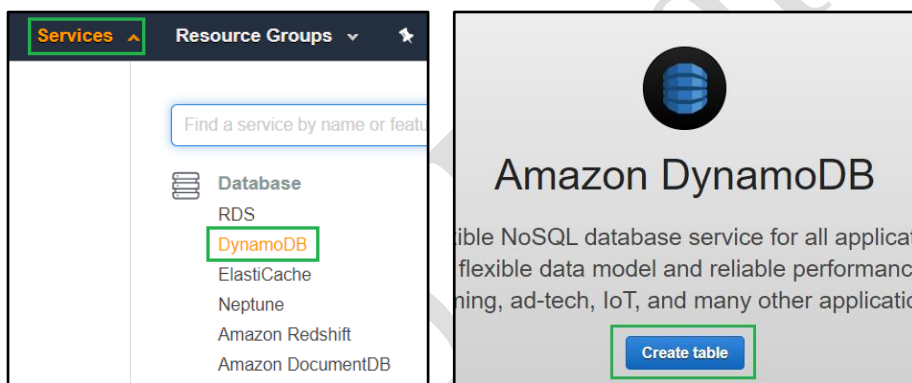
```

4) Check if jars are built.



11. Lambda configuration in Dynamo DB

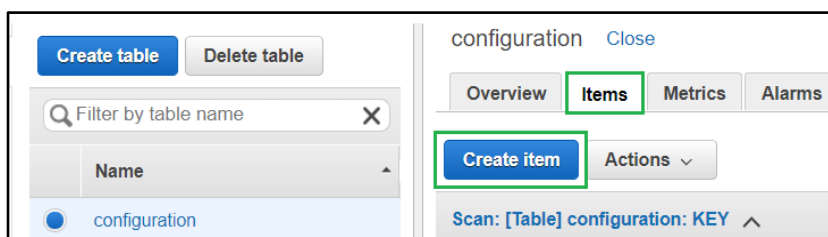
- 1) Go to Services → Database → DynamoDB, which allows to Create table

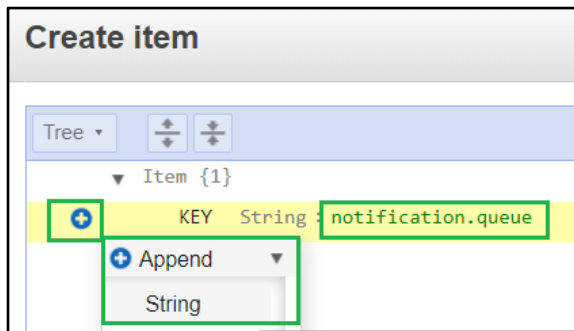


- 2) Provide the Table Name and Primary Key Column with Data Type.

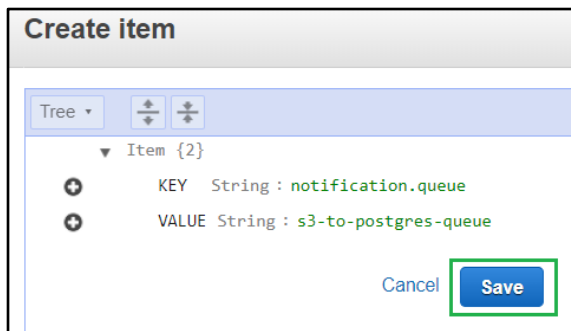
The screenshot shows the 'Create DynamoDB table' form. The 'Table name*' field is filled with 'configuration'. The 'Primary key*' section shows 'Partition key' selected, with 'KEY' entered in the text box and 'String' selected as the data type. The 'Add sort key' checkbox is unchecked. The 'Create' button is visible.

- 3) Table is created, now create item.



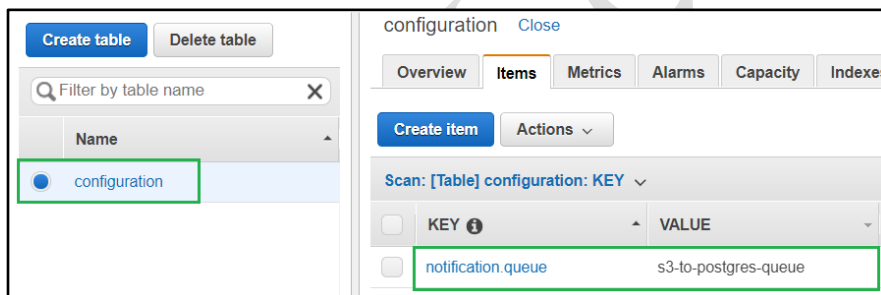


- 4) Once item is created, Save it.



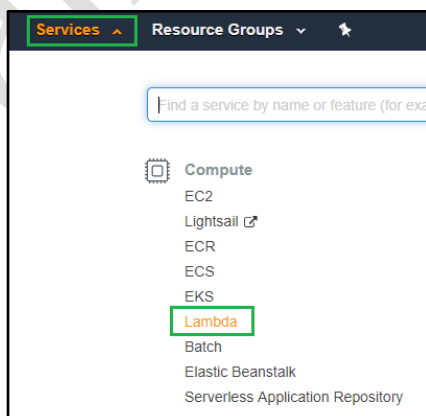
- 5) Upon saving, it would look like below.

KEY: notification.queue



12. Deploying Lambda

- 1) Go to Services → Compute → Lambda



- 2) Click create function.

The screenshot shows the AWS Lambda console interface for the Asia Pacific (Sydney) region. On the left is a navigation menu with 'Dashboard', 'Applications', 'Functions', and 'Layers'. The main content area is titled 'Resources for Asia Pacific (Sydney)' and displays 'Lambda function(s): 2' and 'Full account concurrency: 1000'. A green box highlights the 'Create function' button at the bottom.

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

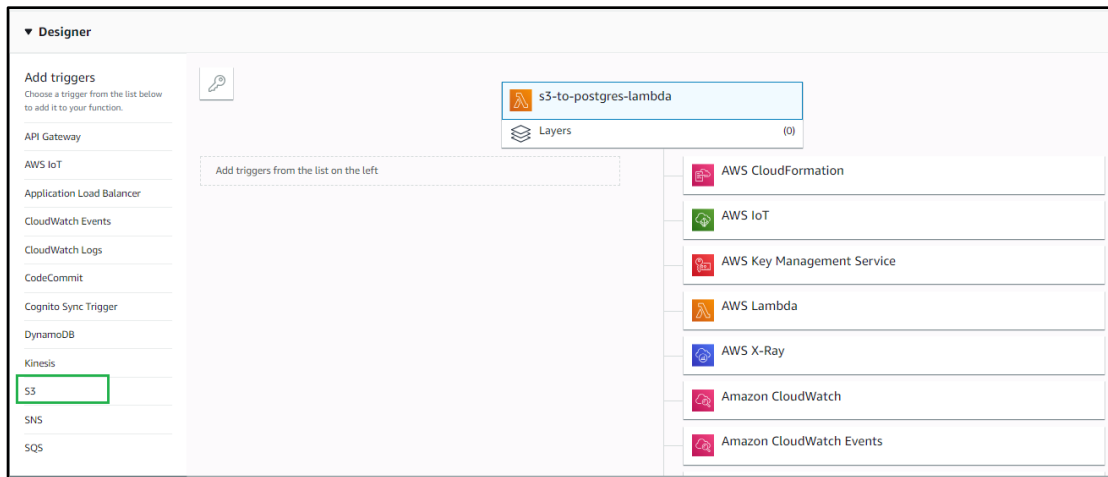
The screenshot shows the 'Basic information' form for creating a new Lambda function. The 'Function name' field contains 's3-to-postgres-lambda'. The 'Runtime' dropdown is set to 'Java 8'. Under the 'Permissions' section, the 'Choose or create an execution role' dropdown is set to 'Use an existing role', and the 'Existing role' dropdown is set to 'LambdaAccess'. A green box highlights the 'Create function' button at the bottom right.

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

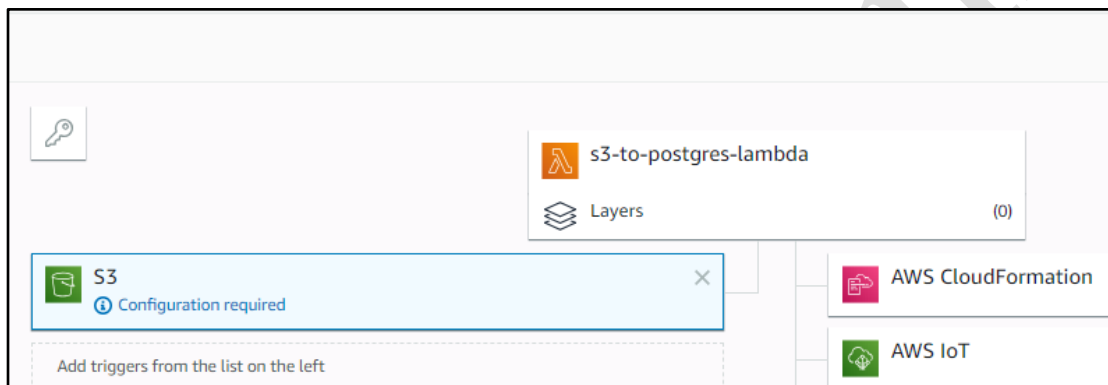
The screenshot shows the 'Summary' page for the 'LambdaAccess' role in the AWS IAM console. It displays the role's ARN, description, and creation time. Under the 'Permissions' tab, it shows two policies attached: 'AWSLambdaFullAccess' (AWS managed policy) and 'oneClick_lambda_basic_execution_155013070' (inline policy). The JSON content of the inline policy is visible at the bottom.

```
1- {
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Effect": "Allow",
6-       "Action": [
7-         "logs:CreateLogGroup",
8-         "logs:CreateLogStream",
9-         "logs:PutLogEvents"
10-      ],
11-       "Resource": "arn:aws:logs:*:*:*"
12-     }
13-   ]
14- }
```

- 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.

Configure triggers

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the Lambda function.

Event type
Select the events that you want to have trigger the Lambda function. You can optionally select multiple event types.

Prefix
Enter a single optional prefix to limit the notifications to objects with keys that start with this prefix. For example, 'images/'

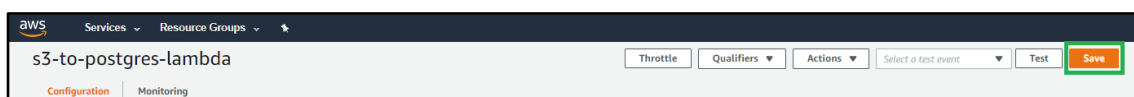
Suffix
Enter a single optional suffix to limit the notifications to objects with keys that end with this suffix. For example, '.csv'

Lambda will add the necessary permissions for Amazon S3 to invoke your function.

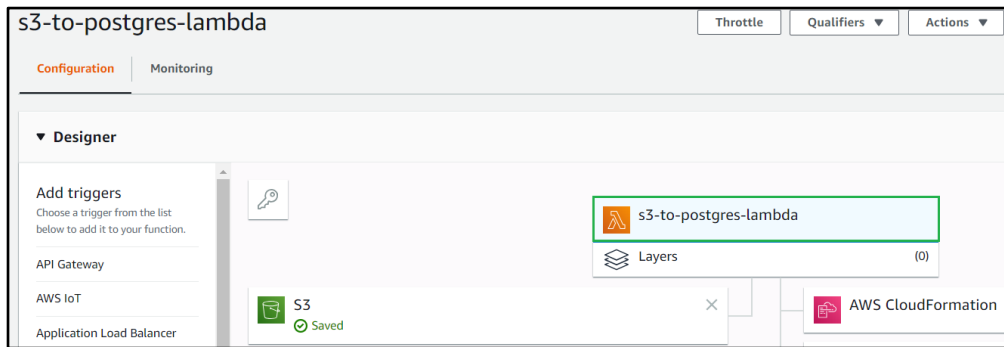
☒ **Enable trigger**
Enable the trigger now, or create it in a disabled state for testing (recommended).

Cancel Add

- 8) Click save button on the top.



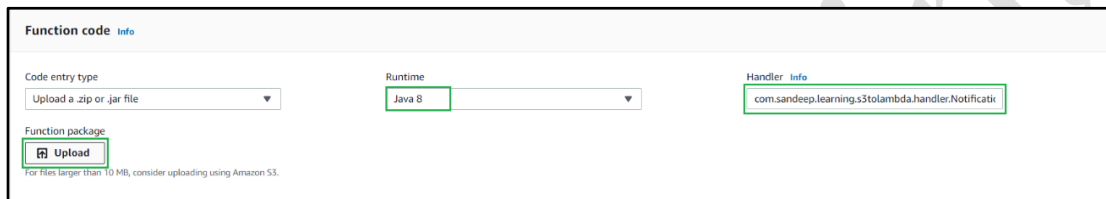
- 9) Now select the lambda as shown below.



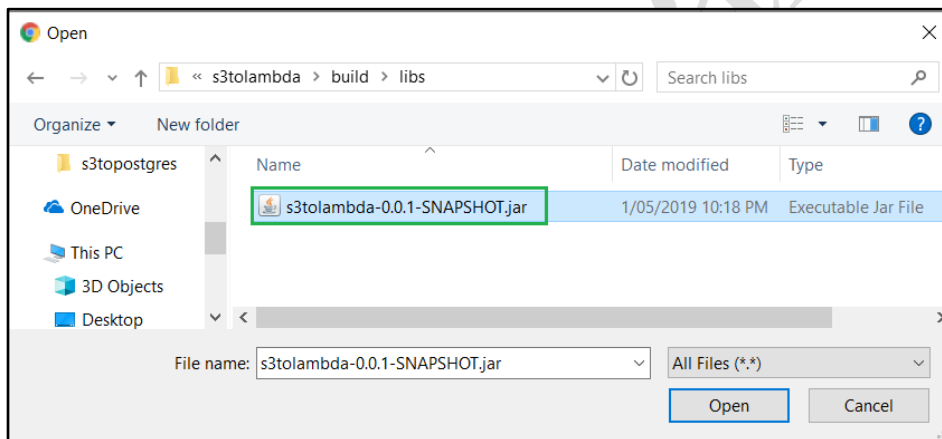
10) In the function code section, select the below shown configuration and click on upload.

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.

```

┌───┴───┐
┌─┴─┐  ┌─┴─┐
└─┴─┘  └─┴─┘  Amazon Linux AMI

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-8-85 ~]$ ls
[ec2-user@ip-172-31-8-85 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-8-85 ~]$ mkdir s3-to-postgres
[ec2-user@ip-172-31-8-85 ~]$ ls
s3-to-postgres
[ec2-user@ip-172-31-8-85 ~]$

```

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.cdbvub9b8ee.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-172-31-8-85
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 default
```

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.cdbvnu9b8ee.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.

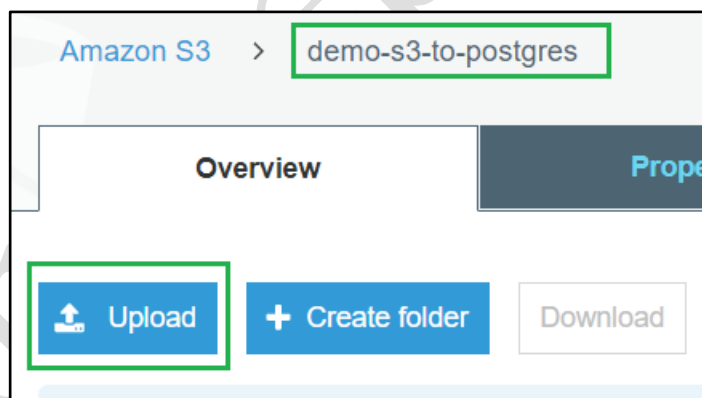
demodb=>
```

- 2) Check if the table has any records.

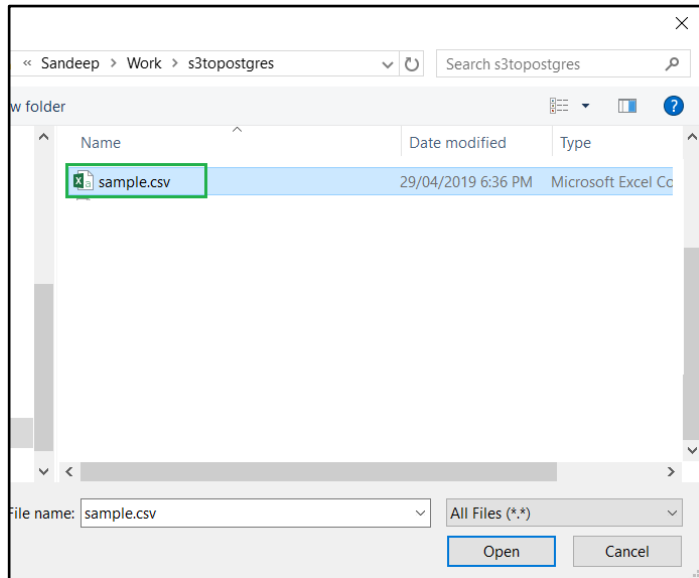
```
demodb=> select * from employees;
 employee_id | first_name | last_name | title | email
-----+-----+-----+-----+-----
(0 rows)

demodb=>
```

- 3) Now Go to S3 bucket and click upload

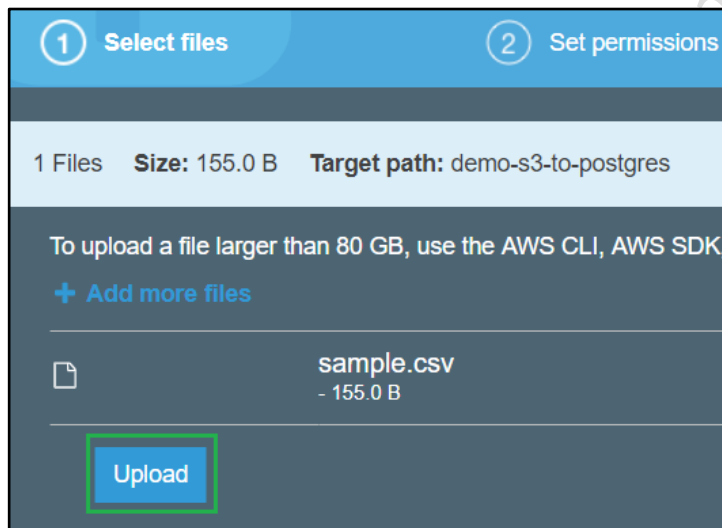


- 4) Upload sample.csv

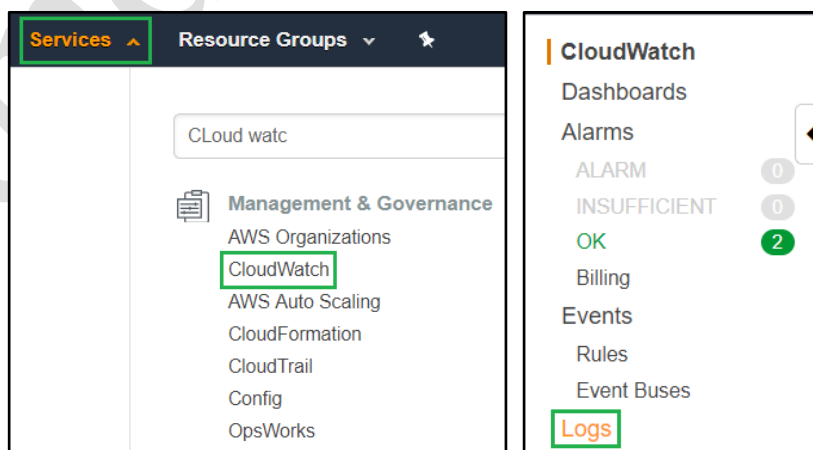


Note: sample data (sample.csv) is available in the resources folder of the project.

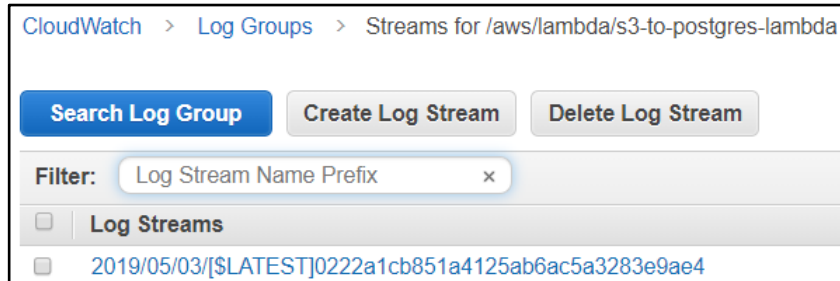
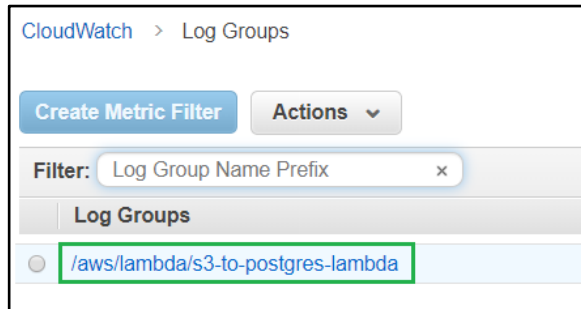
- 5) Click upload.



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



- 7) Click on generated logs



- 8) Click on this to see the Lambda logs.

```
START RequestId: ab342f25-3615-425e-881b-dc5c0f529076 Version: $LATEST
Sending message to queue : s3-to-postgres-queue
Bucket name is : demo-s3-to-postgres
File name is : sample.csv
Sending message to queueUrl : https://sqs.ap-southeast-2.amazonaws.com/ /s3-to-postgres-queue
Message sent to SQS queue is : {"bucketName":"demo-s3-to-postgres","fileName":"sample.csv"}
END RequestId: ab342f25-3615-425e-881b-dc5c0f529076
REPORT RequestId: ab342f25-3615-425e-881b-dc5c0f529076 Duration: 3920.85 ms Billed Duration: 4000 ms Memory Size: 512 MB Max Memory Used: 136 MB
START RequestId: dc5a710a-358e-4d12-b032-fdfb839572e8 Version: $LATEST
Sending message to queue : s3-to-postgres-queue
Bucket name is : demo-s3-to-postgres
File name is : sample.csv
Sending message to queueUrl : https://sqs.ap-southeast-2.amazonaws.com/ /s3-to-postgres-queue
Message sent to SQS queue is : {"bucketName":"demo-s3-to-postgres","fileName":"sample.csv"}
END RequestId: dc5a710a-358e-4d12-b032-fdfb839572e8
REPORT RequestId: dc5a710a-358e-4d12-b032-fdfb839572e8 Duration: 327.67 ms Billed Duration: 400 ms Memory Size: 512 MB Max Memory Used: 136 MB
```

- 9) Now check the logs on EC2.

```
Tomcat initialized with port(s): 7070 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/9.0.17]
Initializing Spring embedded WebApplicationContext
Initializing ExecutorService 'applicationTaskExecutor'
Initializing ExecutorService
Initializing ExecutorService 'taskScheduler'
Adding {logging-channel-adapter: org.springframework.integration.errorLogger} as a sub
Channel 's3-to-postgres.errorChannel' has 1 subscriber(s).
started org.springframework.integration.errorLogger
Tomcat started on port(s): 7070 (http) with context path '/s3-to-postgres'
Started S3topostgresApplication in 10.355 seconds (JVM running for 11.339)
Received notification is {"bucketName":"demo-s3-to-postgres","fileName":"sample.csv"}
Inserted records count 3
```

- 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
```