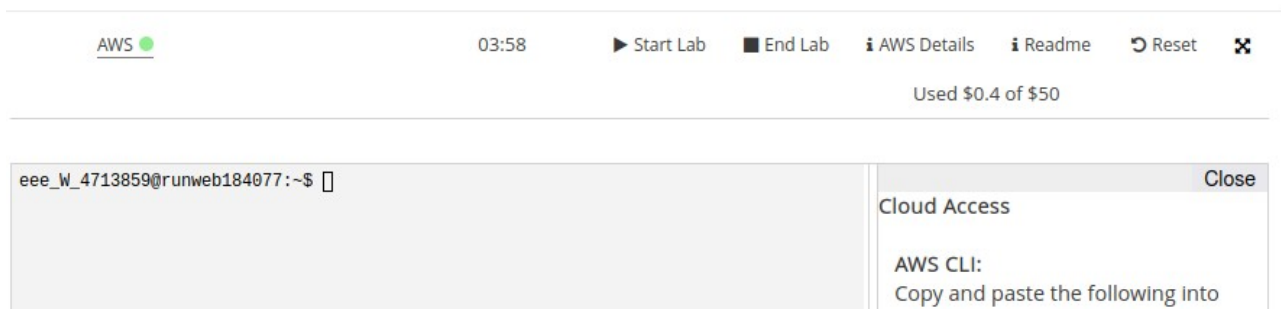


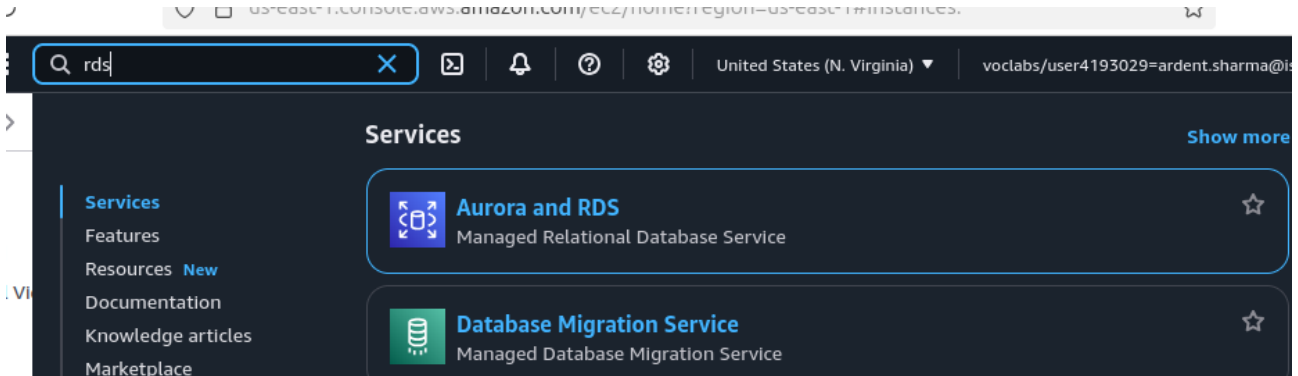
Open the module AWS learner lab and click start lab:



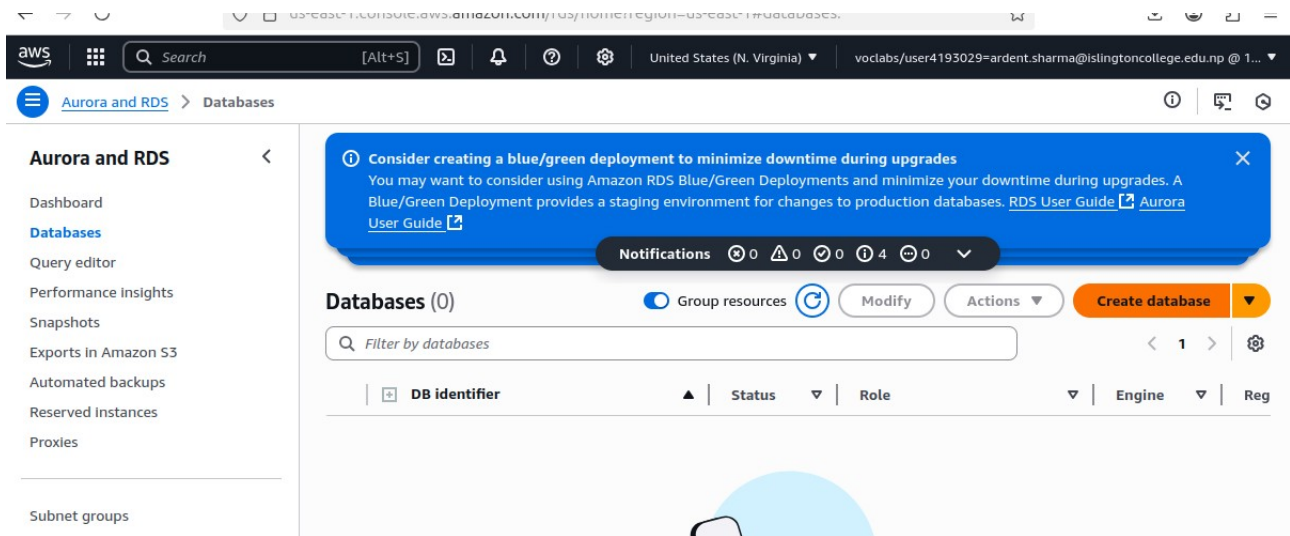
When the AWS has a green light right to it, that means our lab is ready.

If you go to ec2 and view the instances you will see that all 3 of your instances have already been started for you. If you are low on credits you should turn these off if you have other things to do. As they incur charges. These resources are not free of cost.

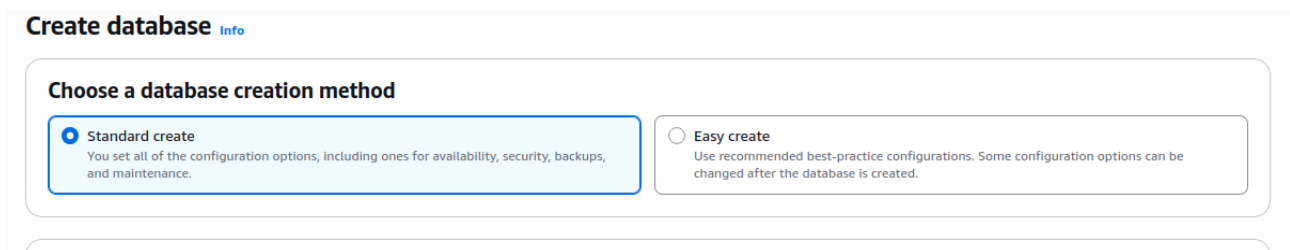
Search for rds in the search bar and click on Aurora and RDS service



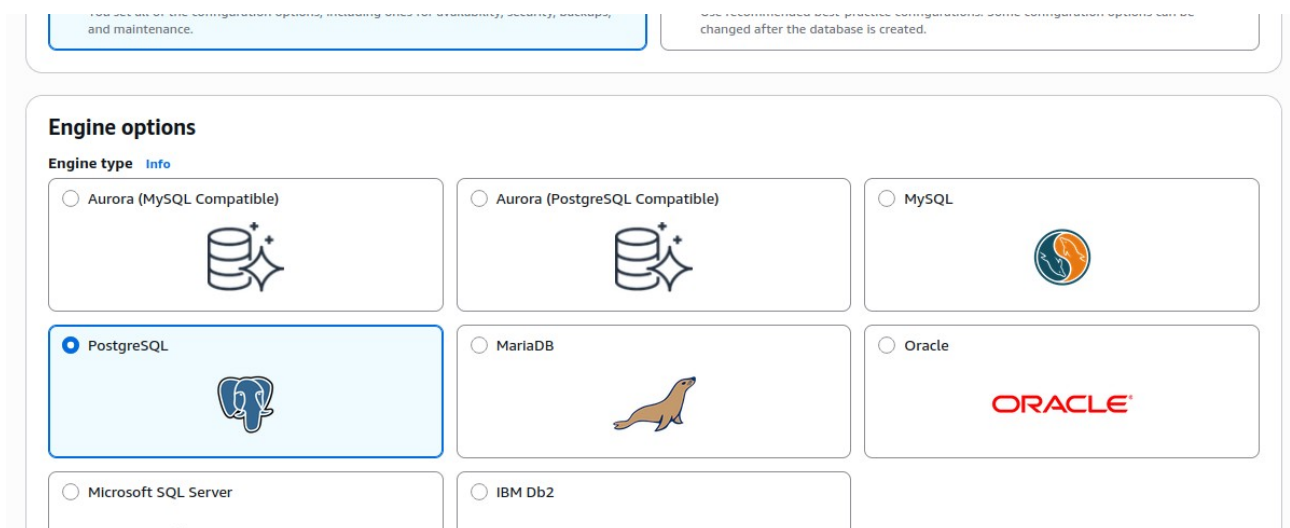
Go to databases section from left hand side and click on Create database button



Choose standard create



Then choose postgresql for engine options



Leave the engine option to default and select Dev/Test for the template. We are not using Production as it might incur some extra charges. The setup should be same for all 3 options.

Engine version [Info](#)
View the engine versions that support the following database features.

▼ Hide filters

☒ **Show only versions that support the Multi-AZ DB cluster** [Info](#)
Create a Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

Engine version
PostgreSQL 17.4-R1

☐ **Enable RDS Extended Support** [Info](#)
Amazon RDS Extended Support is a [paid offering](#). By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the [RDS for PostgreSQL documentation](#).

Templates
Choose a sample template to meet your use case.

☐ **Production**
Use defaults for high availability and fast, consistent performance.

☒ **Dev/Test**
This instance is intended for development use outside of a production environment.

☐ **Sandbox**
To develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.

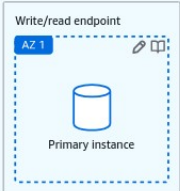
We do not care about availability and durability at this moment so we will select the first single Db instance deployment option. For production, other options will be a better choice based on the application type and expected load

Availability and durability

Deployment options [Info](#)
Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the [Amazon RDS service level agreement \(SLA\)](#).

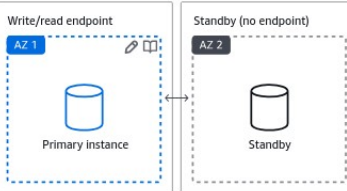
☒ **Single-AZ DB Instance deployment (1 Instance)**
Creates a single DB instance without standby instances. This setup provides:

- 99.9% uptime
- No data redundancy



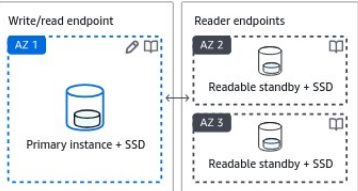
☐ **Multi-AZ DB Instance deployment (2 Instances)**
Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:

- 99.95% uptime
- Redundancy across Availability Zones



☐ **Multi-AZ DB cluster deployment (3 Instances)**
Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:

- 99.95% uptime
- Redundancy across Availability Zones
- Increased read capacity
- Reduced write latency



Settings

Give a name for your db instance. It can be whatever you want just has to be unique

Settings

DB instance identifier [Info](#)
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

spotify

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Give an username, make sure you select self-managed and give a password. Make sure you remember both of these as this is what we will need to access the database server

▼ Credentials Settings

Master username [Info](#)

Type a login ID for the master user of your DB instance.

postgres

1 to 16 alphanumeric characters. The first character must be a letter.

Credentials management

You can use AWS Secrets Manager or manage your master user credentials.

☐ Managed in AWS Secrets Manager - *most secure*

RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

☒ Self managed

Create your own password or have RDS create a password that you manage.

☐ Auto generate password

Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Password strength **Weak**

Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' * @

Confirm master password [Info](#)

For instance classes select the default option i.e., Standard classes and db.m7g.large 2 cpu and 8 gb ram. This should suffice for our use case. Note that this is the ec2 instance we are choosing to setup our postgres in. We could very well do this ourselves but instead we are using managed service provided by AWS

Confirm master password [Info](#)

Instance configuration

The DB Instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class [Info](#)

▼ Hide filters

☐ Include previous generation classes

☒ Standard classes (includes m classes)

☐ Memory optimized classes (includes r and x classes)

☐ Burstable classes (includes t classes)

db.m7g.large

2 vCPUs 8 GiB RAM Network: Up to 10,000 Mbps

Storage

Decrease the allocated Storage to 20gb as we will not need anything more than that. Leave everything else as default

Storage

Storage type [Info](#)
Provisioned IOPS SSD (io2) storage volumes are now available.

General Purpose SSD (gp3)
Performance scales independently from storage

Allocated storage [Info](#)
20 GIB
Minimum: 20 GIB, Maximum: 65,536 GIB

Provisioned IOPS [Info](#)
3000 IOPS
Baseline IOPS of 3,000 IOPS is included for allocated storage less than 400 GIB.

Storage throughput [Info](#)
125 MiBps
Baseline storage throughput of 125 MiBps is included for allocated storage less than 400 GIB.

[To provision additional IOPS and throughput, increase the allocated storage to 400 GIB or greater](#)

Leave everything else to default.

Modify the public access to yes so that we can access the database from anywhere. Again this is not a good idea for production environment. At this point you should already have this instinct. Chosse existing VPC security groups and select the one we created for ec2 instances.

Public access [Info](#)

☒ **Yes**
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

☐ **No**
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) [Info](#)
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

☒ **Choose existing**
Choose existing VPC security groups

☐ **Create new**
Create new VPC security group

Existing VPC security groups
Choose one or more options

☒ launch-wizard-1

☐ default

☐ no preference

RDS Proxy
RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

☐ **Create an RDS Proxy** [Info](#)

Leave most of the things to default. Untick the insights, and monitoring related stuffs.

☐ Database Insights - Advanced

- Retains 15 months of performance history
- Fleet-level monitoring
- Integration with CloudWatch Application Signals

☒ Database Insights - Standard

- Retains 7 days of performance history, with the option to pay for the retention of up to 24 months of performance history

Performance Insights
☐ Enable Performance Insights

With Performance Insights dashboard, you can visualize the database load on your Amazon RDS DB instance load and filter the load by waits, SQL statements, hosts, or users.

▼ Additional monitoring settings

Enhanced Monitoring, CloudWatch Logs and DevOps Guru

Enhanced Monitoring
☐ Enable Enhanced monitoring

Enabling Enhanced Monitoring metrics are useful when you want to see how different processes or threads use the CPU.

Log exports

Select the log types to publish to Amazon CloudWatch Logs

☐ iam-db-auth-error log
☐ PostgreSQL log
☐ Upgrade log

IAM role

The following service-linked role is used for publishing logs to CloudWatch Logs.

In additional configuration give name to database and disable automated backups (this will change depending on use case but enabling backup will incur extra charges).

▼ Additional configuration

Database options, encryption turned off, backup turned off, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

Database options

Initial database name [Info](#)

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

DB parameter group [Info](#)

Option group [Info](#)

Backup
☐ Enable automated backups

Creates a point-in-time snapshot of your database

☐ Enable encryption

Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. [Info](#)

Maintenance

Auto-minor-version-upgrade [Info](#)

Scroll down and select create database button

Monitoring role for OS metrics

default

Clicking "Create database" will authorize RDS to create the IAM role rds-monitoring-role

Log exports

Select the log types to publish to Amazon CloudWatch Logs

☐ iam-db-auth-error log
 ☐ PostgreSQL log
 ☐ Upgrade log

IAM role

The following service-linked role is used for publishing logs to CloudWatch Logs.

RDS service-linked role

► Additional configuration

Database options, encryption turned on, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

ⓘ You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

Cancel

Create database

You should get the below error. Apparently this lab does not allow creation of RDS databases. We need to configure our database manually. If it worked for you, do not follow below steps as you would already have a database setup.

☐ Choose a window
 ☒ No preference

☐ Enable deletion protection

Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

⊘ Access denied to rds:CreateDBInstance

Diagnose with Amazon Q

×

You don't have permission to `rds:CreateDBInstance`. To request access, copy the following text and send it to your AWS administrator. [Learn more about troubleshooting access denied errors.](#)

User: arn:aws:sts::193711398728:assumed-role/voclabs/user4193029=ardent.sharma@islingtoncollege.edu.np

Action: rds:CreateDBInstance

On resource(s): arn:aws:rds:us-east-1:193711398728:db:spotify

Context: no Identity-based policy allows the action

ⓘ You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

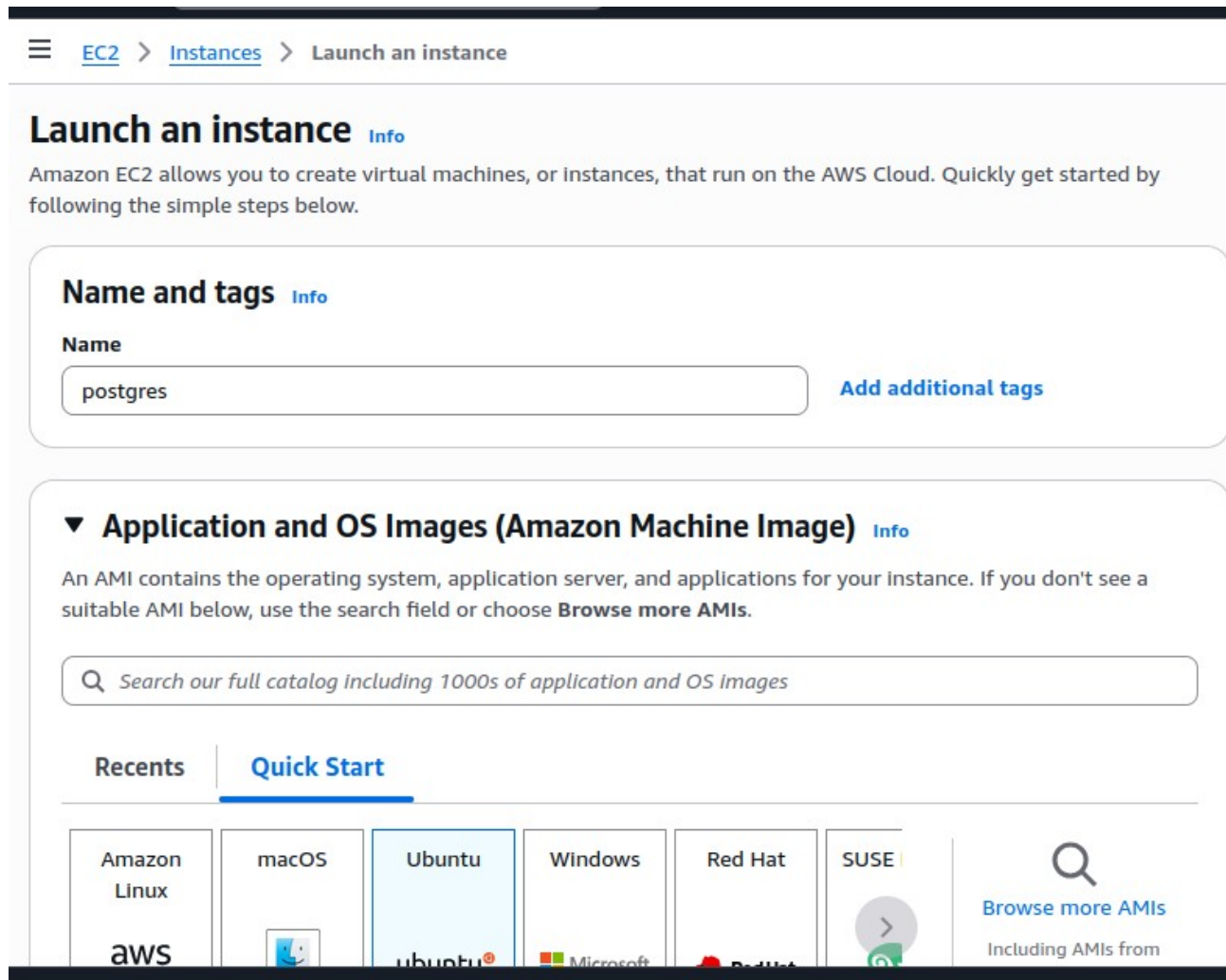
Cancel

Create database

Let us create new ec2 instance and setup postgresql there.

Go to Ec2>instances>launch an instances

Give the instance a name. I have given it postgres. And choose Ubuntu image as we did for our spark instances. Almost everything will be the same except for the instance type.



The screenshot shows the 'Launch an instance' page in the AWS Management Console. The breadcrumb navigation at the top reads 'EC2 > Instances > Launch an instance'. The main heading is 'Launch an instance' with an 'Info' link. Below this is a descriptive paragraph: 'Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.'

The first section is 'Name and tags' with an 'Info' link. It contains a 'Name' label and a text input field with the value 'postgres'. To the right of the input field is a button labeled 'Add additional tags'.

The second section is 'Application and OS Images (Amazon Machine Image)' with a dropdown arrow and an 'Info' link. It contains a paragraph: 'An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.'

Below the paragraph is a search bar with a magnifying glass icon and the placeholder text 'Search our full catalog including 1000s of application and OS images'.

Under the search bar are two tabs: 'Recents' and 'Quick Start'. The 'Quick Start' tab is selected and underlined.

Below the tabs is a row of AMI cards. The cards are: 'Amazon Linux' (with 'aws' logo), 'macOS' (with Apple logo), 'Ubuntu' (with 'ubuntu' logo), 'Windows' (with 'Microsoft' logo), 'Red Hat' (with 'Red Hat' logo), and 'SUSE' (with 'SUSE' logo). The 'Ubuntu' card is highlighted with a blue border. To the right of the cards is a magnifying glass icon and the text 'Browse more AMIs' and 'Including AMIs from'.

For instance type select m5.large

▼ Instance type [Info](#) | [Get advice](#)

Instance type

m5.large

Family: m5 2 vCPU 8 GiB Memory Current generation: true

On-Demand SUSE base pricing: 0.152 USD per Hour

On-Demand Windows base pricing: 0.188 USD per Hour

On-Demand Ubuntu Pro base pricing: 0.1 USD per Hour

On-Demand RHEL base pricing: 0.125 USD per Hour

On-Demand Linux base pricing: 0.096 USD per Hour

☒ All generations

[Compare instance types](#)

[Additional costs apply for AMIs with pre-installed software](#)

For key pair select our previous key pair. This is for ease so that we do not have maintain multiple keys. As always not recommended for production

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

pyspark



[Create new key pair](#)

For networking and security selecting existing group as well

▼ Network settings Info

Edit

Network Info

vpc-0e922083198bce206

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☐ Create security group

☒ Select existing security group

Common security groups Info

Select security groups ▼

launch-wizard-1 sg-08f3a5c71bd5dae6a X
VPC: vpc-0e922083198bce206

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

For storage select 20gb

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▼ Configure storage Info

Advanced

1x 20 GIB gp3 Root volume, 3000 IOPS, Not encrypted

Add new volume

The selected AMI contains instance store volumes, however the instance does not allow any instance store

Then in the summary section select Launch Instance

▼ Summary

Number of instances

Info

1

^

v

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...[read more](#)

ami-020cba7c55df1f615

Virtual server type (instance type)

m5.large

Firewall (security group)


launch-wizard-1

Storage (volumes)

1 volume(s) - 20 GiB

Cancel

Launch instance

 [Preview code](#)

After the instance has been created copy its public ip and ssh into it

Instances > i-0284c248709ee280c

Instance summary for i-0284c248709ee280c (postgres) [Info](#)

Updated less than a minute ago

Instance ID
i-0284c248709ee280c

Public IPv4 address
3.89.38.154 | [open address](#)

Instance state
Running

IPv6 address
-

Private IP DNS name (IPv4 only)
ip-172-31-26-74.ec2.internal

Hostname type
IP name: ip-172-31-26-74.ec2.internal

Answer private resource DNS name
IPv4 (A)

Instance type
m5.large

Auto-assigned IP address
3.89.38.154 [Public IP]

VPC ID
vpc-0e922083198bce206

Public IPv4 address copied

Use the same key and command as before.

```

connection to 34.102.104.40 closed.
ardent@ardent:~/Desktop$ ssh -i pyspark.pem ubuntu@3.89.38.154
The authenticity of host '3.89.38.154 (3.89.38.154)' can't be established.
ED25519 key fingerprint is SHA256:eSpBB7a+yagW+DktbPTWZHcpQZNEapXnWc58V0coC6M.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.89.38.154' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sun Aug  3 06:45:27 UTC 2025

System load:  0.02           Temperature:   -273.1 C
Usage of /:   9.3% of 18.3GB  Processes:    118
Memory usage: 3%            Users logged in: 0

```

Do the initial update and upgrade as before

See "man sudo_root" for details.

```
ubuntu@ip-172-31-26-74:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
...
98 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-26-74:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
```

Now let us install postgresql in this instance/server/node. The process will be exactly same as we did for our local machine.

Run the command to install postgresql

```
ubuntu@ip-172-31-26-74:~$ sudo apt install postgresql
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm17t64 libpq5 libtypes-serialiser-
  postgresql-client-16 postgresql-client-common postgresql-common ssl-cert
Suggested packages:
  postgresql-doc postgresql-doc-16
The following NEW packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm17t64 libpq5 libtypes-serialiser-
  postgresql-16 postgresql-client-16 postgresql-client-common postgresql-common ssl-cert
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 43.6 MB of archives.
After this operation, 175 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Press Y when prompted

Change the password of postgres user as we did before

```
ubuntu@ip-172-31-26-74:~$ sudo -i -u postgres
postgres@ip-172-31-26-74:~$ psql
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))
Type "help" for help.

postgres=# alter user postgres with password 'postgres';
ALTER ROLE
postgres=# \q
postgres@ip-172-31-26-74:~$ exit
logout
ubuntu@ip-172-31-26-74:~$
```

We will need to make certain changes to our config file which will be different from our local setup.

Open the config file in text editor of your choice

```
logout
ubuntu@ip-172-31-26-74:~$ sudo vim /etc/postgresql/16/main/postgresql.conf
```

`sudo vim /etc/postgresql/16/main/postgresql.conf`

Replace vim with nano (or any other editor) if you are comfortable with it

```
# - Connection Settings -

#listen_addresses = 'localhost'          # what IP address(es) to listen on;
#                                           # comma-separated list of addresses;
#                                           # defaults to 'localhost'; use '*' for all
#                                           # (require 'listen' in the dynamic shared object)
```

Uncomment the listen_address parameter and replace 'localhost' with '*' as we want all ip's to be able to access this database

```
# - Connection Settings -

listen_addresses = '*'                  # what IP address(es) to listen on;
#                                           # comma-separated list of addresses;
#                                           # defaults to 'localhost'; use '*' for all
#                                           # (require 'listen' in the dynamic shared object)
```

Save and exit. Now it is time to update the pg_hba.conf file like we did earlier

```
ubuntu@ip-172-31-26-74:~$ sudo vim /etc/postgresql/16/main/postgresql.conf
ubuntu@ip-172-31-26-74:~$ sudo vim /etc/postgresql/16/main/pg_hba.conf
```

`sudo vim /etc/postgresql/16/main/pg_hba.conf`

Move down to the connection settings. You will find the below config

```
# Database administrative login by Unix domain socket
local all postgres peer

# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all all 127.0.0.1/32 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege
```

Change the peer and scram to md5. Also for host change the ip to 0.0.0.0/0 to support all ip's

```
# Database administrative login by Unix domain socket
local  all             postgres              md5

# TYPE  DATABASE  USER  ADDRESS  METHOD
# "local" is for Unix domain socket connections only
local  all             all              md5
# IPv4 local connections:
host   all             all             0.0.0.0/0  md5
# IPv6 local connections:
host   all             all             ::1/128    md5
# Allow replication connections from localhost, by a user with the
# replication privilege
```

Save and exit the file

Restart the postgresql service using below command.

```
ubuntu@ip-172-31-26-74:~$ sudo vim /etc/postgresql/16/main/pg_hba.conf
ubuntu@ip-172-31-26-74:~$ sudo systemctl restart postgresql
```

`sudo systemctl restart postgresql`

Let us try to connect from the system itself.

```
ubuntu@ip-172-31-26-74:~$ sudo systemctl restart postgresql
ubuntu@ip-172-31-26-74:~$ psql -U postgres
Password for user postgres:
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))
Type "help" for help.

postgres=#
```

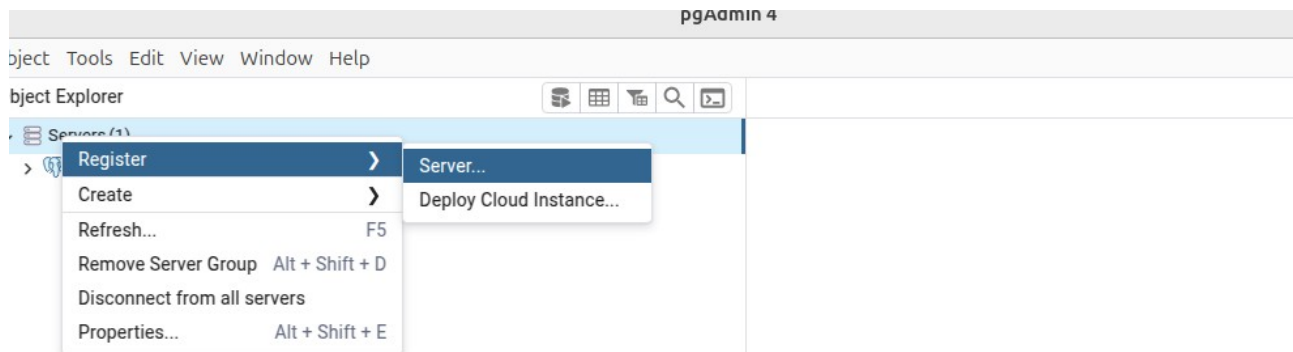
`psql -U postgres`

If it worked successfully let us try from our own/local system pgadmin. If it did not succeed debug on your own.

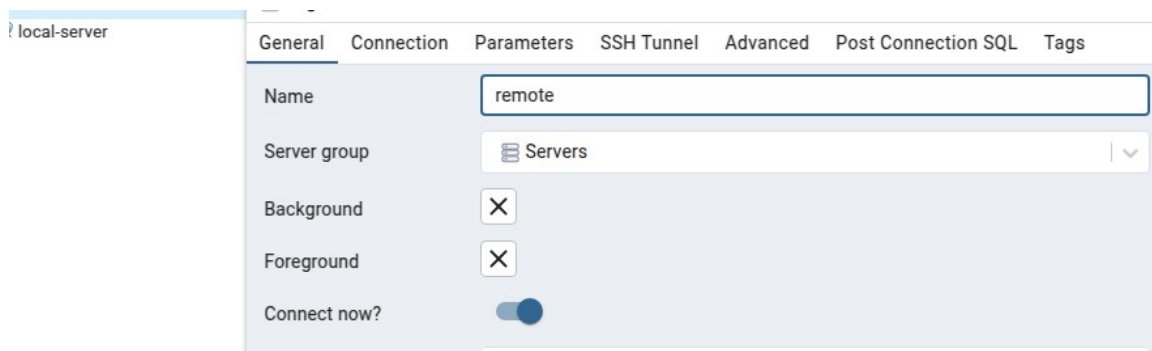
Open pgadmin on your local machine



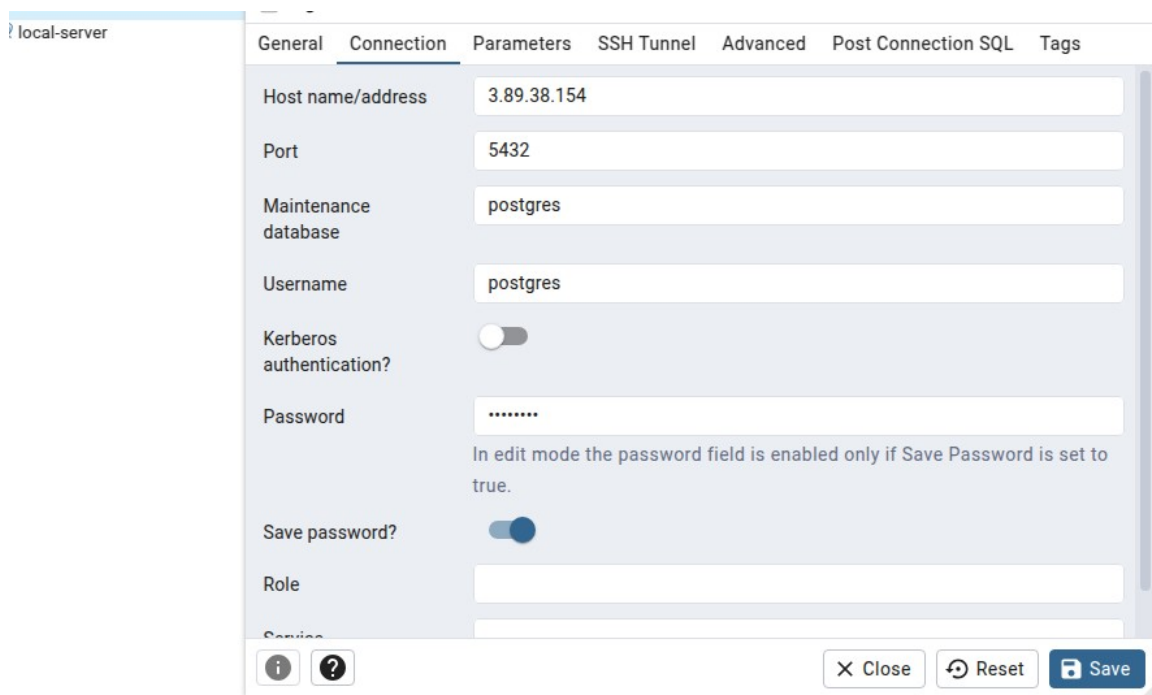
In the server right click and select register and then server



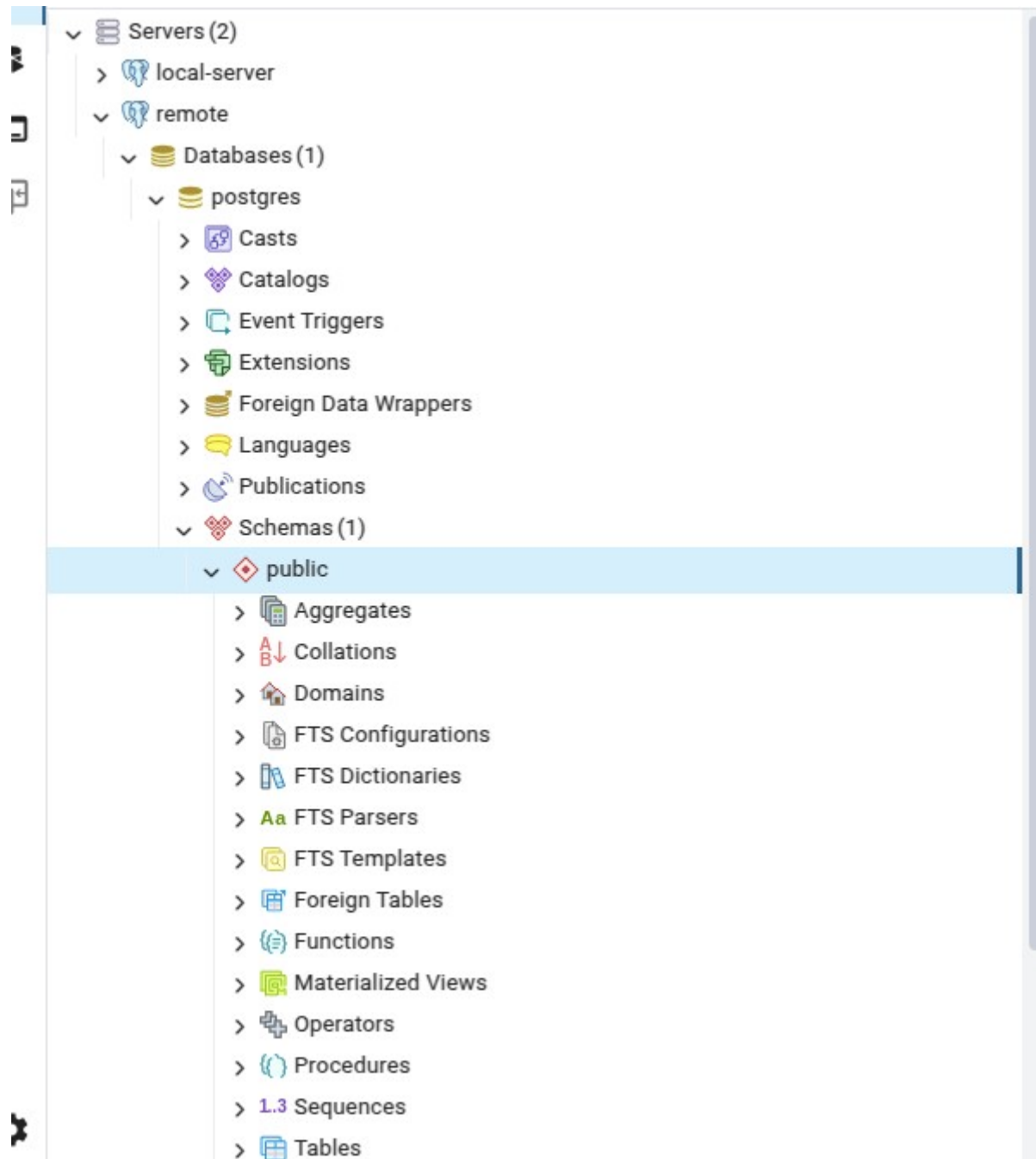
Give it a name. I have given remote to distinguish our local and remote ec2 instance based postgres



In the connection tab. For host provide the public ip of the instance. The port, database and username should be default if you did not make any changes yourself. For password entered the password you created and click save.



Remote should be added and now you should be able to view the database instance. Note that currently we do not have any table but we have verified that our database instance has been created and it can be connected from anywhere.



Go to AWS Details and copy the parameters available there.

AWS03:57Start LabEnd LabAWS DetailsReadmeResetUsed \$0.4 of \$50

eee_w_4713859@runweb184077:~\$

Cloud AccessClose

AWS CLI:
Copy and paste the following into ~/.aws/credentials

[default]
aws_access_key_id=ASIAS2GQZL5EBS4CFFLR
aws_secret_access_key=xIk+v5DBokfhX0T8CoDWILVpyFtUuAMbW
LpcFPw5
aws_session_token=IQoJb3JpZ2luX2VjE03////////wEaCXVzL
Xdlc3QtMiJIMEYCIQCd00VMry/Tviw3cwTzu177xPnirg0wqY6Bx7ii
mUPF0gIhAJbCn0Ch5TgAkn52unXQ79A2UC/RCwmvSJsQVuoHUU5JKr8
CCCYQABoMMTkzNzExMzk4NzI4IgzgJbap070BDkiY1mcqnAL37IT8c0
nqpuh10U9s9r4hv2XC8+Q5XniRqXFecFPKiHQECZbD5nnjyD6NHAGXx
qeKr0fSkwL20AJZ7T8HNbetK/XjUbdn20qaCMWTdGyKuqU7qkJ0H2fS
jhuiNn23p6x44MY00NeJ0+/T++1wZGq/62ZIpbv3w17Br5rrsjzYQ2
+ivE99wSyNruqlk7LYzyS379ijLVFj6N3yzEafHXrpSpXzJTgyu0EqC
0a0UeTWn0A09i2kEMZ+ANqrdkKLoXU/h6nZNNSxB0twcGih9xYyn59W
PDkvtJDAkaQMyCwmKHvriDNqQw2TTSbZBVBigoU16MDs0EB0fx2L2ly
51xJzICiL3+P2UIb292Gr2cc6NjNACqBuId+PB5/rJjCP27vEBjqcAc

This is not the case in production code. The access key id and secret key is usually same across different sessions. But since we are using learner lab we will have to get these key and setup in our .bashrc file every time we start our lab anew.

If you had stopped the spark based ec2 instances earlier then start them again. We will work on it now. Get the public ip of your spark based instances and ssh into your instance. You will have to do this for all 3 instances.

alarms +	us-east-1b	ec2-54-162-164-46.co...	54.162.164.46	-	-
alarms +	us-east-1d	ec2-54-205-31-65.com...	54.205.31.65	-	-
alarms +	us-east-1d	ec2-3-82-128-137.com...	3.82.128.137	-	-

```
ssh [-q query_option]
ardent@ardent:~/Desktop$ ssh -i pyspark.pem ubuntu@54.162.164.46
The authenticity of host '54.162.164.46 (54.162.164.46)' can't be established.
ED25519 key fingerprint is SHA256:bwa9r6liAHDGN9bjfyIF5pVskhIFJrPU0hfucHyLzhU.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:4: [hashed name]
  ~/.ssh/known_hosts:13: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.162.164.46' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.14.0-1010-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
```

Then update the 3 values of aws_keys in your .bashrc file

```
fi
export AWS_ACCESS_KEY_ID=ASIAS2GQZL5EBS4CFFLR
export AWS_SECRET_ACCESS_KEY=xIk+v5DBokfhXOT8CoDWILVpyFtUuAMbWlpcFPw5
export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjE03////////wEaCXVzLXdlc3QtMiJlMEYCIQcd00VMry/Tviw3cwTzu177xPnirg0wqY6Bx7iimUP
F0gIhAJbCn0Ch5TgAkn52unXQ79A2UC/RCwmvSJsQVuohUu5JKr8CCCYQABoMMTKzNzExMzk4NzI4IgzgJbap070BDkiY1mcqnAL37IT8c0nqpuh10U9s9r4
hv2XC8+Q5XniRqXFecFPKIHQECZbD5nnjyD6NHAGXxqeKr0fSkwL20AJZ7T8HNBetK/XjUbnd20qaCMWTDGyKuU7qkJOH2fSjhuiNn23pGx44MY00NeJO+/
T++1wZGq/62ZIpbv3w17Br5rrsjzYQ2+lvE99wSyNruqlk7lYzyS379ijlVFj6N3yzEafHXrpSpXzJTGyu0EqC0a0UeTwn0A09i2kEMZ+ANqrdkKLoXU/h6n
ZNNSxB0twcGih9xYyn59WPDkvtJDAkaQMyCwmKHvriDNQW2TTSbZBVBigoU16MDs0EB0fx2L2lyS1xJzICiL3+P2UIb292Gr2cc6NjNACqBuid+PBS/rJjC
P27vEBjqcAcBIrcpnUt/RJDT3R7T/yfbDpMgSMcULqtCB4owDB+GCXZHLtRGsQs0ybsNox0dqcCatJKbXCNBiagSiMsWR2KF91BzhctEwlgdZVgPx/RfSod/
2KBUIIY396/ooy8NEE6tU2mU7qfmamPE9FvgB0sl6frkJiMLG8otnjhkix0BK/1qXmbbK0pfocZhu4tFrYwL1ANg6BD06FJx7ag==
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
export PYSPARK_PYTHON=/usr/bin/python3

-- INSERT --
```

```
ubuntu@ip-172-31-40-25:~$ vim .bashrc
ubuntu@ip-172-31-40-25:~$ source .bashrc
ubuntu@ip-172-31-40-25:~$
```

Make sure you run source .bashrc as always to update the change

Do the above two steps for all 3 instances.

On the master node run the script to run master

```
Last login: Sun Aug 3 05:04:37 2025 from 110.44.115.197
ubuntu@ip-172-31-40-25:~$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark
-ubuntu-org.apache.spark.deploy.master.Master-1-ip-172-31-40-25.out
ubuntu@ip-172-31-40-25:~$
```

Verify that it is running by visiting the webui master_ip:8080

Spark Master at spark://172.31.40.25:7077

URL: spark://172.31.40.25:7077
 Alive Workers: 0
 Cores in use: 0 Total, 0 Used
 Memory in use: 0.0 B Total, 0.0 B Used
 Resources in use:
 Applications: 0 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

▼ Workers (0)

Worker Id	Address	State	Cores	Memory	Resources
-----------	---------	-------	-------	--------	-----------

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	Sta
----------------	------	-------	---------------------	------------------------	----------------	------	-----

▼ Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	Sta
----------------	------	-------	---------------------	------------------------	----------------	------	-----

Now let us add workers by going to ssh instance of slave and running start-worker script
 start-worker.sh spark://master_private_ip:7077

Private ip of master is always the same so this could be automatized later

```
Last login: Sat Aug 2 11:42:51 2025 from 110.44.115.197
ubuntu@ip-172-31-84-134:~$ start-worker.sh spark://172.31.40.25:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark
-ubuntu-org.apache.spark.deploy.worker.Worker-1-ip-172-31-84-134.out
ubuntu@ip-172-31-84-134:~$
```

Note: we use public ip to ssh because our local system is not in the same network as our instances.
 But these three instances are in the same network so they can communicate using the private ip as well as public ip

Run the above script it next worker/slave instance as well.

If you visit the webui after activating both the workers. It should be showing up



Spark Master at spark://172.31.40.25:7077

URL: spark://172.31.40.25:7077

Alive Workers: 2

Cores in use: 4 Total, 0 Used

Memory in use: 5.6 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory
worker-20250803072404-172.31.84.118-41573	172.31.84.118:41573	ALIVE	2 (0 Used)	2.8 GiB (0.0 B Used)
worker-20250803072446-172.31.84.134-32913	172.31.84.134:32913	ALIVE	2 (0 Used)	2.8 GiB (0.0 B Used)

Note in the above image we can also see the available cores and memory for that particular worker.

Now, that everything is setup let us run our load script. That reads from s3 and loads into our newly created postgresql database.

For this go to the ssh session of master instance.

If we cd into etl and try to run the load code we notice that we do not have pycpg2 installed in the master server. Lets install that first

```
ubuntu@ip-172-31-40-25:~$ cd etl
ubuntu@ip-172-31-40-25:~/etl$ python3 load/execute.py
Traceback (most recent call last):
  File "/home/ubuntu/etl/load/execute.py", line 4, in <module>
    import pycpg2
ModuleNotFoundError: No module named 'pycpg2'
ubuntu@ip-172-31-40-25:~/etl$ pip3 install --break-system-packages pycpg2-binary
Defaulting to user installation because normal site-packages is not writeable
Collecting pycpg2-binary
  Downloading pycpg2_binary-2.9.10-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 MB)
  Downloading pycpg2_binary-2.9.10-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.0 MB)
    3.0/3.0 MB 45.7 MB/s eta 0:00:00
Installing collected packages: pycpg2-binary
Successfully installed pycpg2-binary-2.9.10
ubuntu@ip-172-31-40-25:~/etl$
```

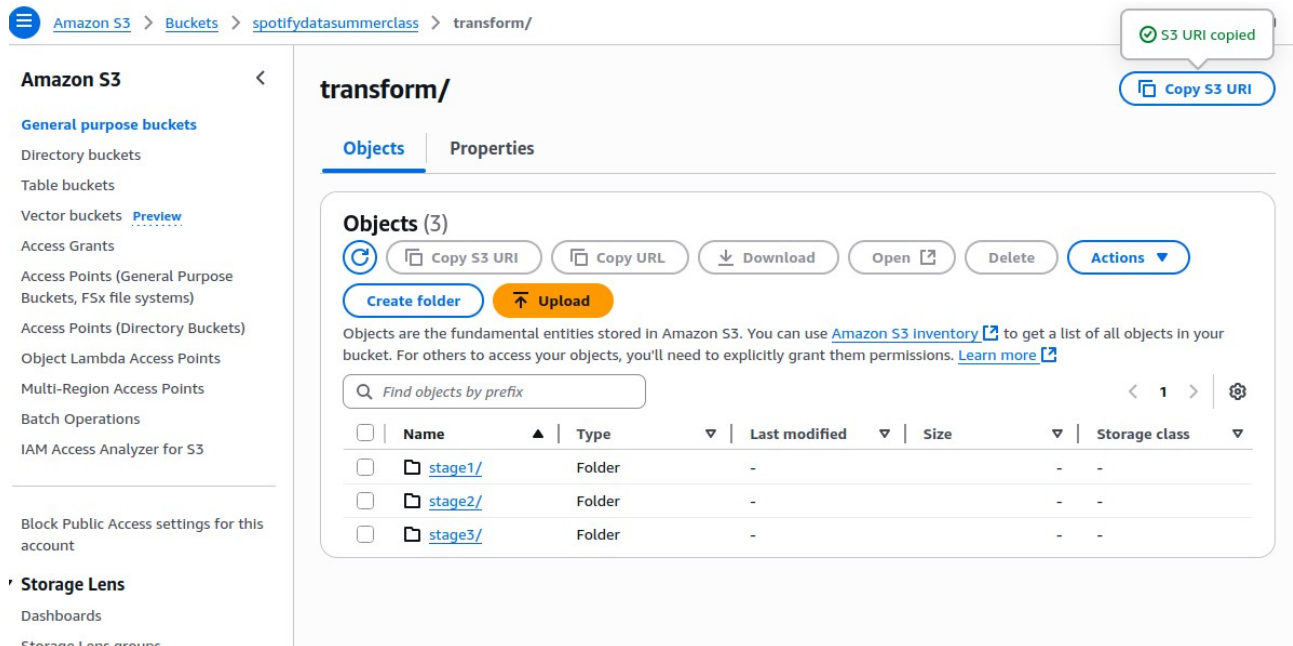
`pip3 install --break-system-packages pycpg2-binary`

Trying to run the code without any parameter throws an error as it should

```
ubuntu@ip-172-31-40-25:~/etl$ python3 load/execute.py
2025-08-03 07:36:03,845 [MainThread ] [ERROR] Usage: python load/execute.py <input_dir> <pg_un> <pg_pw> pg_host master
ip_d_mem e_mem e_core e_inst
```


We will be needing several inputs. `input_dir` in our case the `s3 uri` for our transform. The `pg_un` and `pg_pw` is the username and password we created earlier. `pg_host` is the public ip of the ec2 instance we created earlier and installed postgresql in. `master_ip` is the public ip of the master. Other parameters are driver memory, executor memory, executor core, and executor instances

Let us get our `s3 uri`



The screenshot shows the Amazon S3 console interface. The breadcrumb navigation at the top indicates the path: `Amazon S3 > Buckets > spotifydatasummerclass > transform/`. On the left sidebar, under 'Amazon S3', the 'General purpose buckets' section is expanded, showing various bucket types. The main content area displays the 'transform/' bucket with the 'Objects' tab selected. It shows three folders: 'stage1/', 'stage2/', and 'stage3/'. Above the object list, there are buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', and 'Actions'. A notification at the top right states 'S3 URI copied'. Below the object list, there is a search bar and a table with columns: Name, Type, Last modified, Size, and Storage class.

Name	Type	Last modified	Size	Storage class
stage1/	Folder	-	-	-
stage2/	Folder	-	-	-
stage3/	Folder	-	-	-

In my case it is: `s3://spotifydatasummerclass/transform/` I will add a to the uri as pyspark needs in that format

`s3a://spotifydatasummerclass/transform/`

The required public ips can be retrieved from instances page. After gathering required information run the code again by passing them as arguments.

```
ubuntu@ip-172-31-40-25:~/etl$ python3 load/execute.py s3a://spotifydatasummerclass/transform/ postgres postgres 3.89.38
154 54.162.164.46 2g 2700m 2 2
2025-08-03 07:44:29,691 [MainThread ] [INFO ] Load stage started
```

Now, the tables will be created but the data won't be transferred to postgres server using spark. If you have closely followed the manual from the beginning of the course till now you should have the idea why that is happening.


```

ubuntu@ip-172-31-40-25:~/etl$ python3 load/execute.py s3a://spotifydatasummerclass/transform/ postgres postgres 3.89.38.
154 54.162.164.46 2g 2700m 2 2
2025-08-03 07:53:24,470 [MainThread ] [INFO ] Load stage started
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/08/03 07:53:29 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
2025-08-03 07:53:32,866 [MainThread ] [INFO ] PostgreSQL tables created successfully
25/08/03 07:53:34 WARN MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-system.properties,hado
op-metrics2.properties
2025-08-03 07:53:47,128 [MainThread ] [WARNI] Error loading master_table: An error occurred while calling o39.jdbc.
: java.lang.ClassNotFoundException: org.postgresql.Driver
    at java.base/java.net.URLClassLoader.findClass(URLClassLoader.java:476)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:594)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:527)
    at org.apache.spark.sql.execution.datasources.jdbc.DriverRegistry$.register(DriverRegistry.scala:46)
    at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions.$anonfun$driverClass$1(JDBCOptions.scala:101)
    at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions.$anonfun$driverClass$1$adapted(JDBCOptions.scala:
101)
    at scala.Option.foreach(Option.scala:407)
    at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions.<init>(JDBCOptions.scala:101)
    at org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions.<init>(JDBCOptions.scala:230)

```

If you go and check in your pgadmin remote postgres server. You will find that tables have been created but there are no data in the tables. This is because we have not added the jar file for postgres. Lets do that before we run the load code again.

Change directory to where sparks jars is located. Then use wget to download the jar file and change the directory back to our etl

```

2025-08-03 07:53:53,280 [MainThread ] [INFO ] Closing down clientserver connection
ubuntu@ip-172-31-40-25:~/etl$ cd /opt/spark/jars
ubuntu@ip-172-31-40-25:/opt/spark/jars$ wget https://jdbc.postgresql.org/download/postgresql-42.7.7.jar
--2025-08-03 07:57:04-- https://jdbc.postgresql.org/download/postgresql-42.7.7.jar
Resolving jdbc.postgresql.org (jdbc.postgresql.org)... 72.32.157.228, 2001:4800:3e1:1::228
Connecting to jdbc.postgresql.org (jdbc.postgresql.org)|72.32.157.228|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1098916 (1.0M) [application/java-archive]
Saving to: 'postgresql-42.7.7.jar'

postgresql-42.7.7.jar      100%[=====] 1.05M  5.17MB/s  in 0.2s

2025-08-03 07:57:05 (5.17 MB/s) - 'postgresql-42.7.7.jar' saved [1098916/1098916]

ubuntu@ip-172-31-40-25:/opt/spark/jars$ cd ~/etl
ubuntu@ip-172-31-40-25:~/etl$

```

wget <https://jdbc.postgresql.org/download/postgresql-42.7.7.jar>

This needs to only be done in all 3 nodes

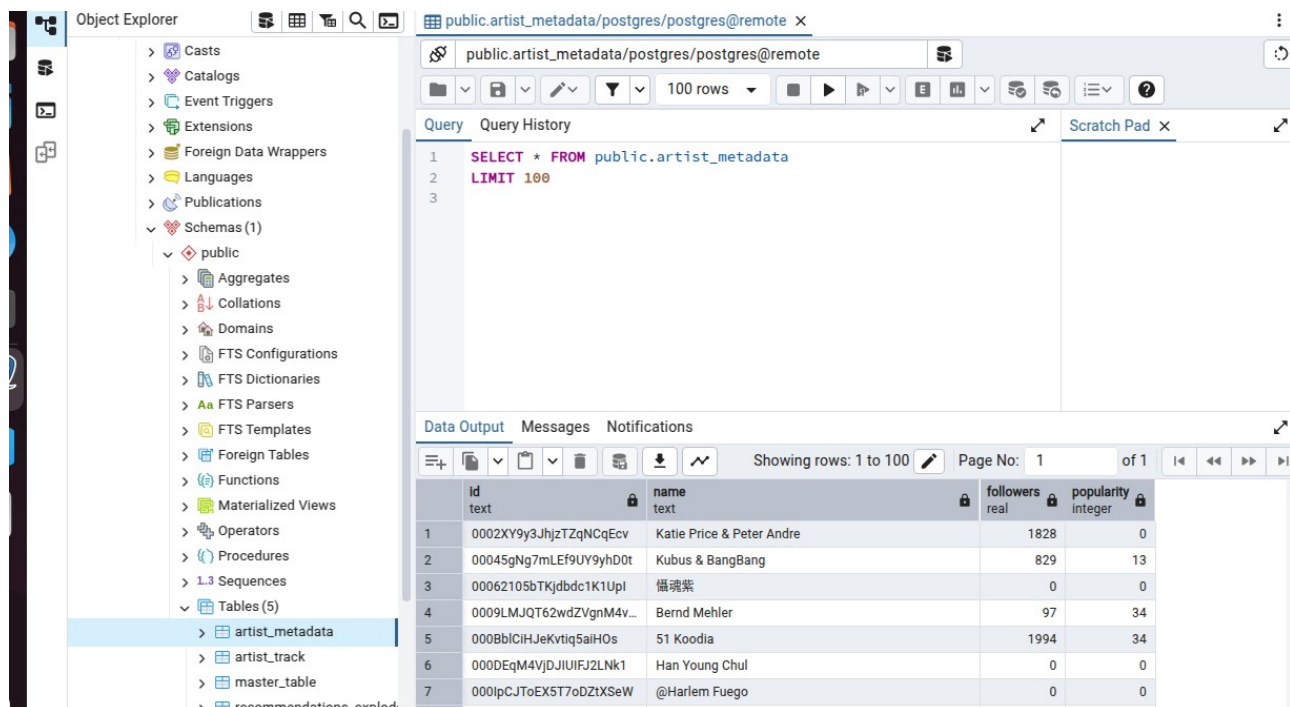
Now, let us try to run our spark code again from our master node

```

ubuntu@ip-172-31-40-25:~/etl$ python3 load/execute.py s3a://spotifydatasummerclass/transform/ postgres postgres 3.89.38.
154 54.162.164.46 2g 2700m 2 2
2025-08-03 08:09:34,023 [MainThread ] [INFO ] Load stage started
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/08/03 08:09:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
2025-08-03 08:09:42,095 [MainThread ] [INFO ] PostgreSQL tables created successfully
25/08/03 08:09:43 WARN MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-system.properties,hado
op-metrics2.properties
2025-08-03 08:10:15,530 [MainThread ] [INFO ] Loaded master_table to PostgreSQL
2025-08-03 08:11:09,776 [MainThread ] [INFO ] Loaded recommendations_explored to PostgreSQL
2025-08-03 08:11:15,015 [MainThread ] [INFO ] Loaded artist_track to PostgreSQL
2025-08-03 08:11:20,897 [MainThread ] [INFO ] Loaded track_metadata to PostgreSQL
2025-08-03 08:11:29,468 [MainThread ] [INFO ] Loaded artist_metadata to PostgreSQL
2025-08-03 08:11:29,469 [MainThread ] [INFO ] Load stage completed
2025-08-03 08:11:29,470 [MainThread ] [INFO ] Total time taken 0 hours, 1 minutes, 55 seconds
2025-08-03 08:11:29,470 [MainThread ] [INFO ] Closing down clientserver connection
ubuntu@ip-172-31-40-25:~/etl$

```

let us go to our pgadmin to verify that tables are populated with data



The screenshot shows the pgAdmin 4 interface. On the left, the 'Object Explorer' pane displays the 'public' schema with a list of tables. The 'artist_metadata' table is selected. The main pane shows the 'Query' editor with the following SQL query:

```

1 SELECT * FROM public.artist_metadata
2 LIMIT 100
3

```

Below the query editor, the 'Data Output' pane displays the results of the query. It shows 7 rows of data with the following columns: id, name, followers, and popularity.

id	name	followers	popularity
1	0002XY9y3JhjzTZqNCqEc	1828	0
2	00045gNg7mLEf9UY9yhD0t	829	13
3	00062105bTKjdbdc1K1UpI	0	0
4	0009LMJQT62wdZVgnM4v...	97	34
5	000BblCihJeKvtiq5aiHOs	1994	34
6	000DEqM4VJDJIUfJ2LNk1	0	0
7	000lpCJT0EX5T7oDZtXSeW	0	0

Congratulation! We have now loaded your data into remote postgres database using AWS cluster setup.

If the issue of driver persists we might need to stop and start the worker and slave nodes

In the master node run stop-master.sh

```

ubuntu@ip-172-31-40-25:~/etl$
ubuntu@ip-172-31-40-25:~/etl$
ubuntu@ip-172-31-40-25:~/etl$ stop-master.sh
stopping org.apache.spark.deploy.master.Master
ubuntu@ip-172-31-40-25:~/etl$

```

In the worker nodes run stop-worker.sh It might not be strictly necessary to stop the workers but for safety let's do it

```
ubuntu@ip-172-31-84-134:~$ stop-worker.sh
stopping org.apache.spark.deploy.worker.Worker
ubuntu@ip-172-31-84-134:~$ start-worker.sh spark://172.31.40.25:7077
```

Do this on both worker's nodes

Now let us start all 3 nodes again

```
ubuntu@ip-172-31-40-25:~/etl$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.master-1-ip-172-31-40-25.out
```

```
ubuntu@ip-172-31-84-118:~$ start-worker.sh spark://172.31.40.25:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.worker-1-ip-172-31-84-118.out
```

```
ubuntu@ip-172-31-84-134:~$ start-worker.sh spark://172.31.40.25:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-ubuntu-org.apache.spark.deploy.worker-1-ip-172-31-84-134.out
```

Verify in the web-ui that everything is working fine and master is connected to worker and run the above code