Hands-on Workshop details

## 

[Objective of this workshop 1](#_Toc152057632)

[How to access the workshop 2](#_Toc152057633)

[Create an S3 bucket and upload a file [Optional] 3](#_Toc152057634)

[Create your own Virtual Private Cloud 5](#_Toc152057635)

[Prepare the server to build and host your application 7](#_Toc152057636)

[Create and launch your Linux virtual machine 7](#_Toc152057637)

[Create EC2 host and install packages 7](#_Toc152057638)

[Create a database for your application 14](#_Toc152057639)

[Create an RDS Postgresql database 14](#_Toc152057640)

[Create a Secret for the database credentials 17](#_Toc152057641)

[Run scripts to create table and populate DB 19](#_Toc152057642)

[Build the demo application 21](#_Toc152057643)

[Install Java on the box. 21](#_Toc152057644)

[Set-up maven and git and download source from git 22](#_Toc152057645)

[Download and build the demo application 24](#_Toc152057646)

[Run your first application to the cloud 25](#_Toc152057647)

[Run application and access it 28](#_Toc152057648)

[Steps to create application machine image (AMI) and start application on booting 31](#_Toc152057649)

[Add a start-up script 31](#_Toc152057650)

[Create an AMI 32](#_Toc152057651)

[Launch the AMI 33](#_Toc152057652)

[Load balancer to route traffic to all instances launched [Optional] 37](#_Toc152057653)

[Feedback 38](#_Toc152057654)

## 

## Objective of this workshop

This workshop walks-through steps to set-up a deployment environment (VPC, Database) for sample Java Spring boot application followed by hosting the application on one and them multiple EC2s. The java application exposes a http based APIs on port 8080.

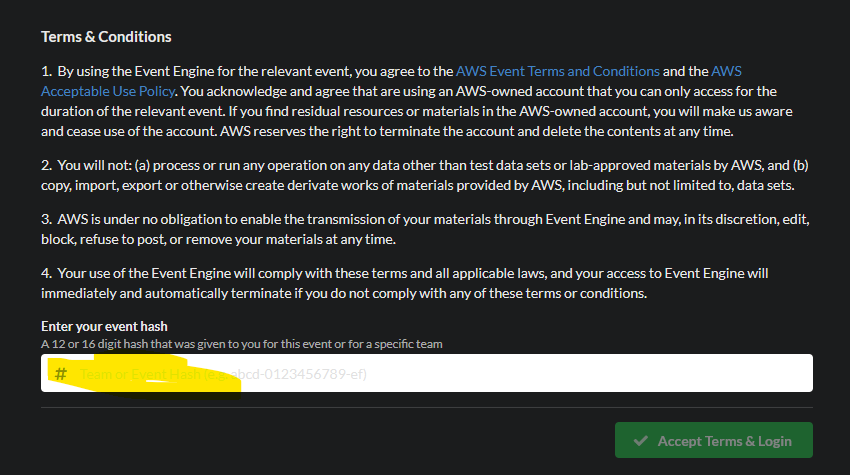
We will go through details of setting up security groups and IAM roles during the workshop.

By the end of this workshop you will be ready to set-up a basic Java based application on the cloud and host it on multiple VMs. This will cover the basics needed to build for the Hackathon event.

## How to access the workshop

Go to Event engine URL: <https://dashboard.eventengine.run/login>

Enter event hash: **<<EVENT HASH CODE>>(REFER EXCEL SHEET)**



Follow instructions on the page to login to your AWS console.

## Create an S3 bucket and upload a file [Optional]

Search for S3 service

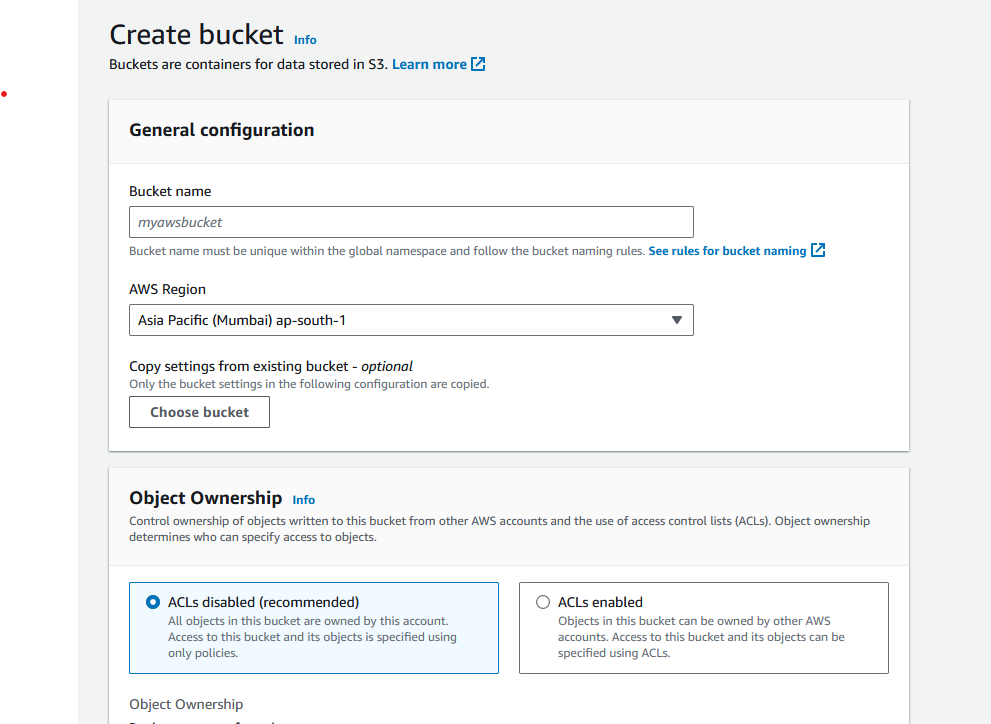


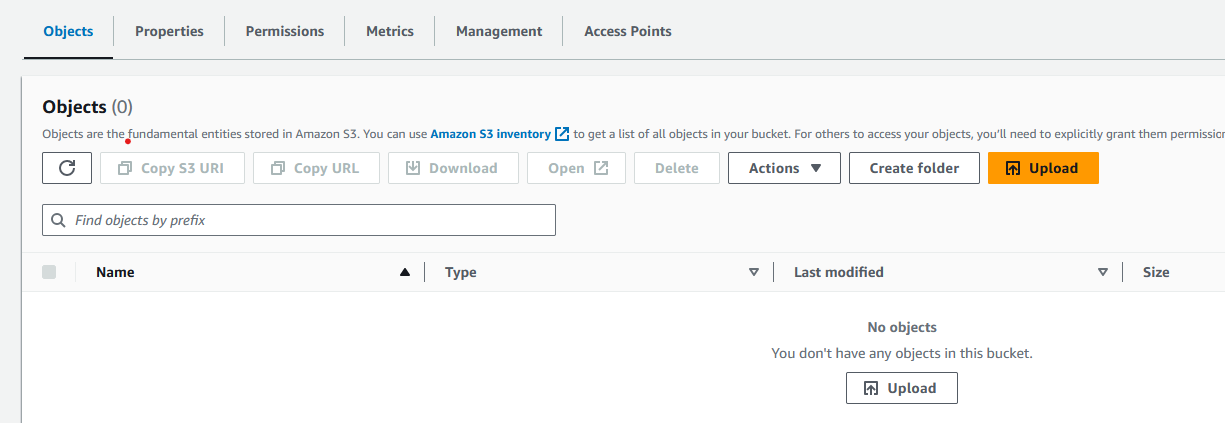
Open S3 service in new tab and click Create bucket

Given a unique name to bucket : “myfirstbucket-yourname”

Ensure region is Mumbai

Rest options you can retain default



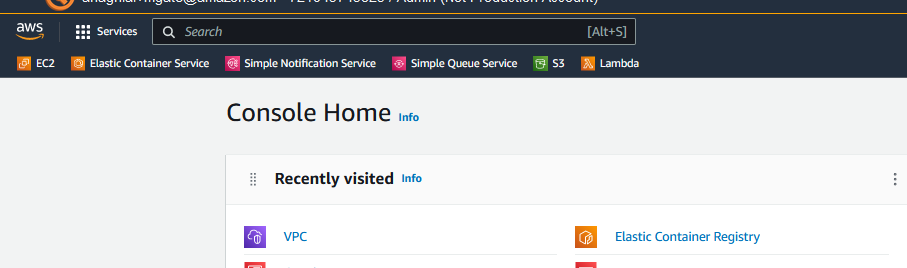
Select your bucket

Create a text file with some dummy text (say a single word)

Upload it on the S3 bucket using the upload button

## Create your own Virtual Private Cloud

On the console, search for the service VPC, open it in a new tab of the browser:



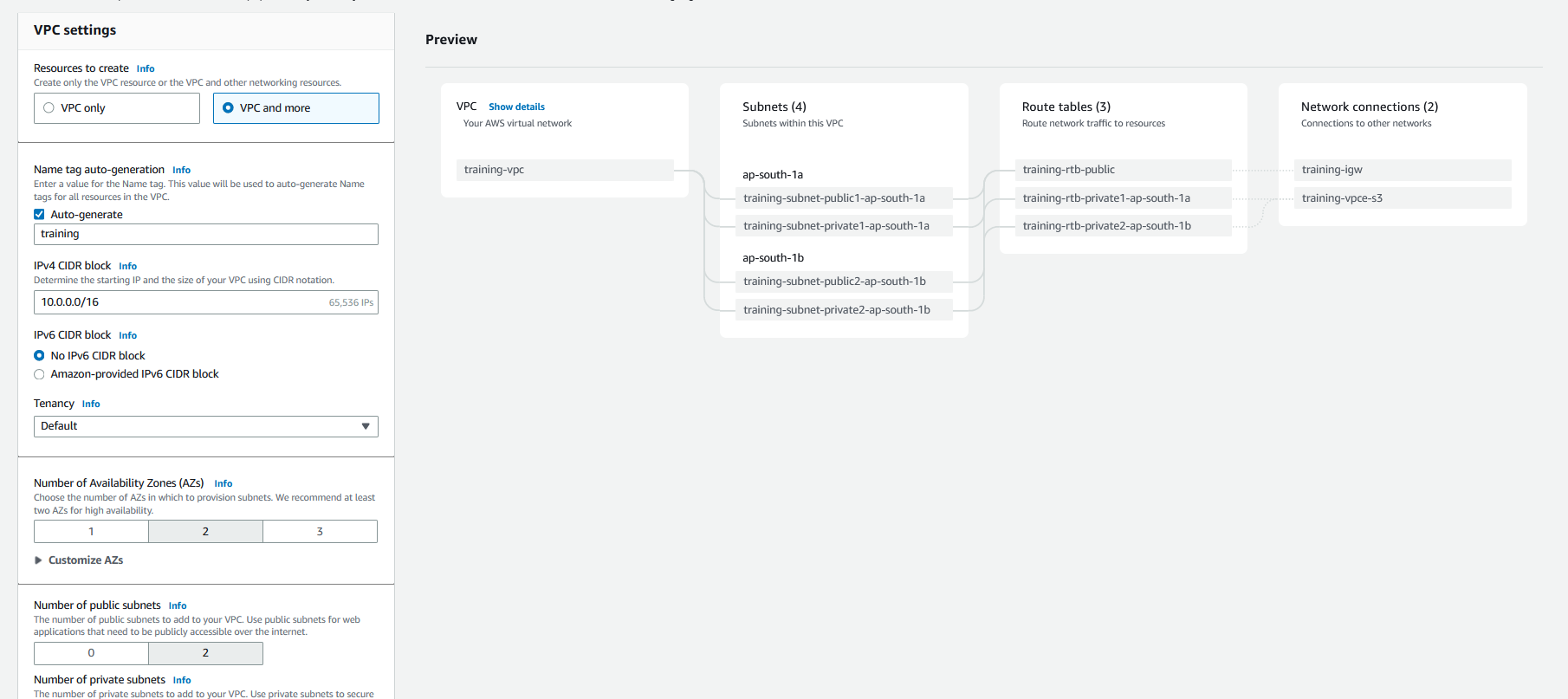
Click on create VPC

Select VPC and more.

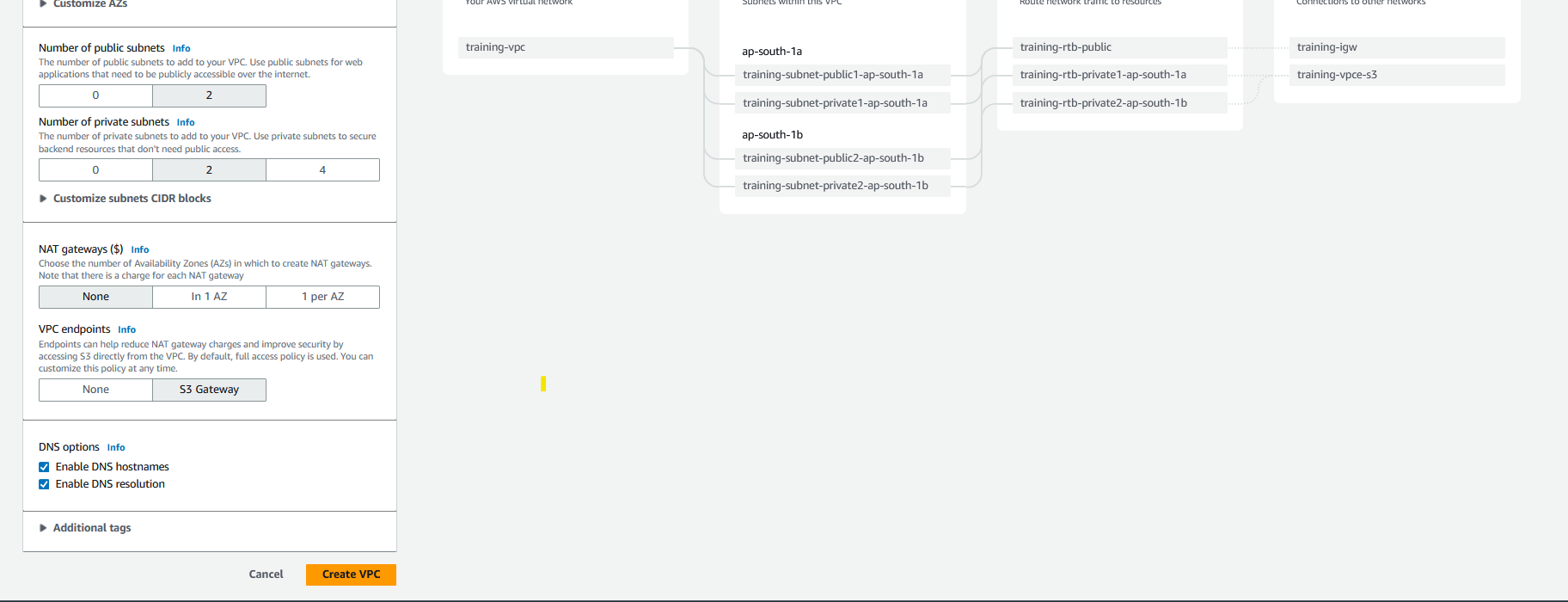
Select a name for the VPC

Keep CIDR block range as default

Keep all other options as default



Click on Create VPC





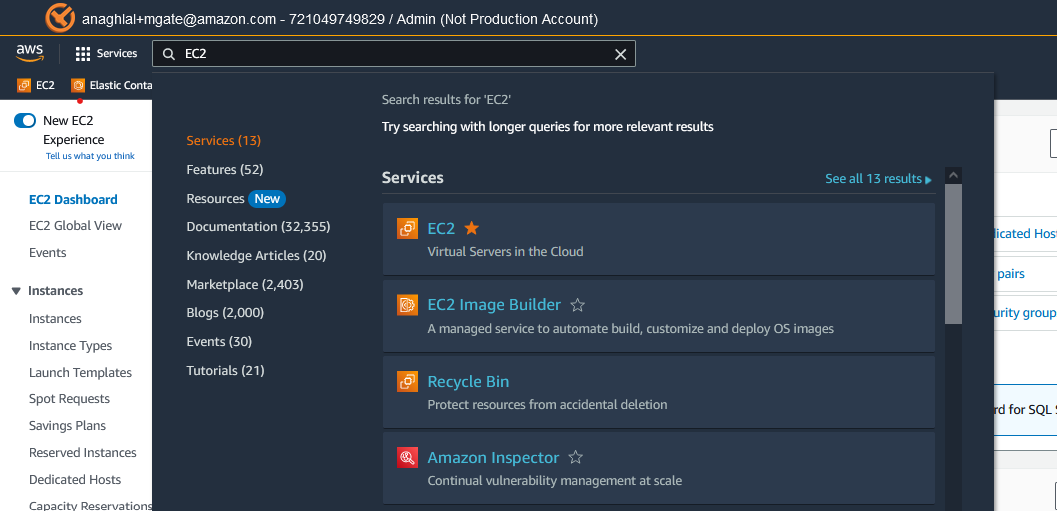
## Prepare the server to build and host your application

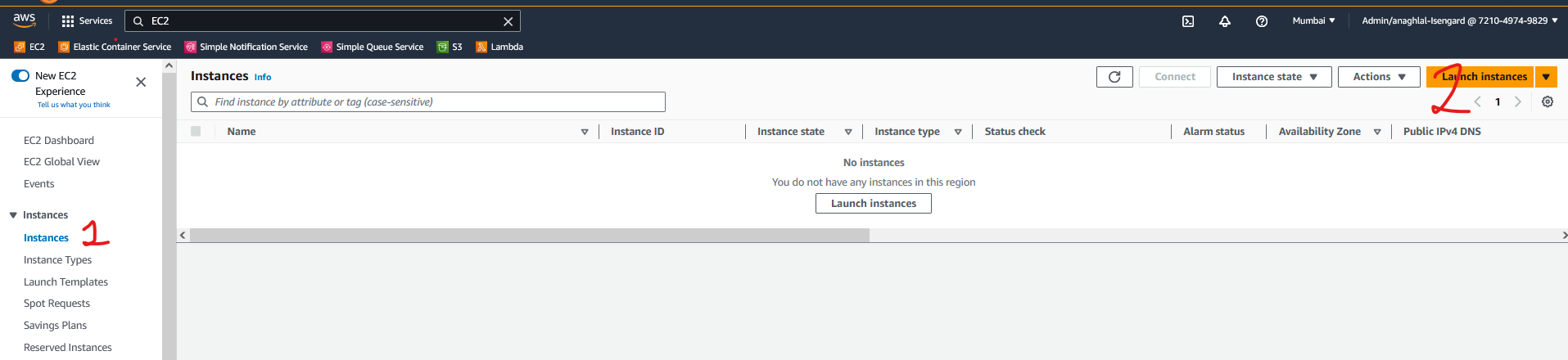
### Create and launch your Linux virtual machine

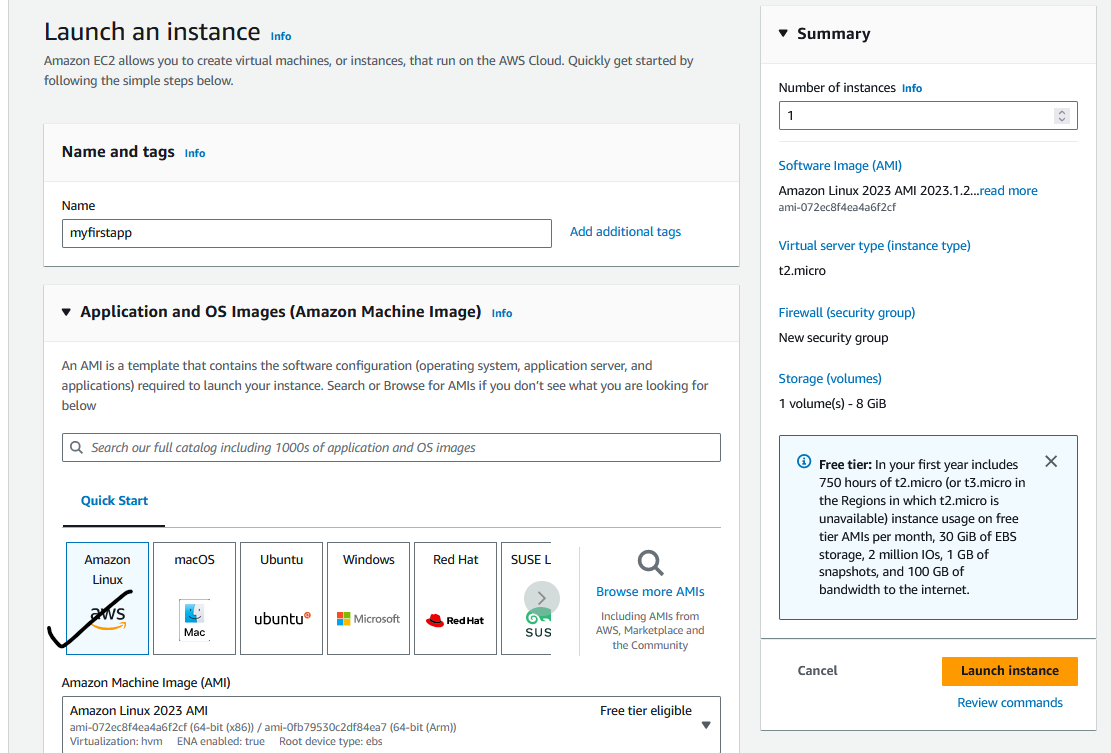
We will create a VM, login and install packages on it to download, build and run the Java application.

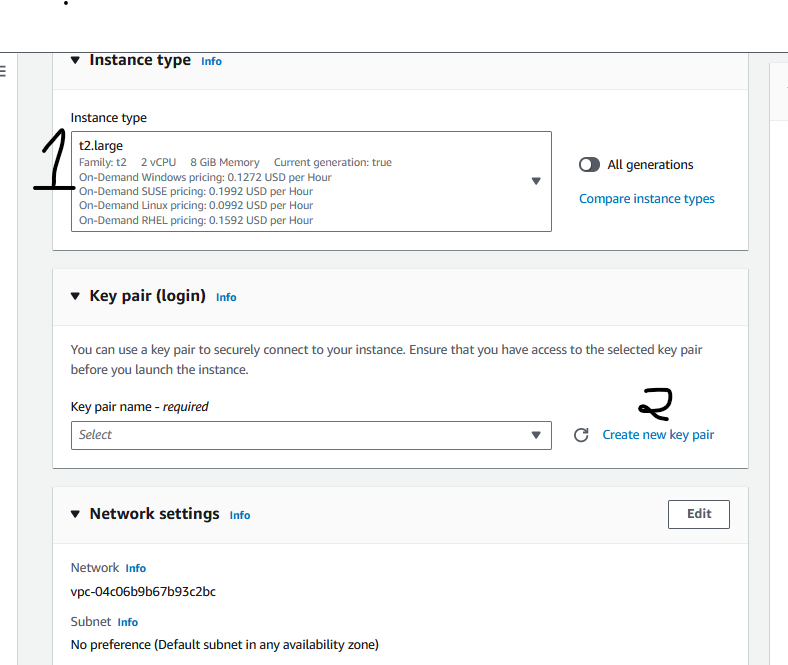
#### Create EC2 host and install packages

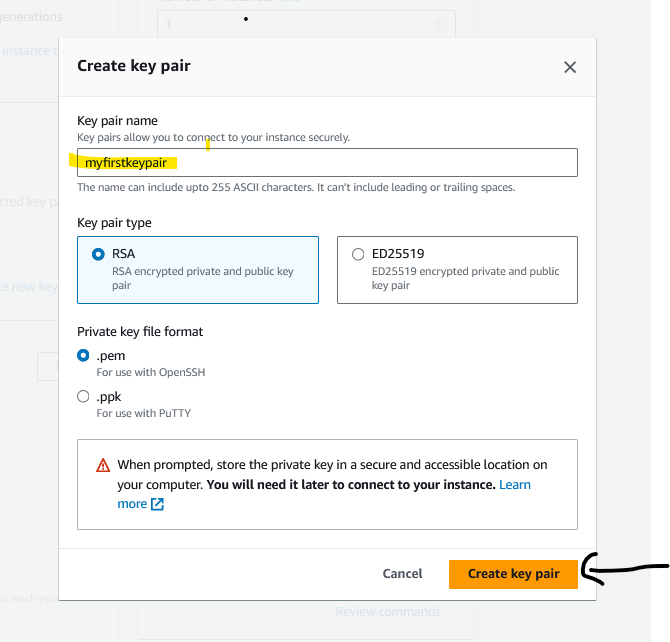
Create an EC2 instance in public subnet.







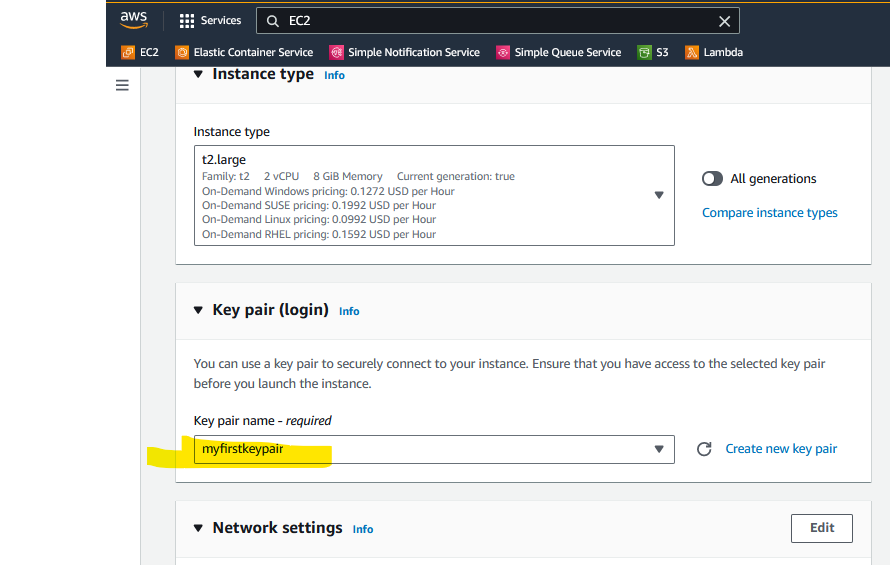




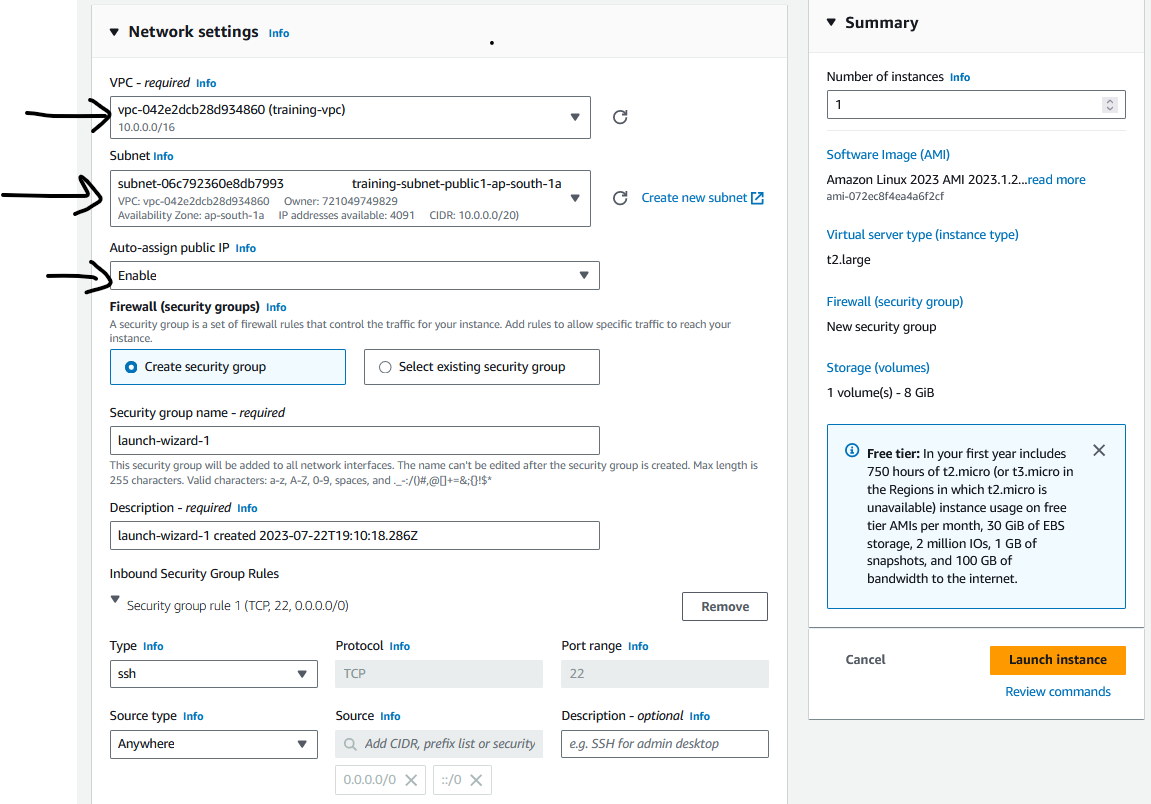
Copy it to a folder: say c:\dev



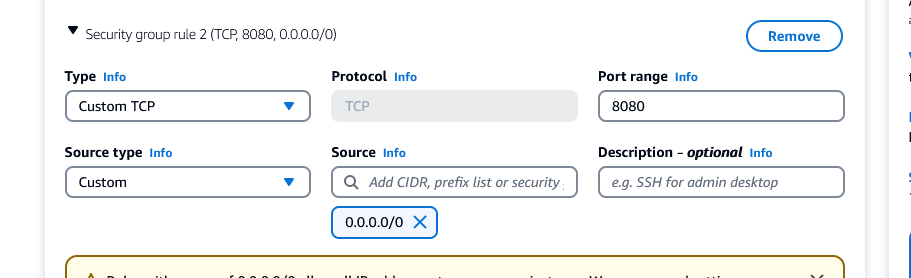
Go back to EC2 screen



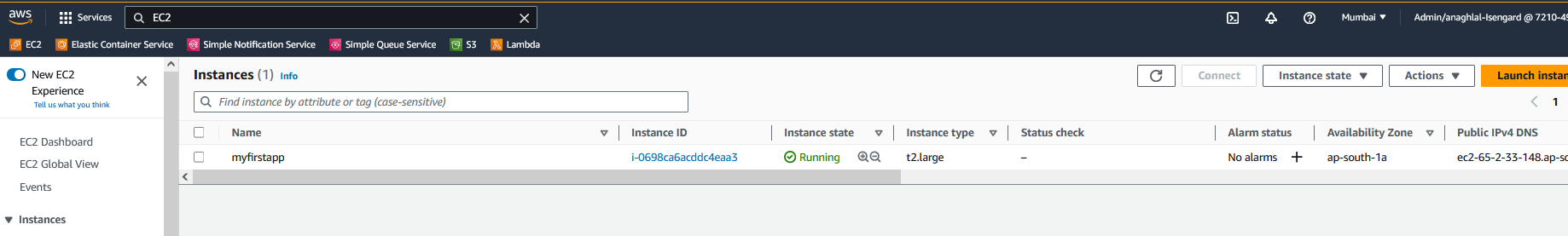
Click on Edit under Network settings:



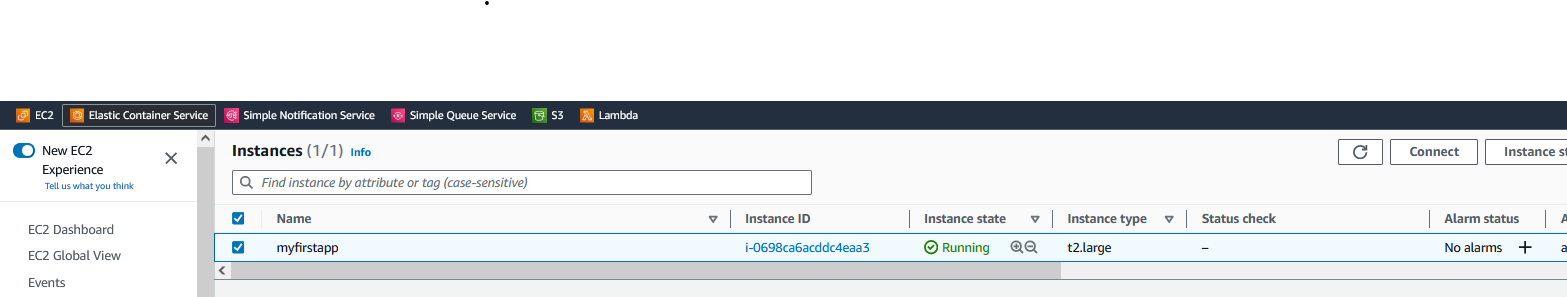
Also add security group rule to allow traffic from TCP port 8080.

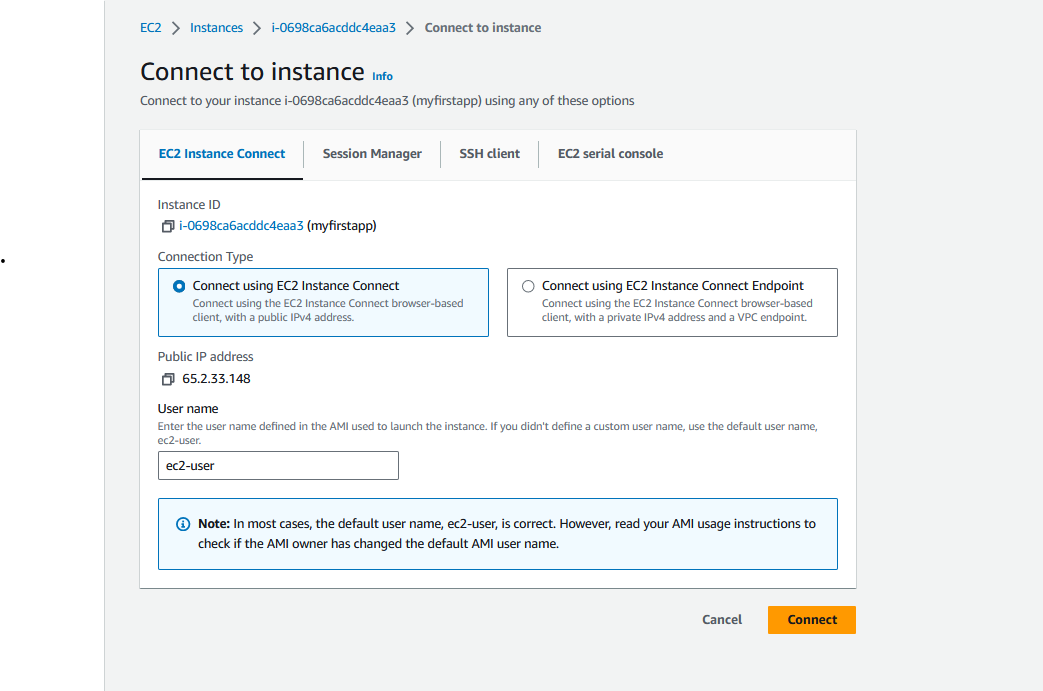


Click on Launch instance

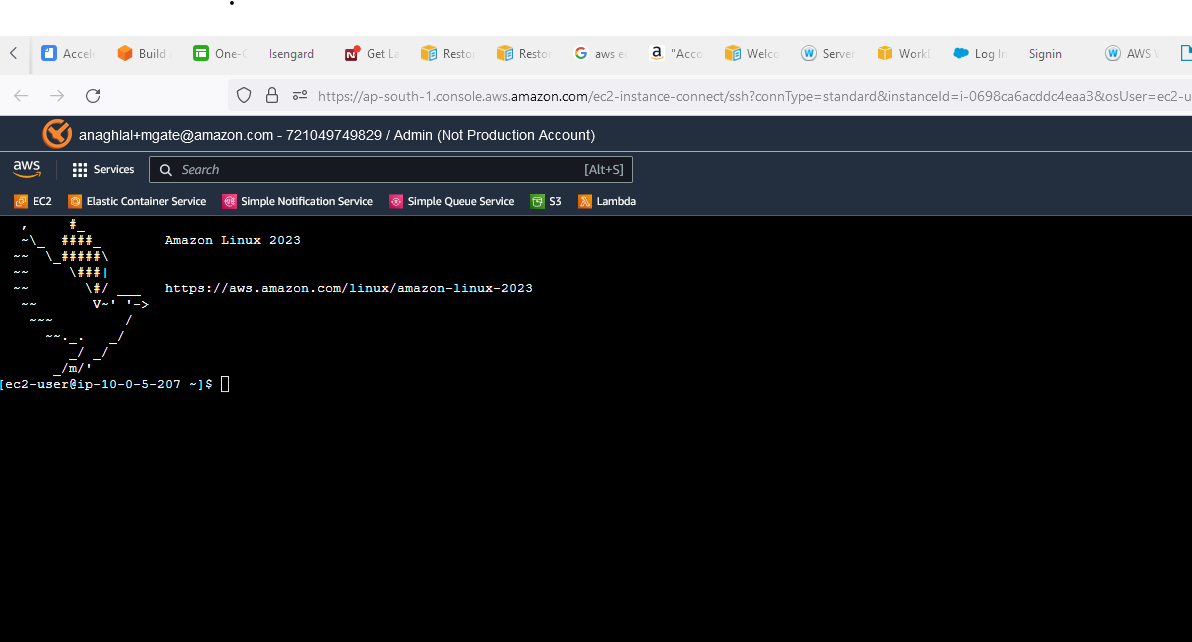


Once instance is in running state, select instance and click connect



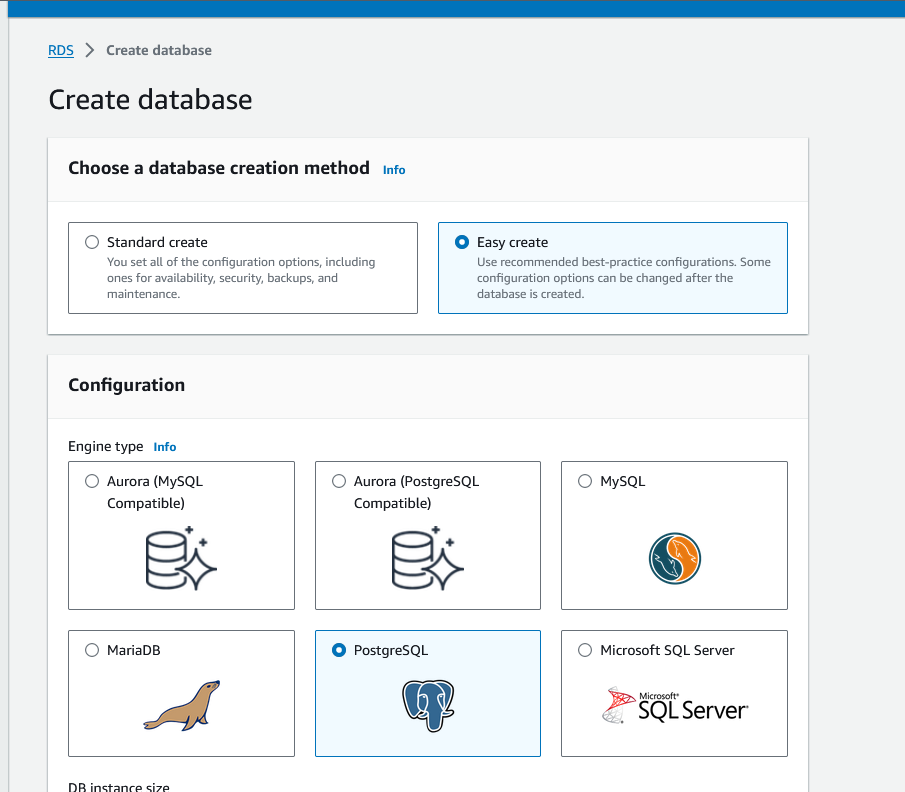


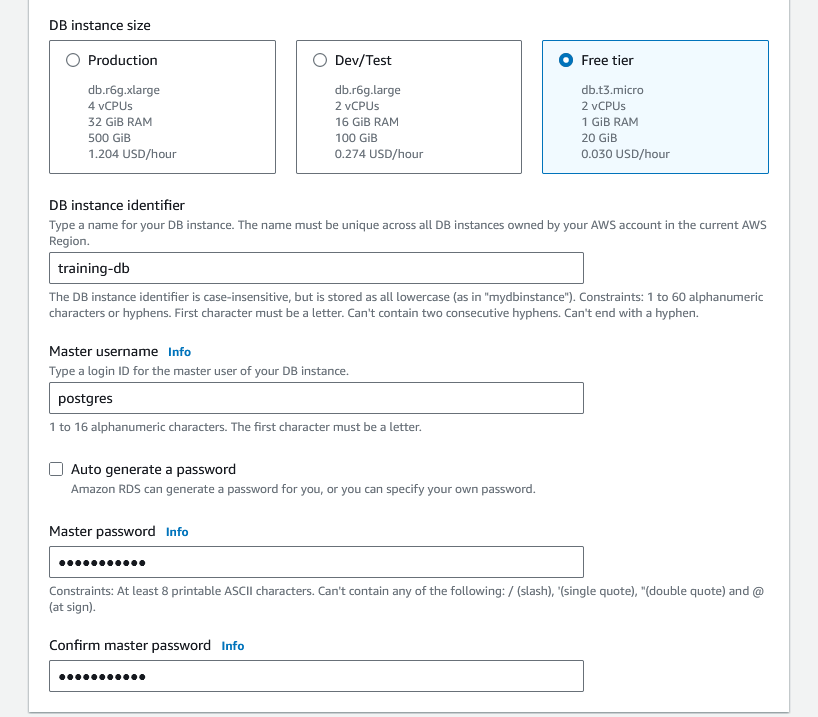
Click connect

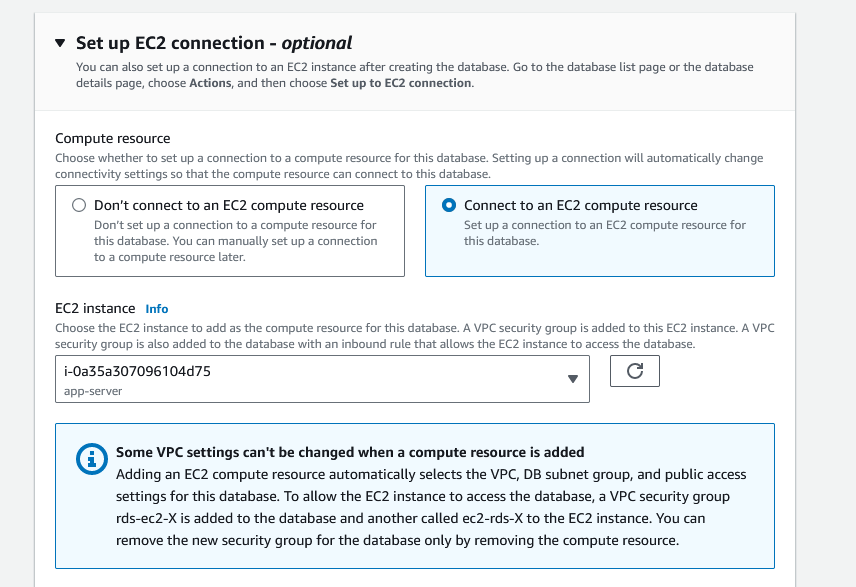


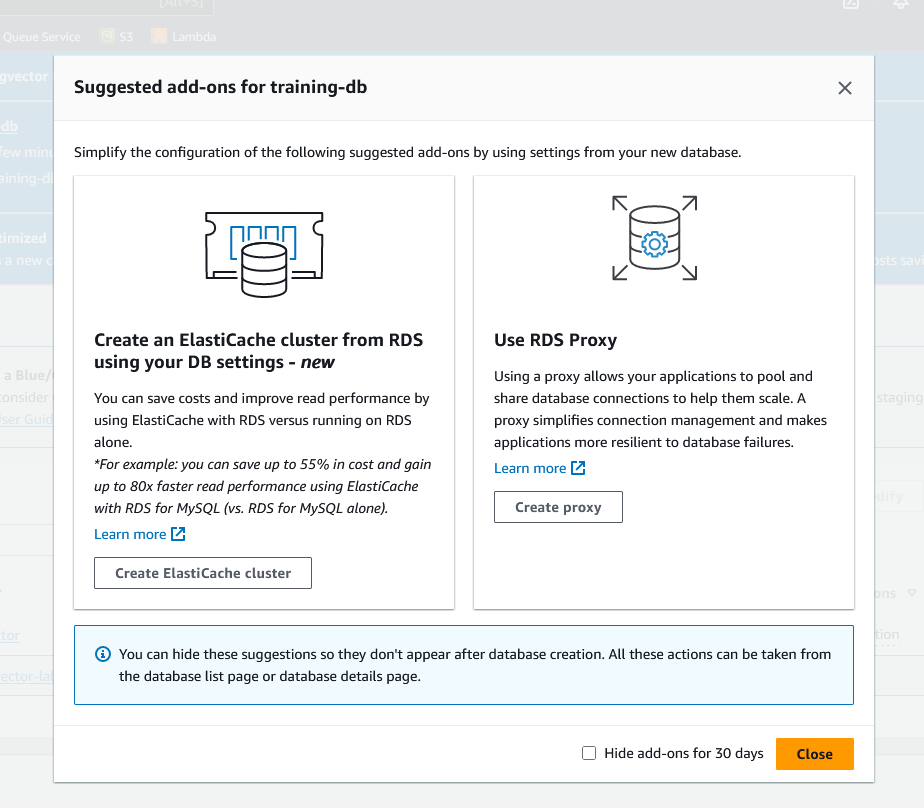
## Create a database for your application

### Create an RDS Postgresql database



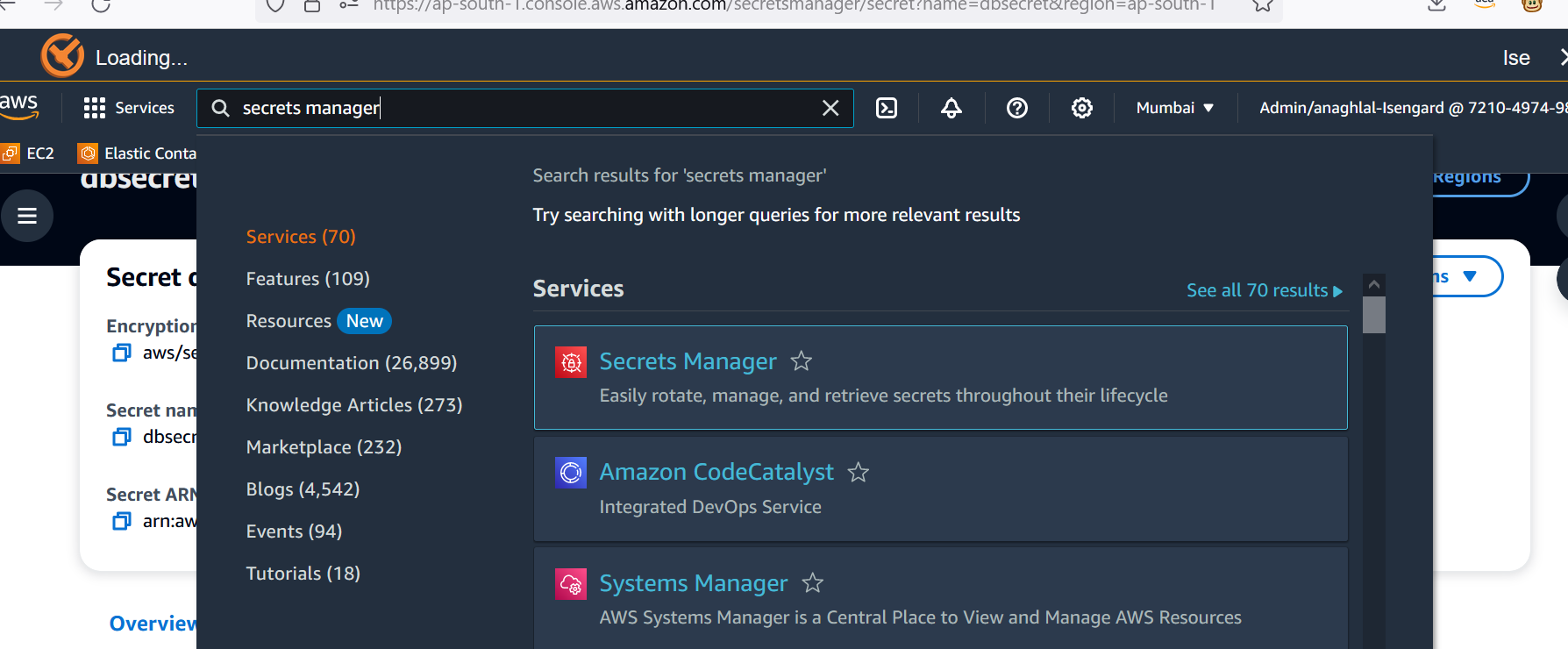




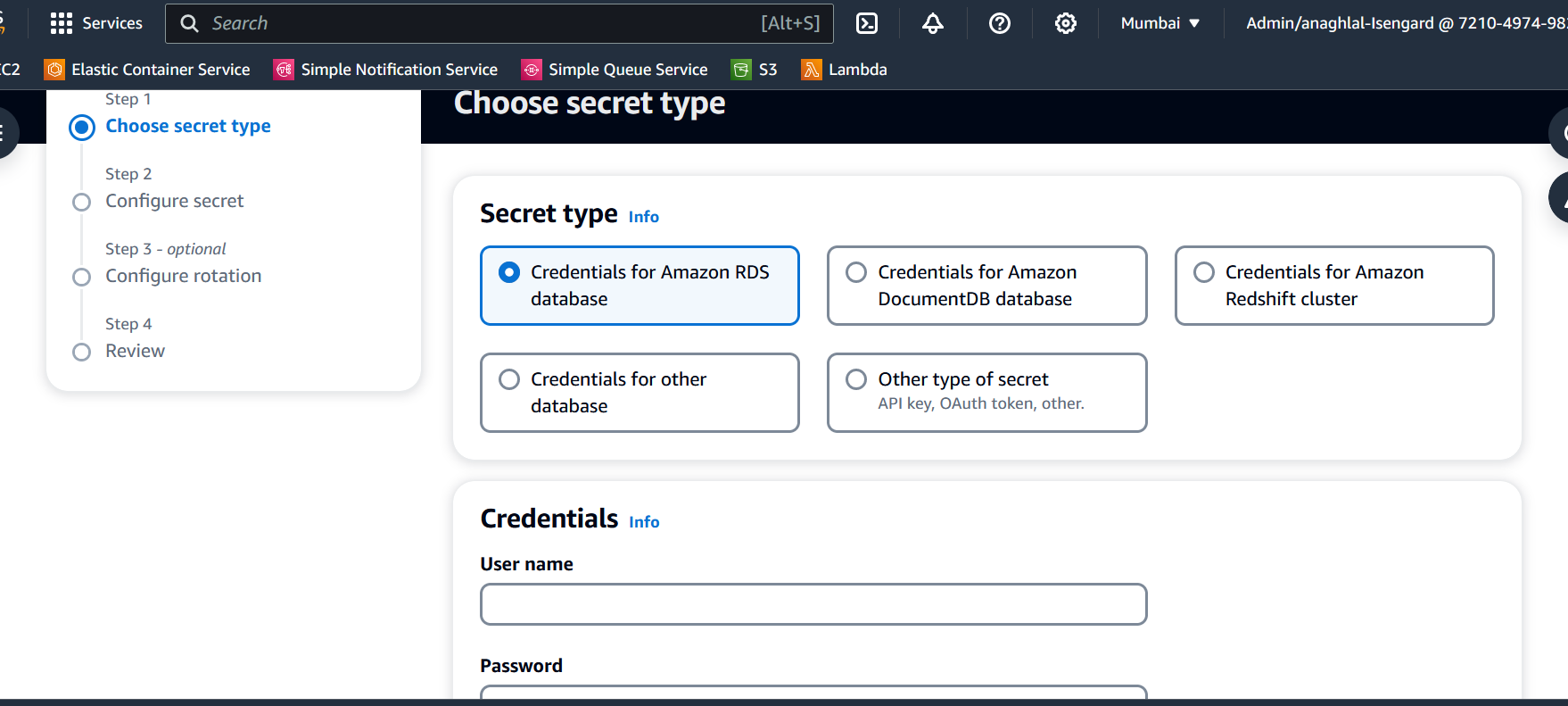


Click close

### Create a Secret for the database credentials

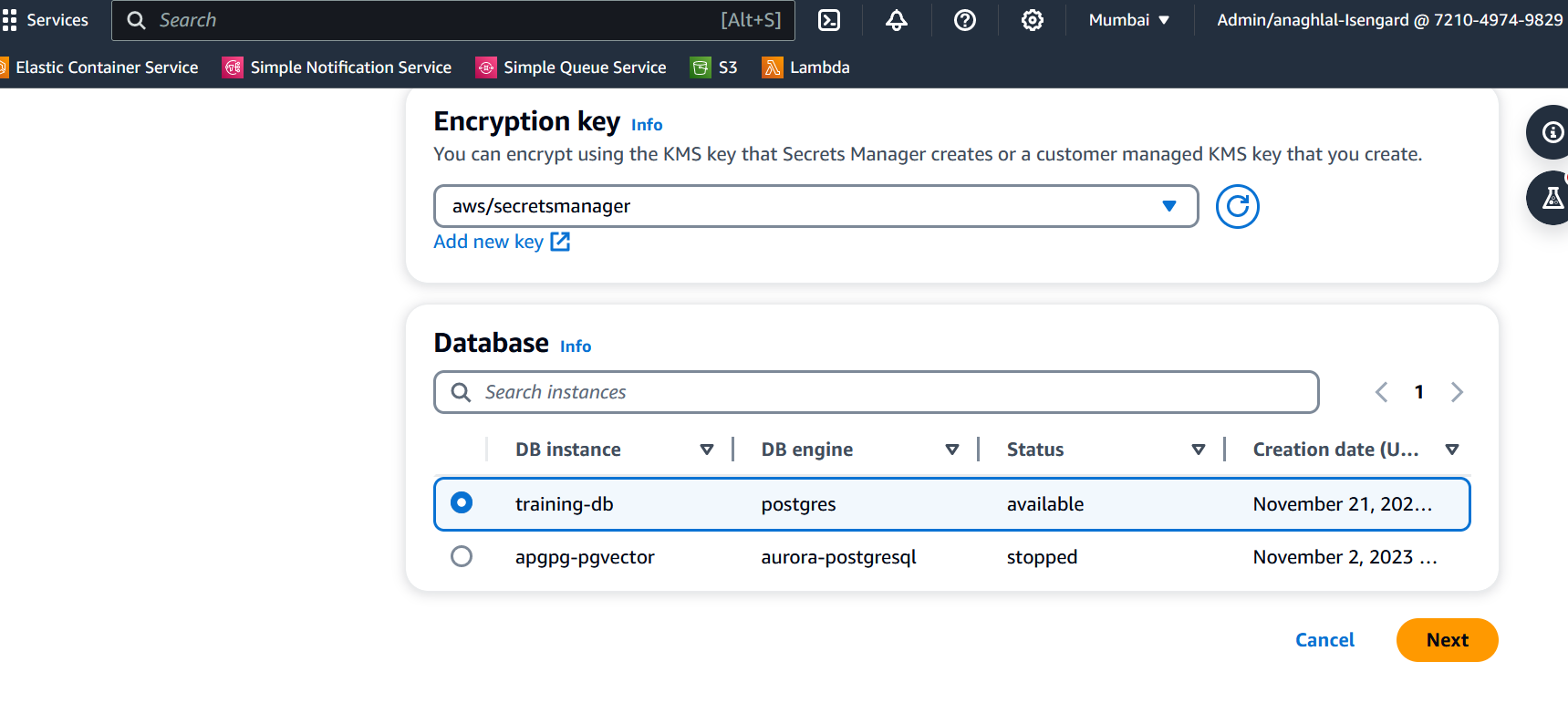


Choose create a new secret



Username: postgres

Password: <entered at db creation time>

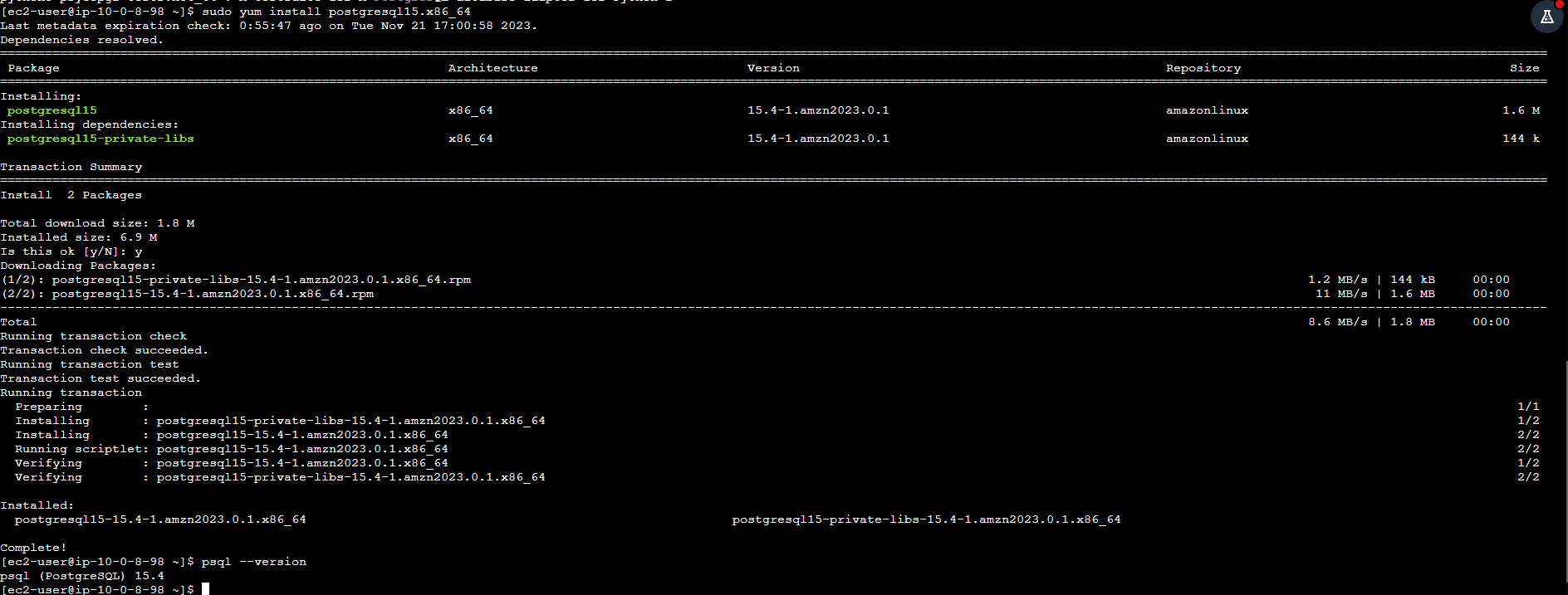




Keep the defaults and create the secret

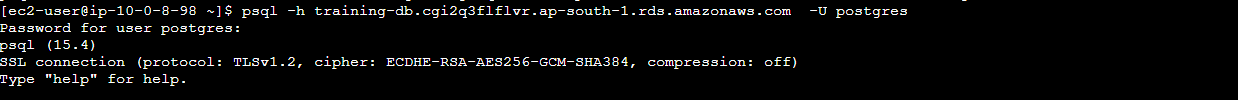
### Run scripts to create table and populate DB

sudo yum install postgresql15.x86\_64



Check the database endpoint from AWS console and replace the endpoint in below command.

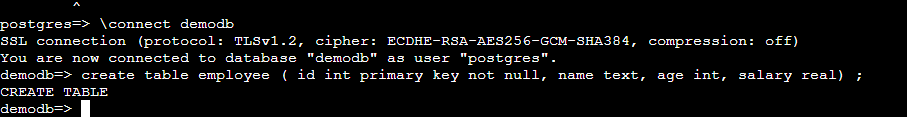
psql -h XXXXXX.ap-south-1.rds.amazonaws.com -U postgres



create database demodb ;

\connect demodb

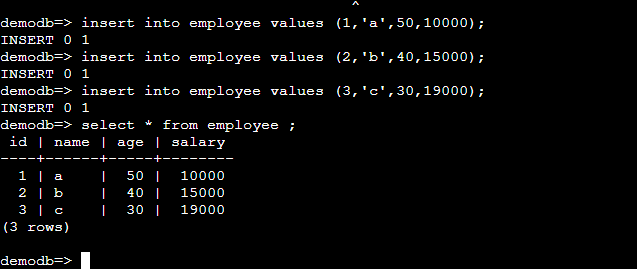
create table employee ( id int primary key not null, name text, age int, salary real) ;



insert into employee values (1,'a',50,10000);

insert into employee values (2,'b',40,15000);

insert into employee values (3,'c',30,19000);



## Build the demo application

Go back to the Ec2 instance (re-connect if you are disconnected)

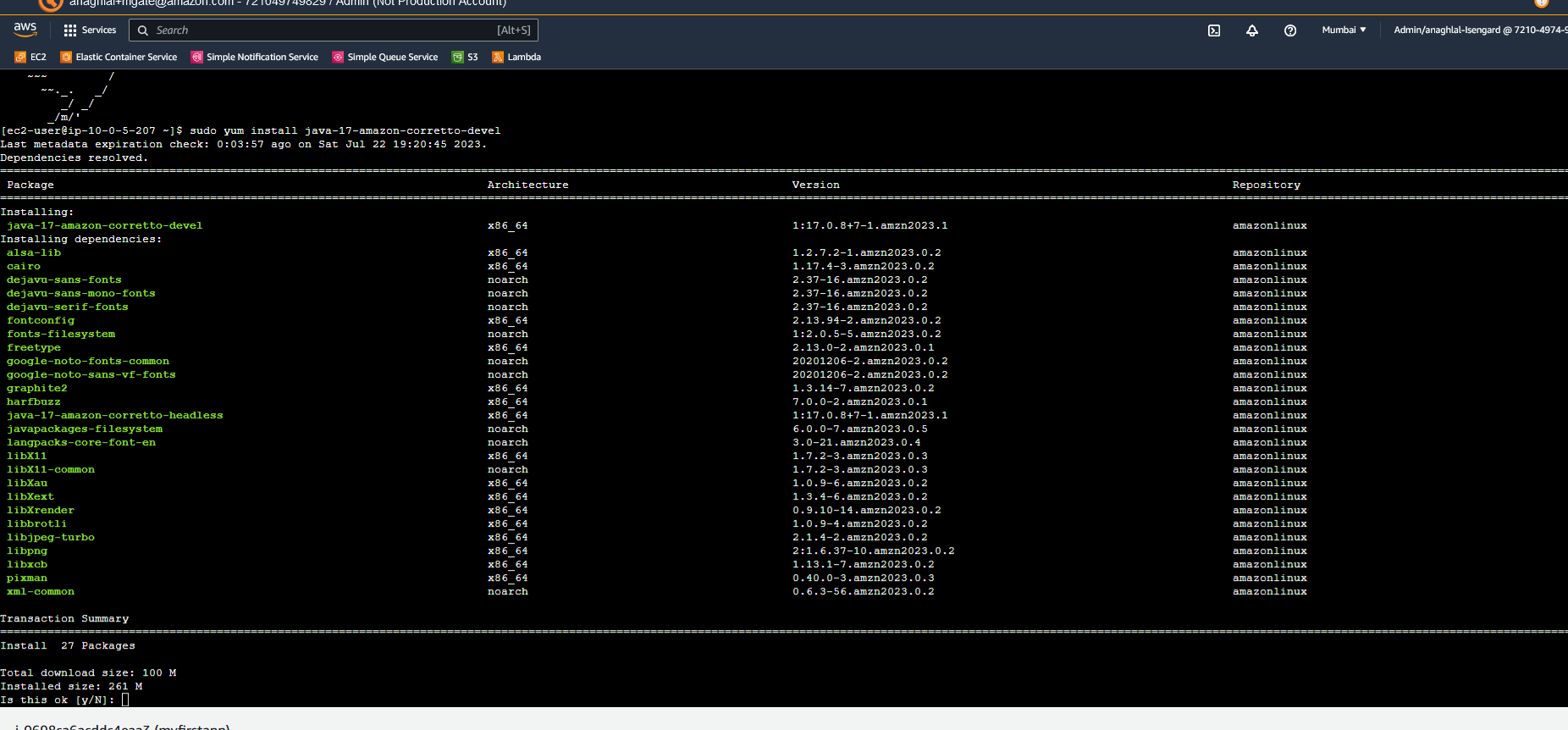
### Install Java

Follow the steps from here

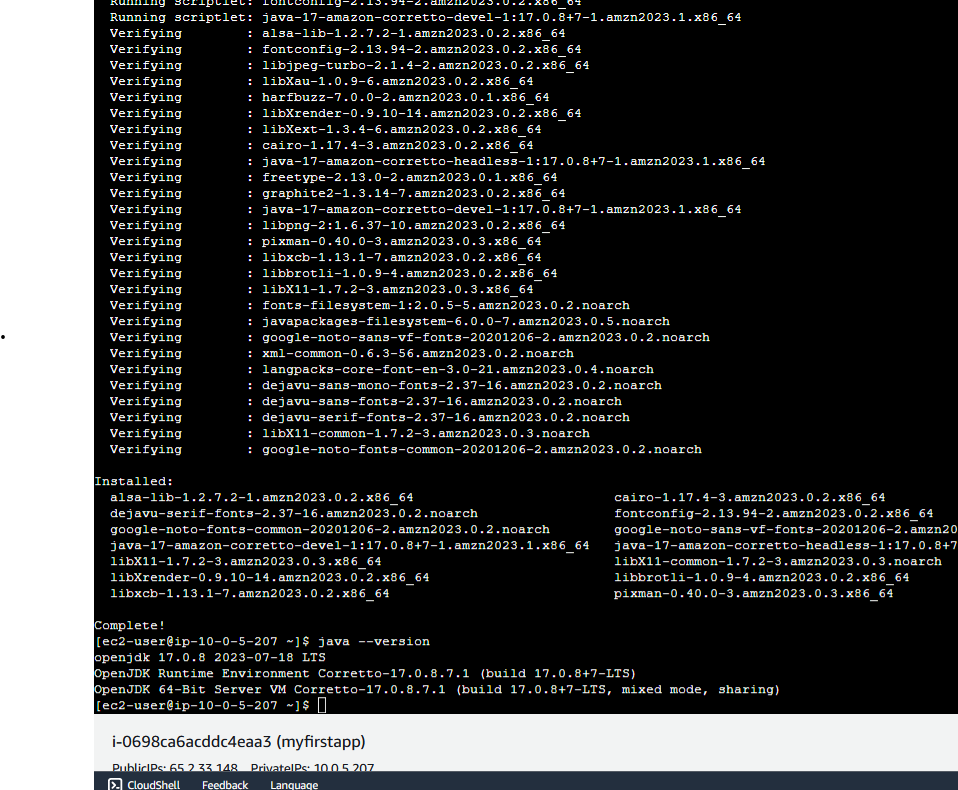
<https://docs.aws.amazon.com/corretto/latest/corretto-17-ug/amazon-linux-install.html>

Or

sudo yum install java-17-amazon-corretto-devel



Select y and finish installation. Verify installation.



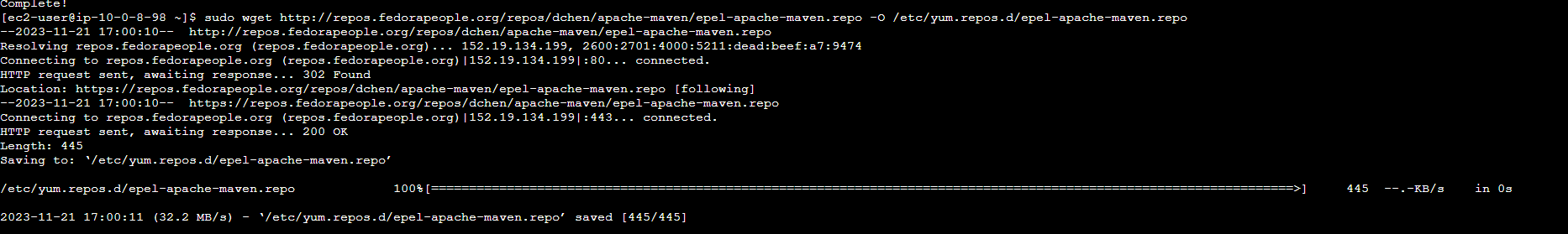
### Set-up maven and git and download source from git

The below command is a single line. Copy paste into notepad and ensure it is single-line and no additional spaces instroduced.

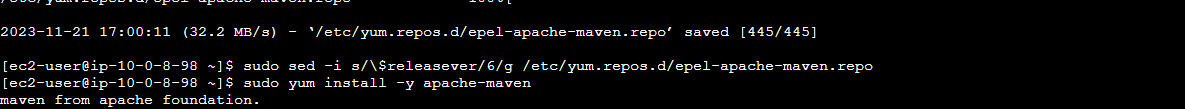
sudo wget [http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo](http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo%20-O%20/etc/yum.repos.d/epel-apache-maven.repo)

For your reference: the url in the command for reference in a single line.

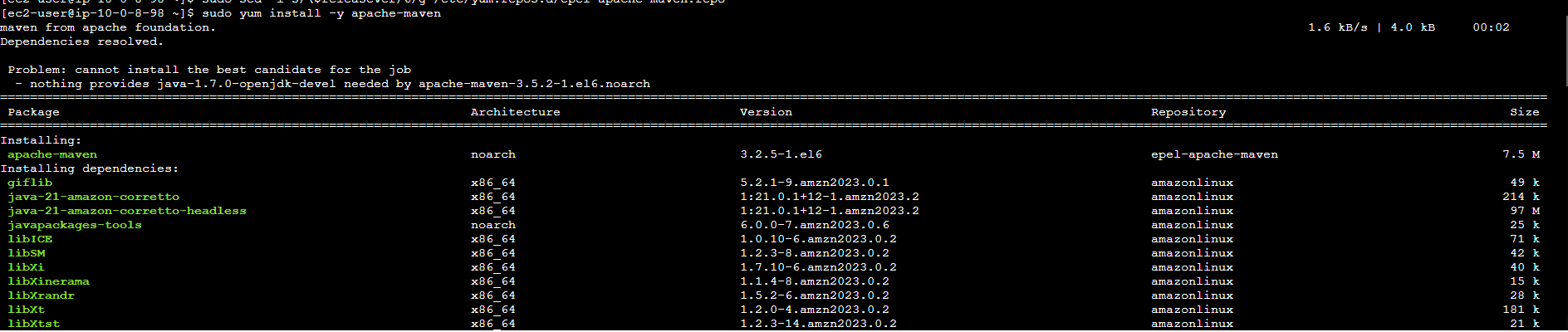
http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo

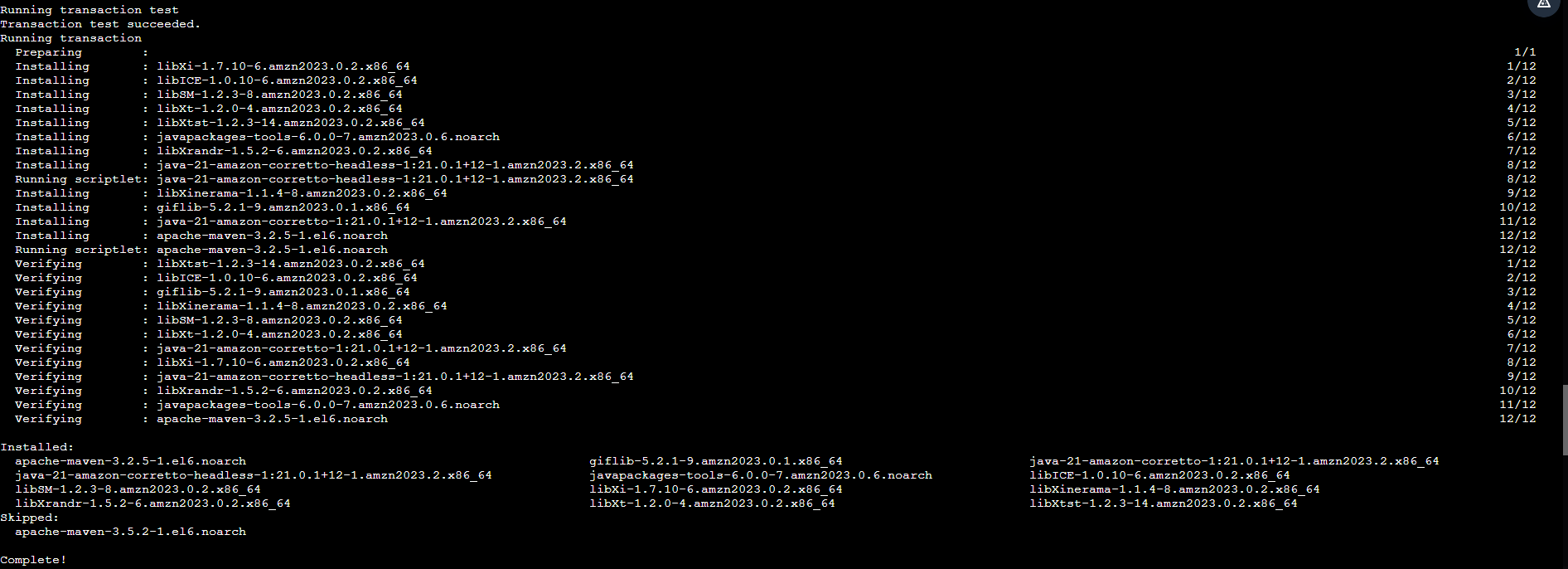


sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

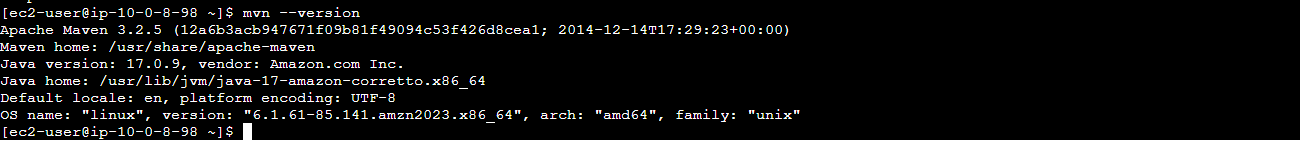


sudo yum install -y apache-maven

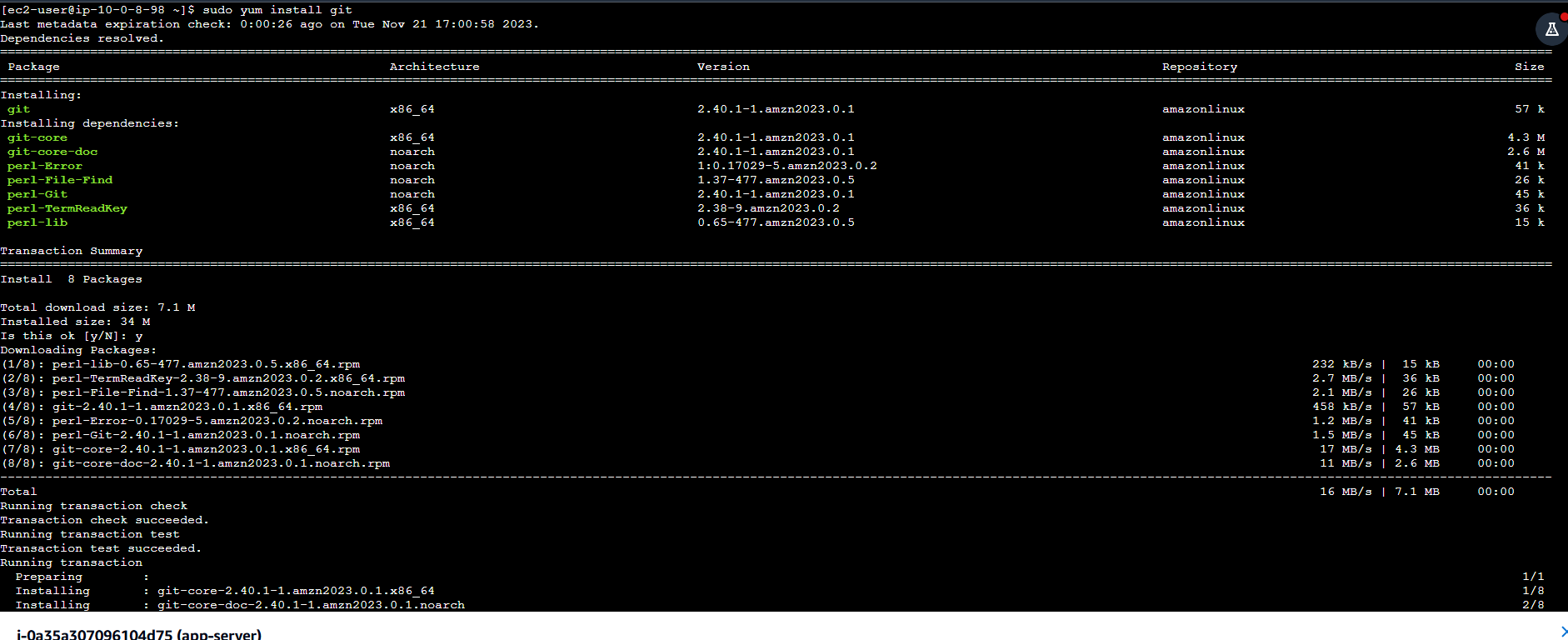


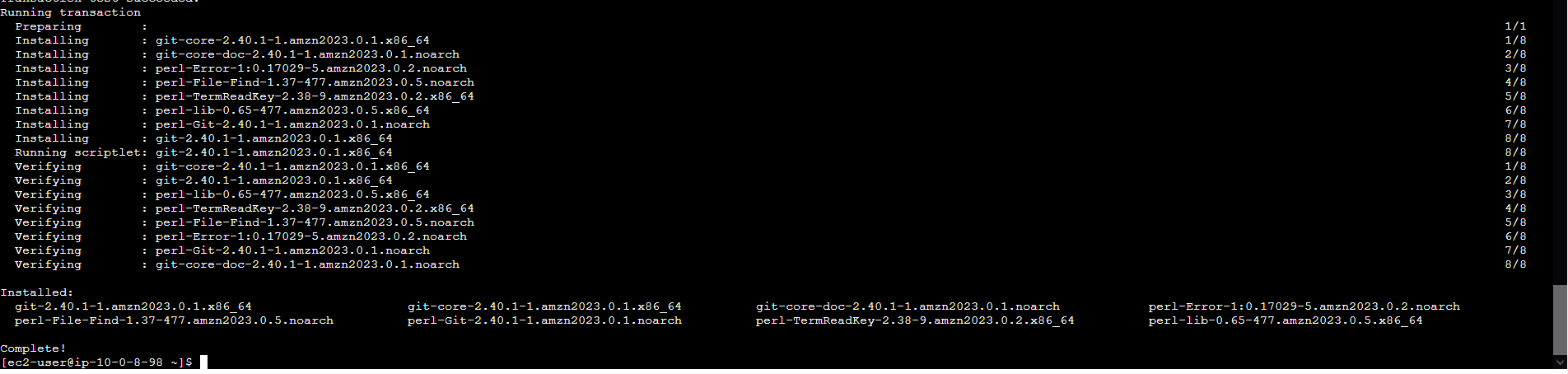


mvn --version

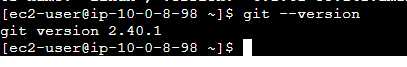


sudo yum install git





git --version



### Download and build the demo application

mkdir lab

cd lab/

git clone https://github.com/anaghlal/workshop.git

cd workshop/

cd intro/

Point your application config to the DB and S3 you created:

cd /home/ec2-user/lab/workshop/intro/

cd src/main/resources/

vi application.properties

Set the following properties

(All entries are on a single line.)

s3.bucket.name=bucket-name

s3.bucket.object.key=y

rds.jdbc.url= jdbc:postgresql://training-db.XXXX.ap-south-1.rds.amazonaws.com:5432/demodb

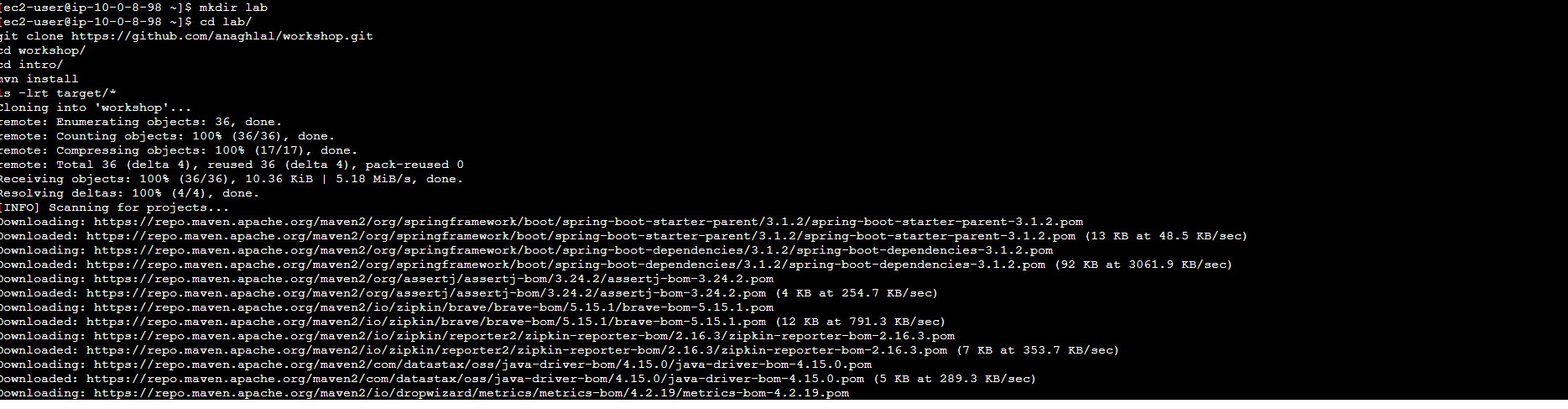
rds.secret.name=dbsecret

Build the package:

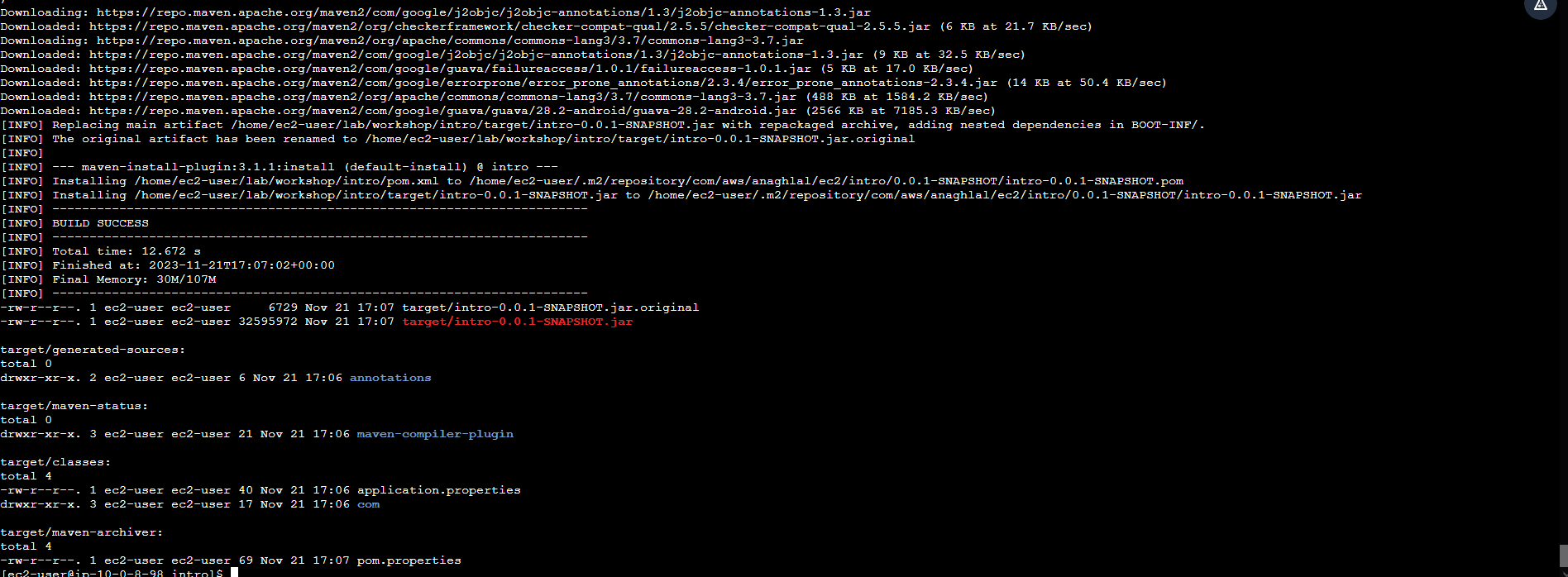
cd /home/ec2-user/lab/workshop/intro/

mvn install

ls -lrt target/\*



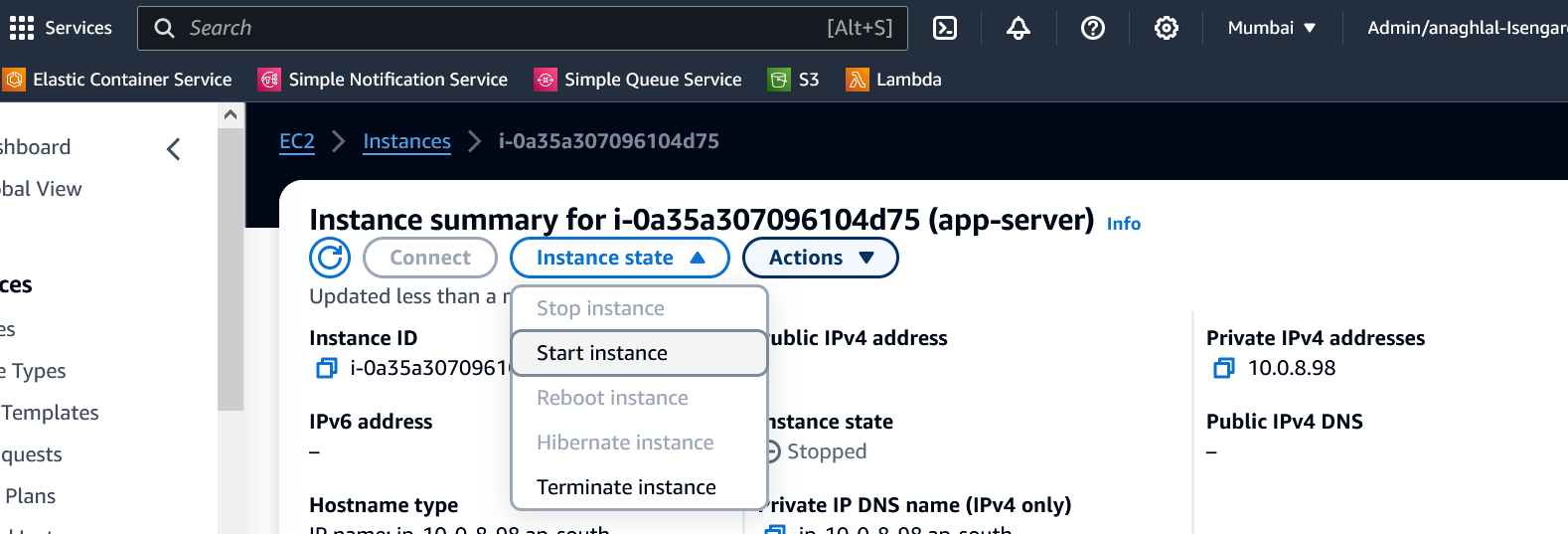
The maven install command will download all the dependencies and take a few minutes to complete



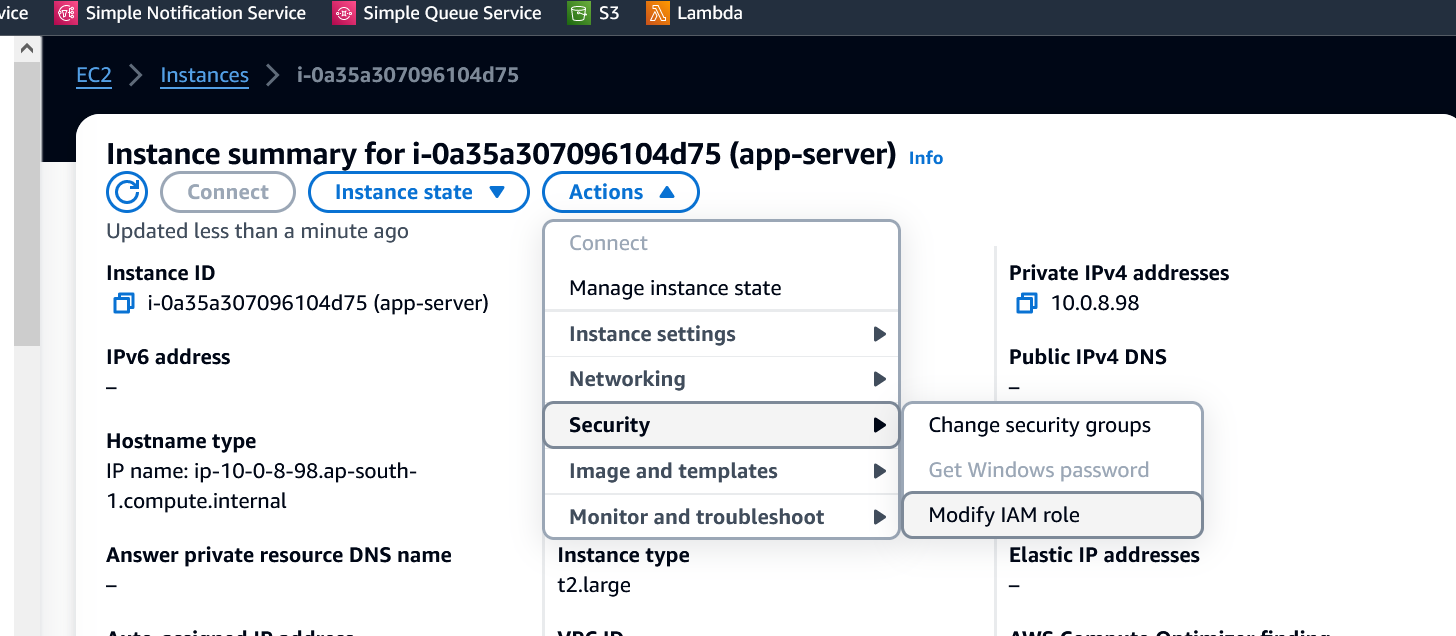
## Run your first application to the cloud

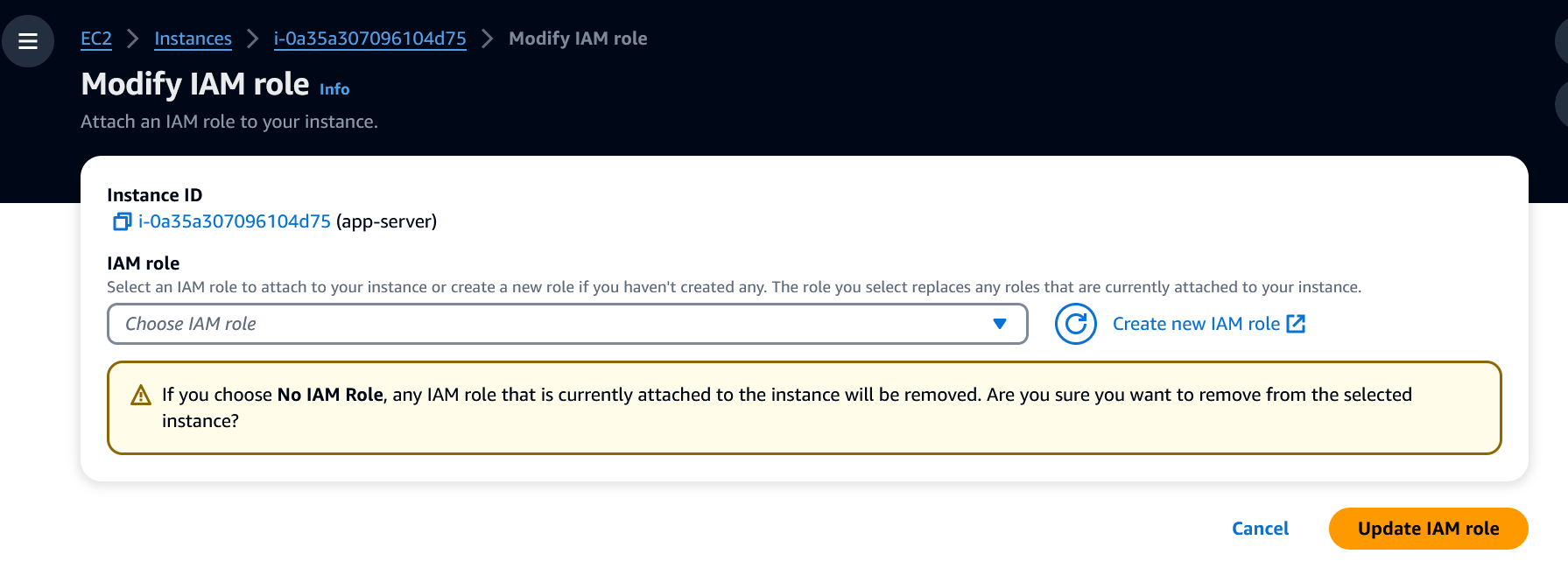
To access S3 and Secrets manager we need to give access to this EC2 instance.

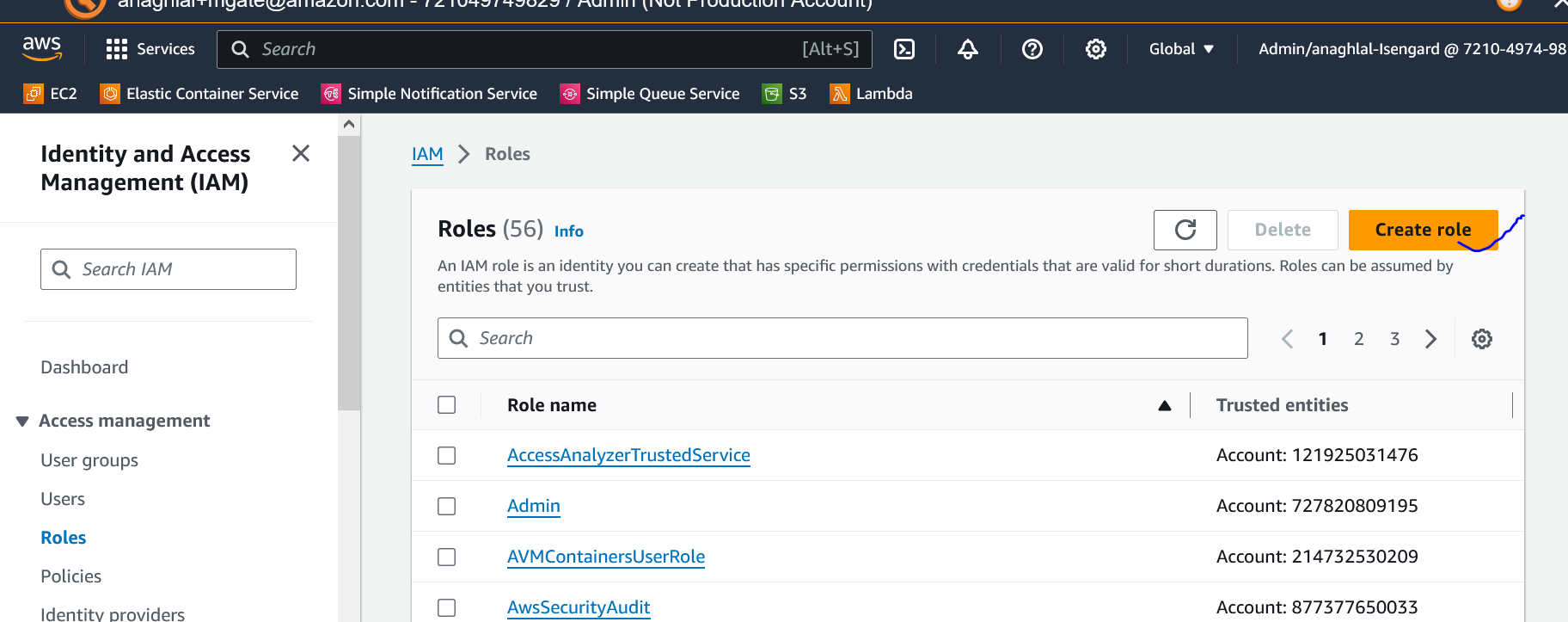
Stop instance

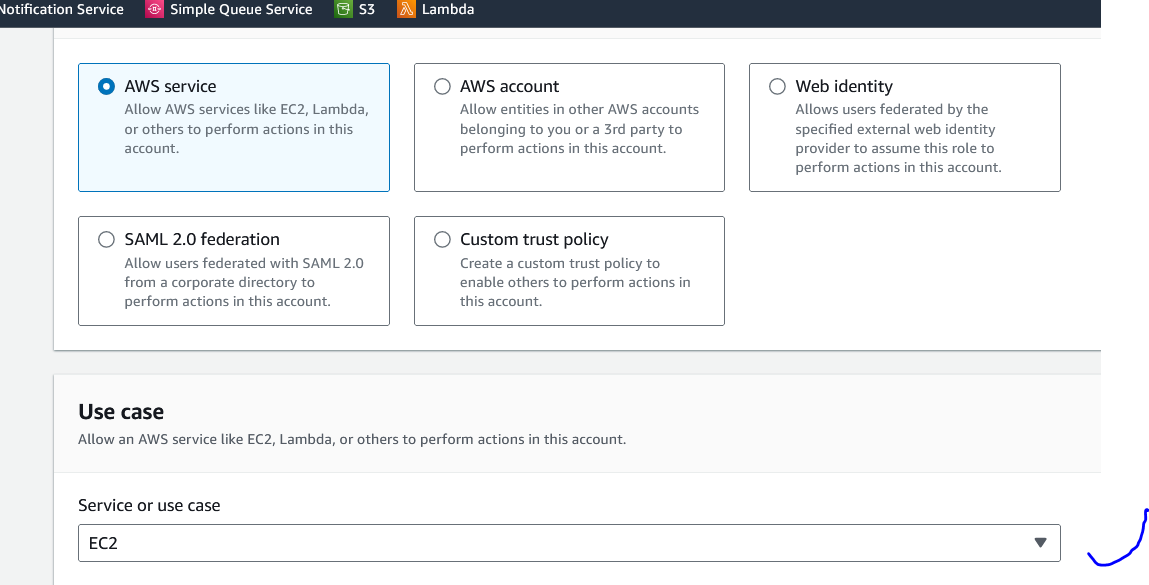


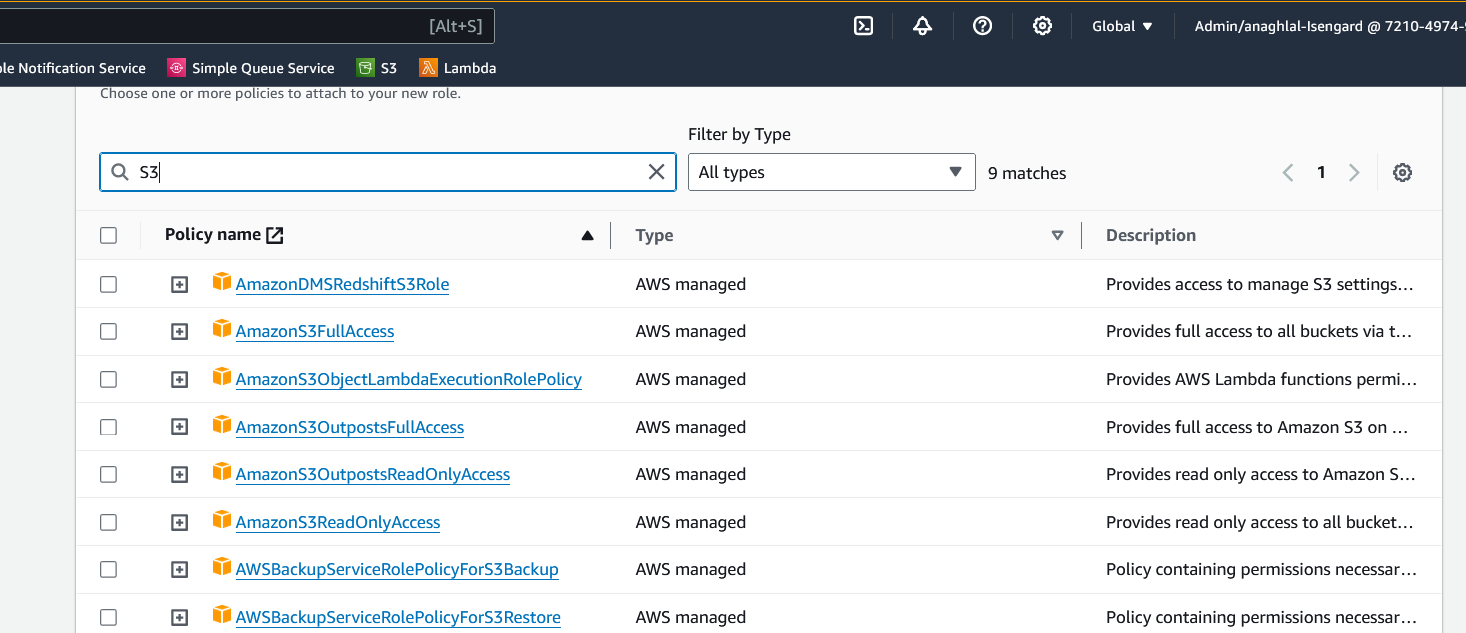
Add IAM role: Access to S3 and secrets manager



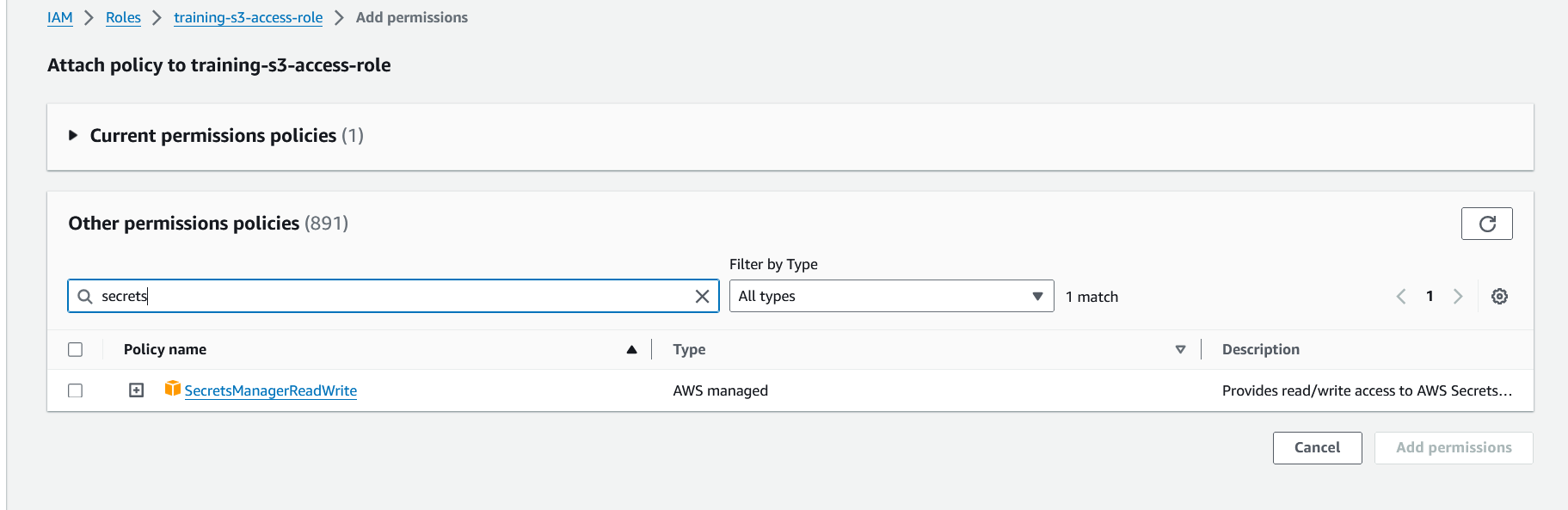


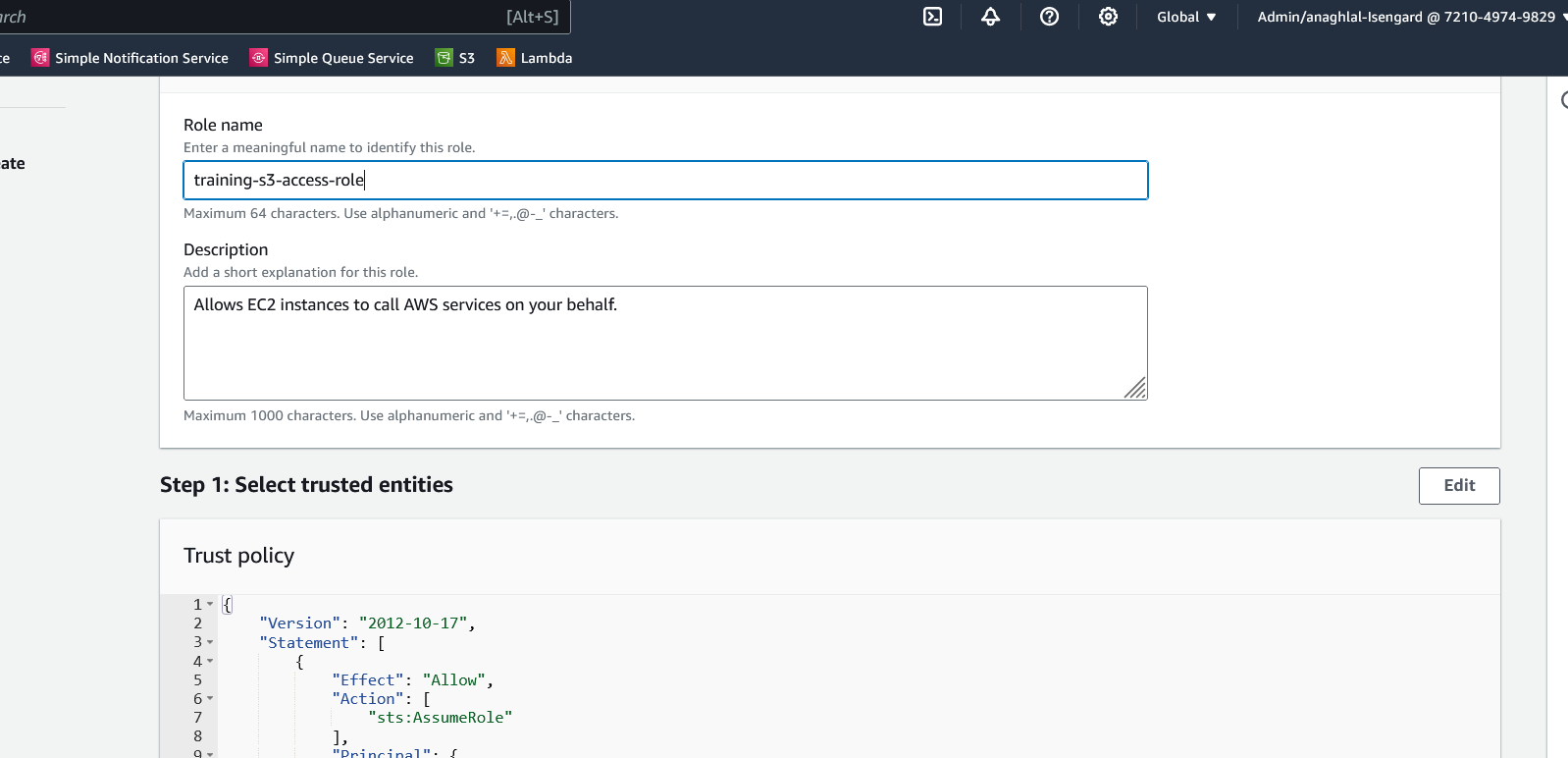




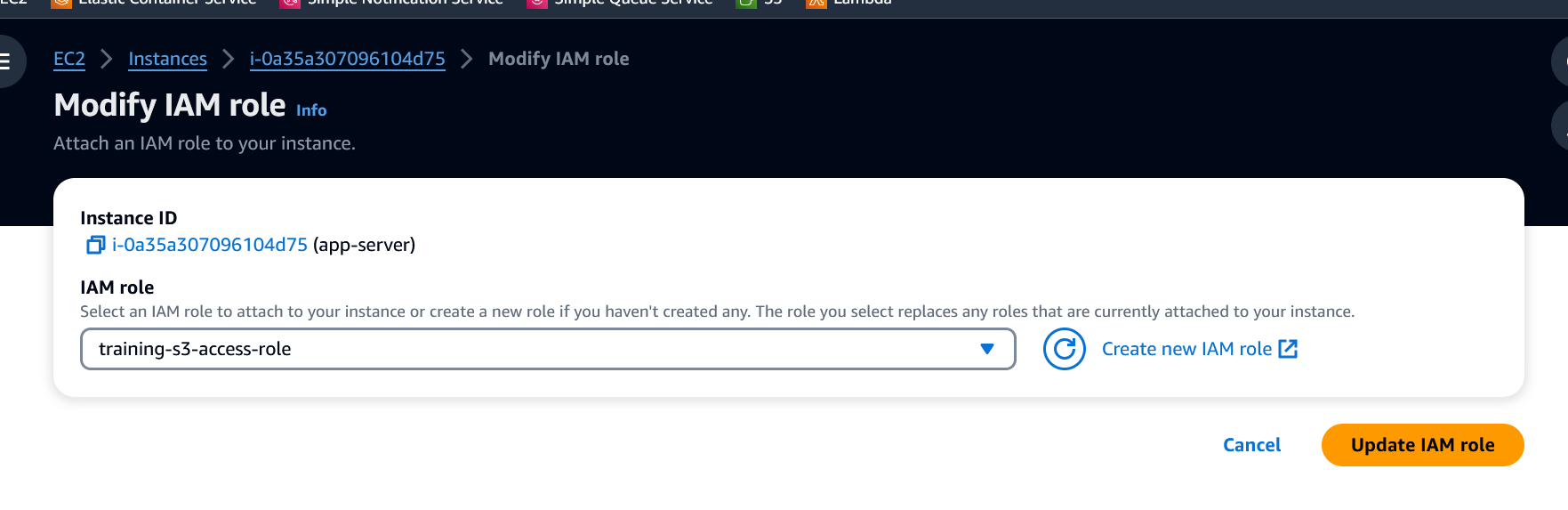


Add policy to access secrets manager:





Click create role



Start the instance from console

Once instance is in running state

Connect to it as above as ec2-user using Session manager.

### Run application and access it

cd /home/ec2-user/lab/workshop/intro/target

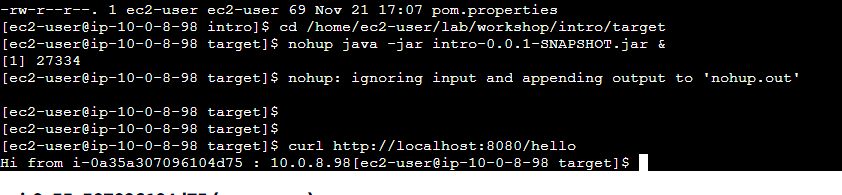
nohup java -jar intro-0.0.1-SNAPSHOT.jar &

curl <http://localhost:8080/hello>

to check process is running :

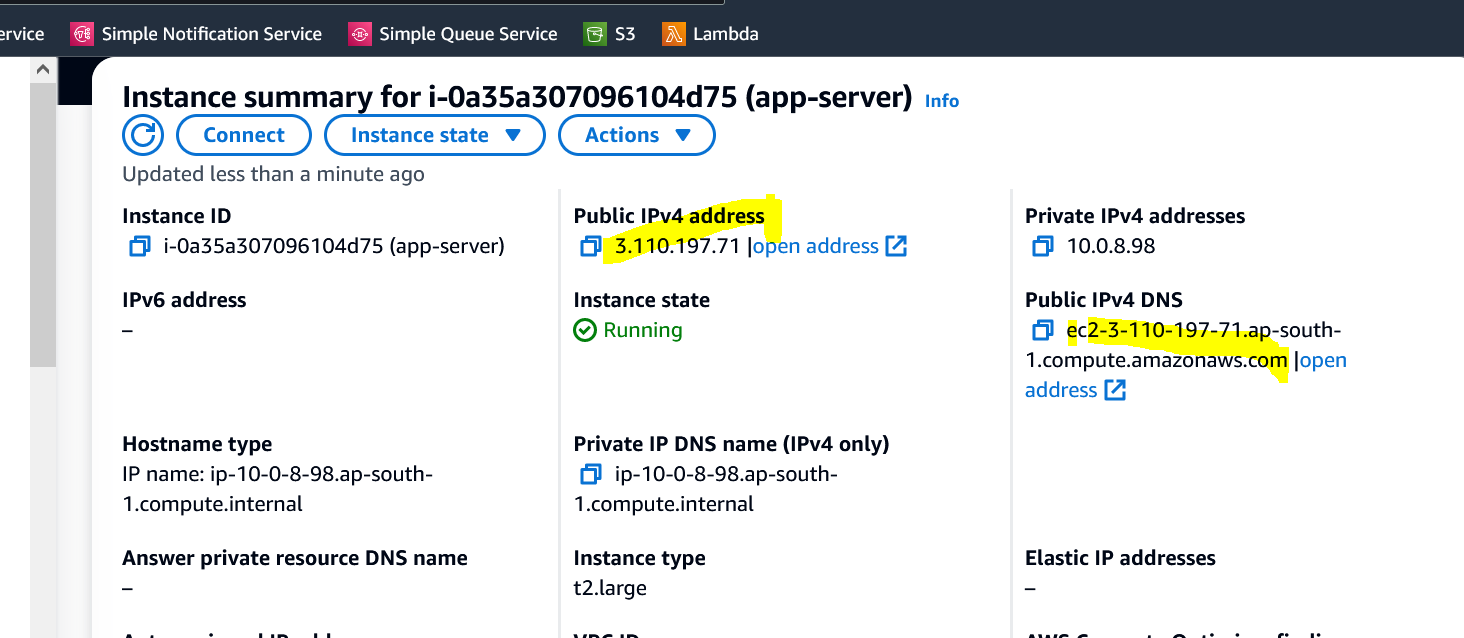
ps -ef | grep java

logs are at /home/ec2-user/lab/workshop/intro/target/nohup.out



Access from browser:

Get the DNS address of host



From the browser try

http://<DNS hostname>:8080/hello

http://<DNS hostname>:8080/employees

http://<DNS hostname>:8080/reads3?bucket=<bucket-name>&key=<file-name>

example

http://<DNS hostname>:8080/reads3?bucket=myfirstbucket-anagh&key=dummy.txt

Application logs are at:

/home/ec2-user/lab/workshop/intro/target/nohup.log

### Steps to create application machine image (AMI) and start application on booting

Only if you did not set the properties already :

cd /home/ec2-user/lab/workshop/intro/

cd src/main/resources/

vi application.properties

Set the following properties

s3.bucket.name=x

s3.bucket.object.key=y

rds.jdbc.url=<>

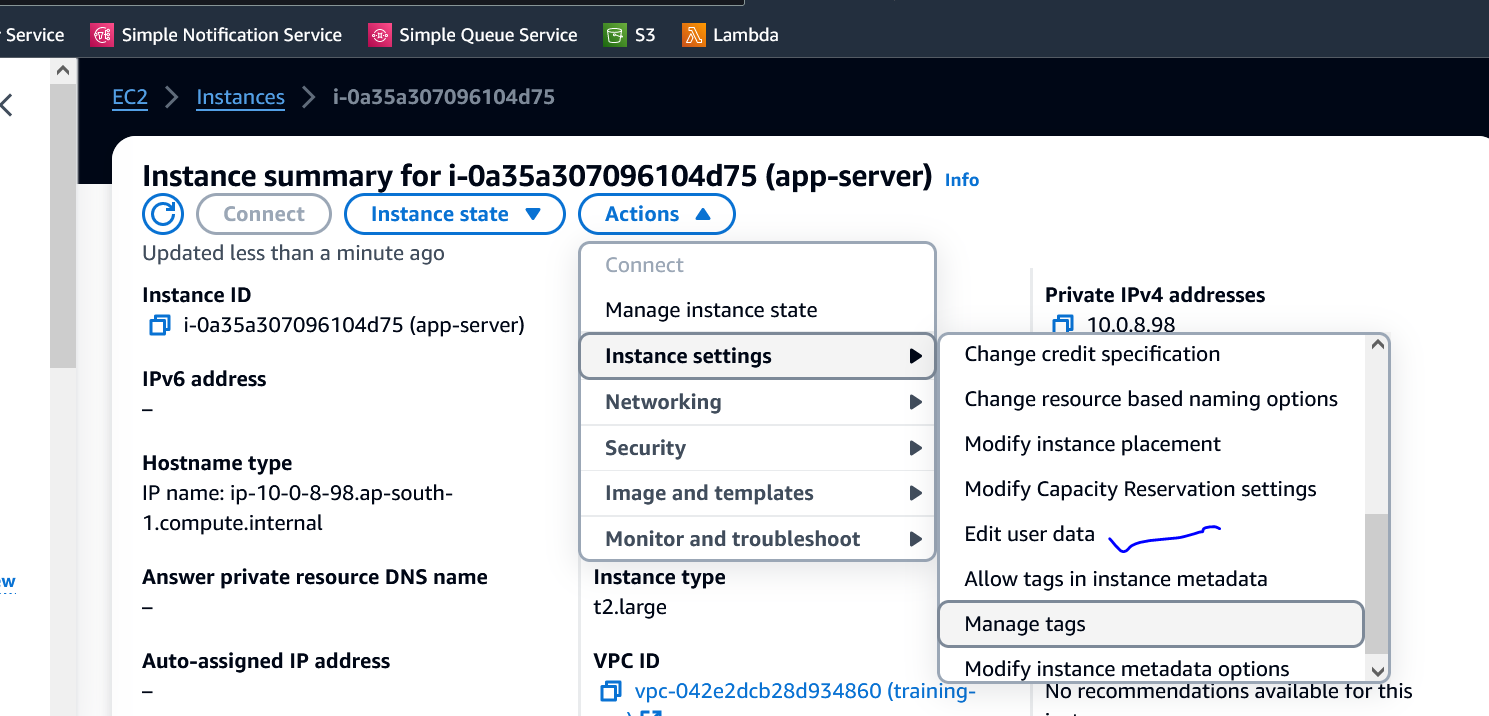
rds.secret.name=dbsecret

cd /home/ec2-user/lab/workshop/intro/

mvn install

ls -lrt target/\*

##### Add a start-up script



Paste the following contents in the user-data section:

Content-Type: multipart/mixed; boundary="//"

MIME-Version: 1.0

--//

Content-Type: text/cloud-config; charset="us-ascii"

MIME-Version: 1.0

Content-Transfer-Encoding: 7bit

Content-Disposition: attachment; filename="cloud-config.txt"

#cloud-config

cloud\_final\_modules:

- [scripts-user, always]

--//

Content-Type: text/x-shellscript; charset="us-ascii"

MIME-Version: 1.0

Content-Transfer-Encoding: 7bit

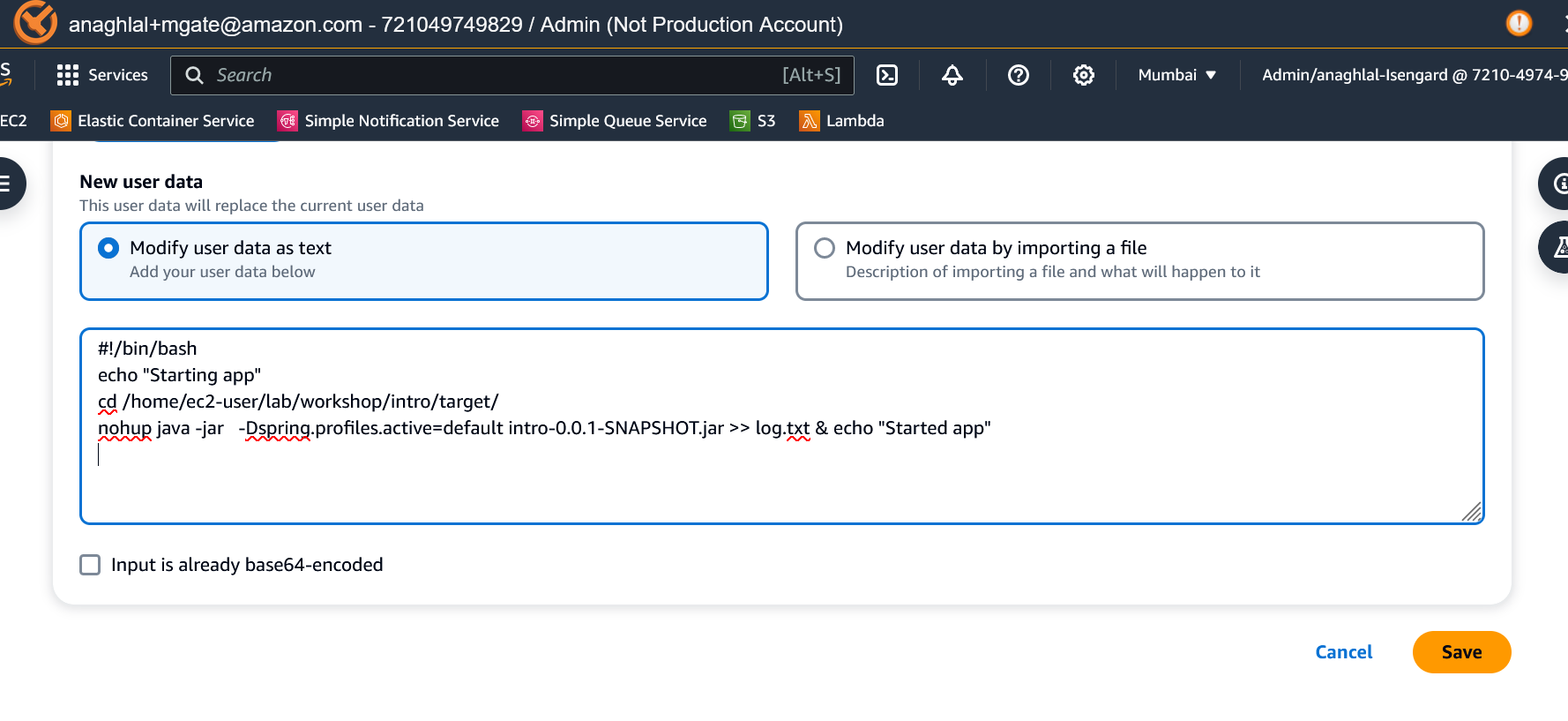
Content-Disposition: attachment; filename="userdata.txt"

#!/bin/bash

echo "Starting app"

cd /home/ec2-user/lab/workshop/intro/target/

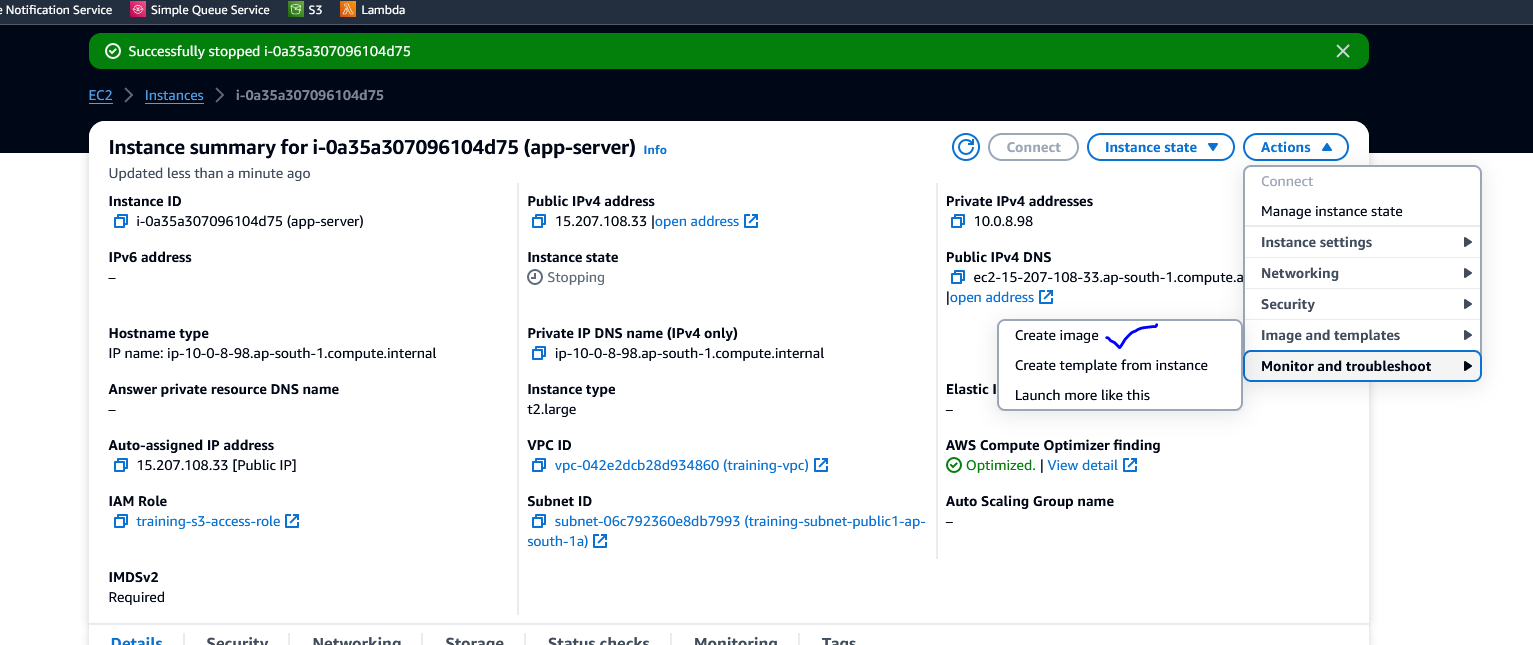
nohup java -jar -Dspring.profiles.active=default intro-0.0.1-SNAPSHOT.jar >> log.txt & echo "Started app"



Restart the server.

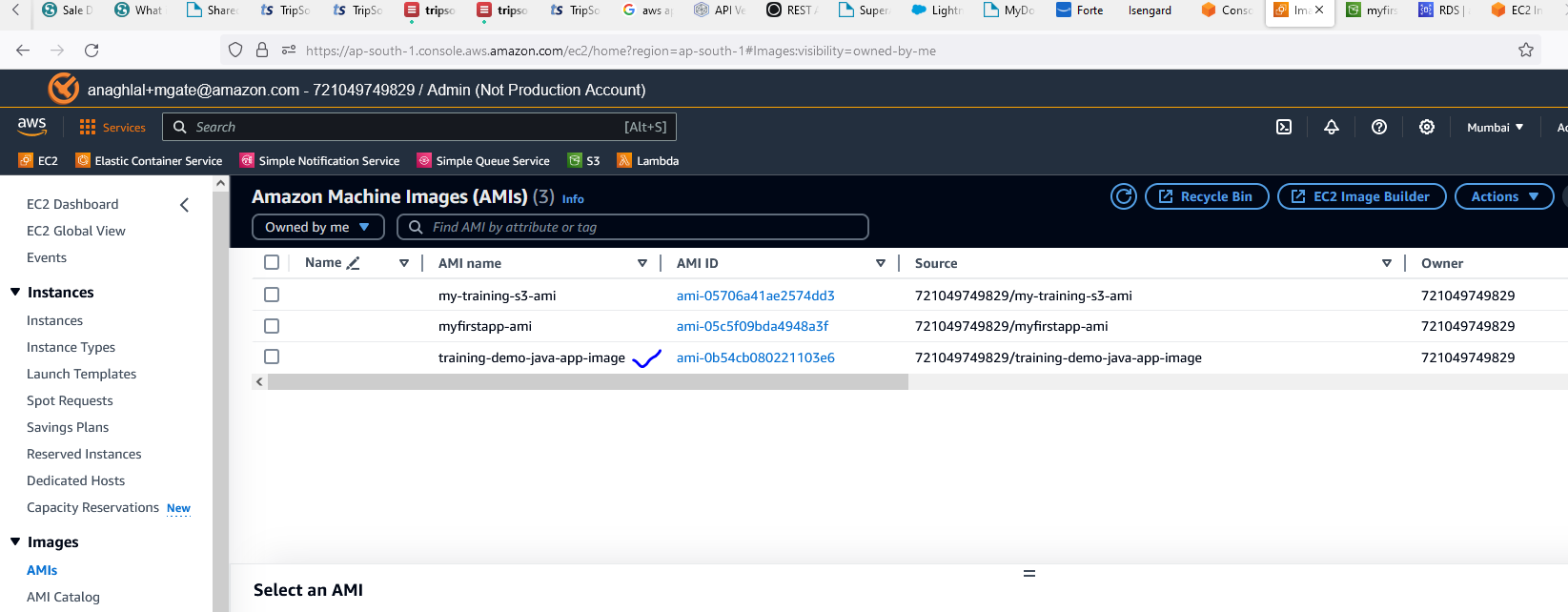
#### Create an AMI

Stop instance



Give the image a name and select defaults for the rest and hit Create Image

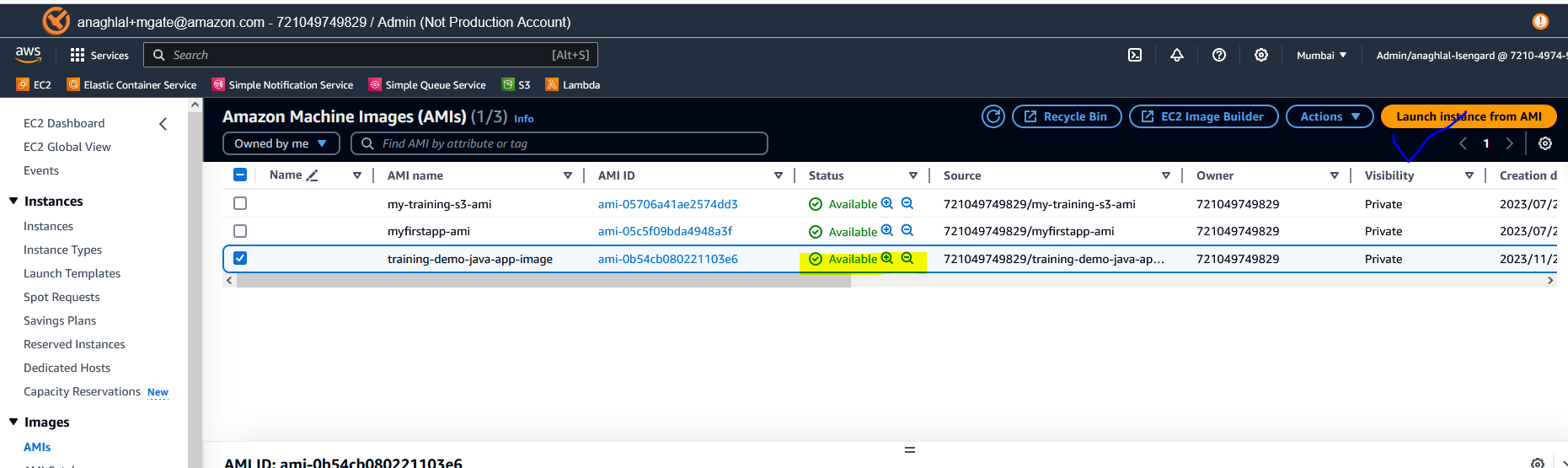
(training-demo-java-app-image)



The image will be in pending state initially. Wait for it to move to Available state and move to next step.

#### Launch the AMI

Select the AMI in available status and Launch instance from AMI

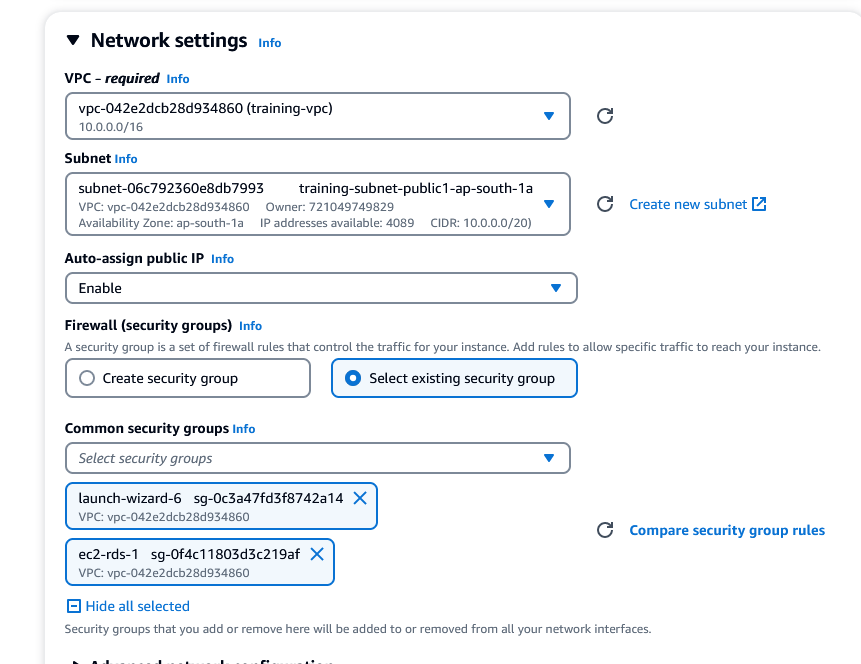


Instance type : t2.xlarge

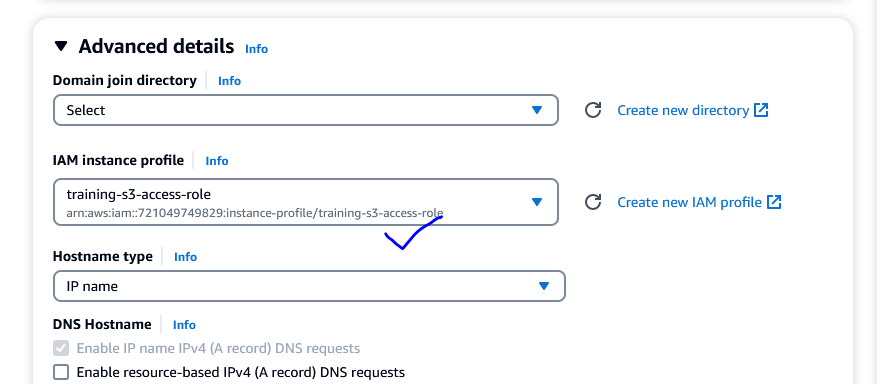
Select key-pair login

Network setting click Edit. Select your VPC and enable public IP.

1. VPC
2. Public subnet
3. Enable public Ip
4. Select security groups same as original instance



Select IAM role we created earlier



Edit UserData to define launch script of application



Content-Type: multipart/mixed; boundary="//"

MIME-Version: 1.0

--//

Content-Type: text/cloud-config; charset="us-ascii"

MIME-Version: 1.0

Content-Transfer-Encoding: 7bit

Content-Disposition: attachment; filename="cloud-config.txt"

#cloud-config

cloud\_final\_modules:

- [scripts-user, always]

--//

Content-Type: text/x-shellscript; charset="us-ascii"

MIME-Version: 1.0

Content-Transfer-Encoding: 7bit

Content-Disposition: attachment; filename="userdata.txt"

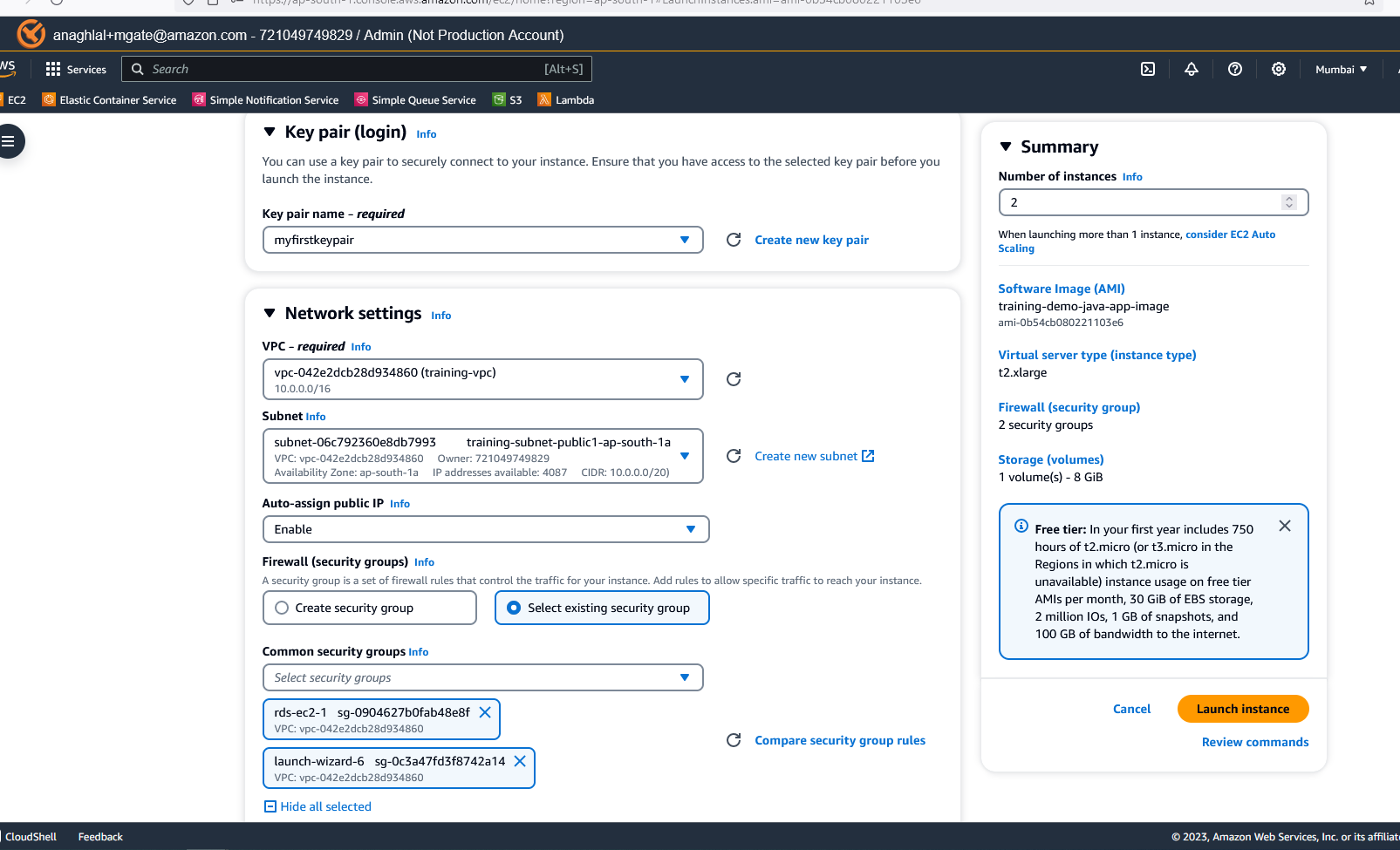
#!/bin/bash

echo "Starting app"

cd /home/ec2-user/lab/workshop/intro/target/

nohup java -jar -Dspring.profiles.active=default intro-0.0.1-SNAPSHOT.jar >> log.txt & echo "Started app"

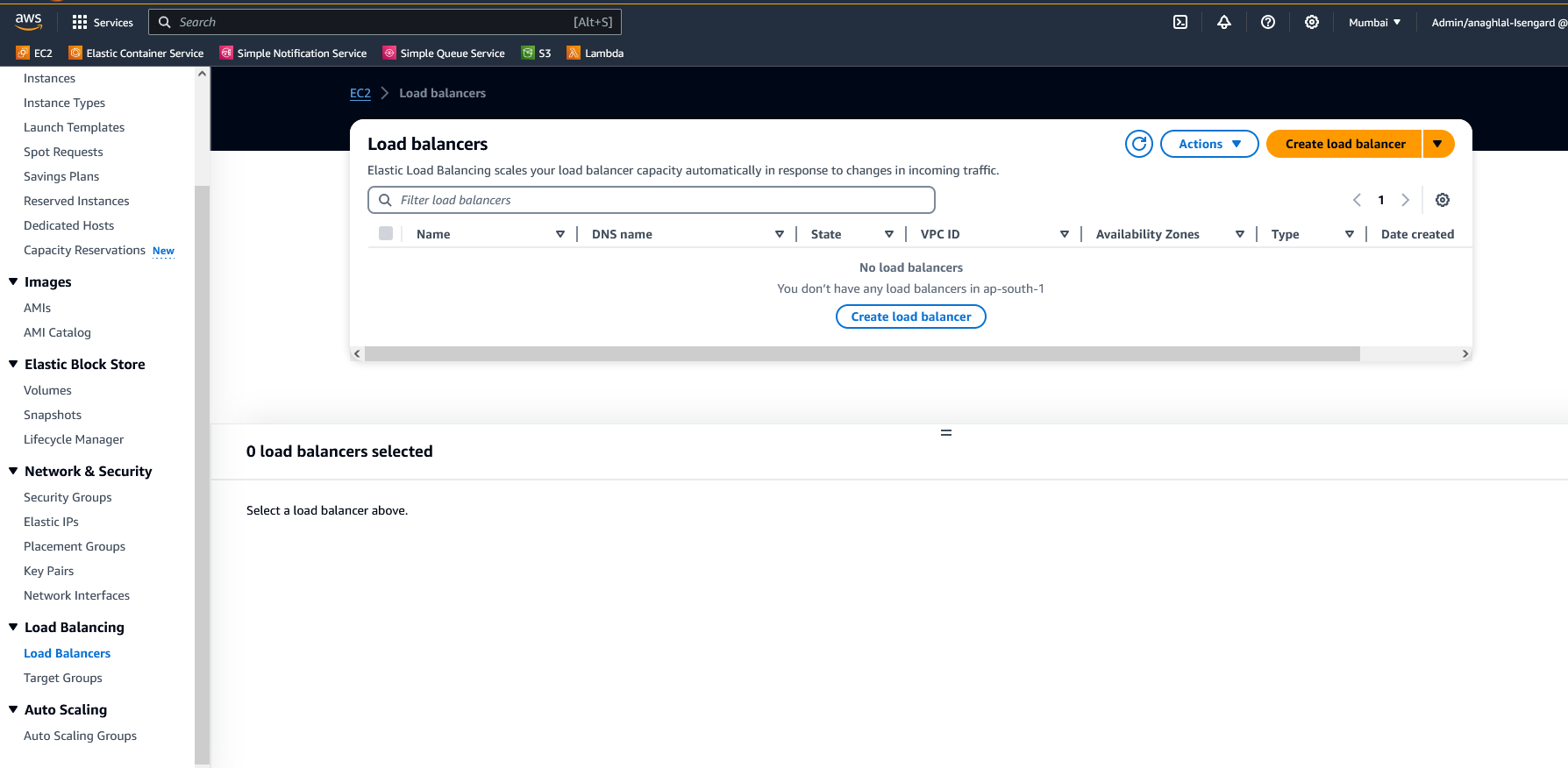
Select two instances to launch

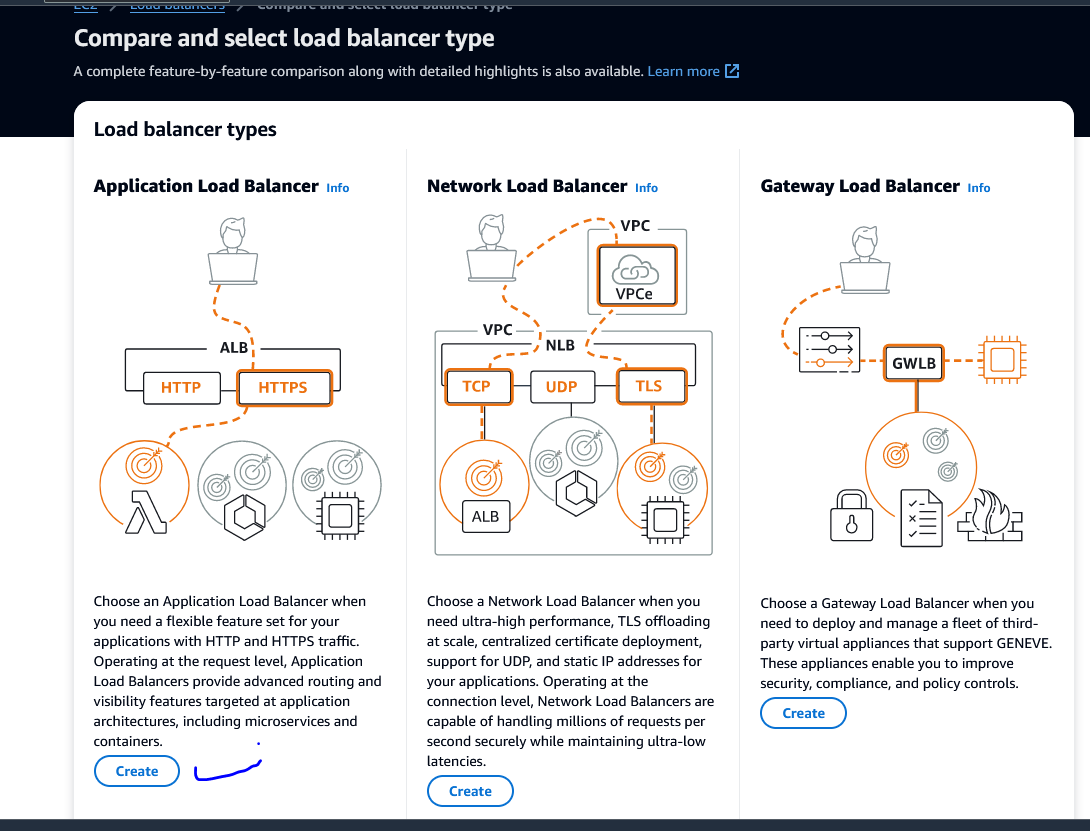




Try out the URLs with the new servers.

### Load balancer to route traffic to all instances launched [Optional]

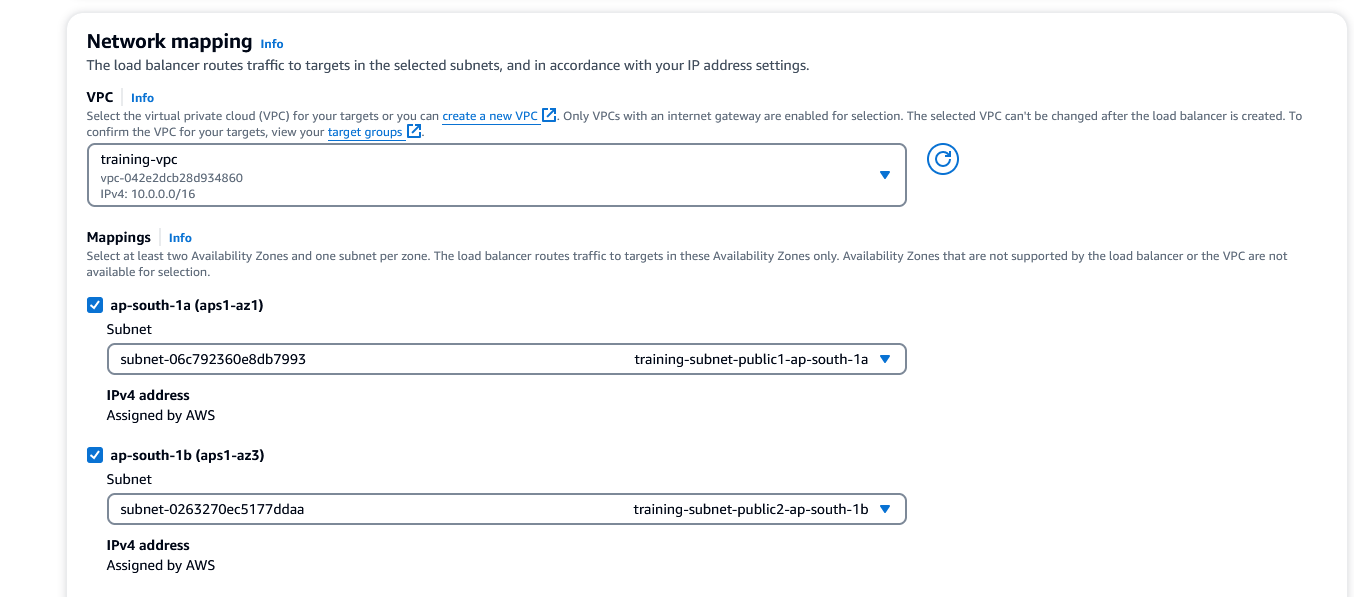




Internet Facing

IPV4

Network-mapping : Select VPC



## Feedback

Please fill out the survey link to help us in getting better.