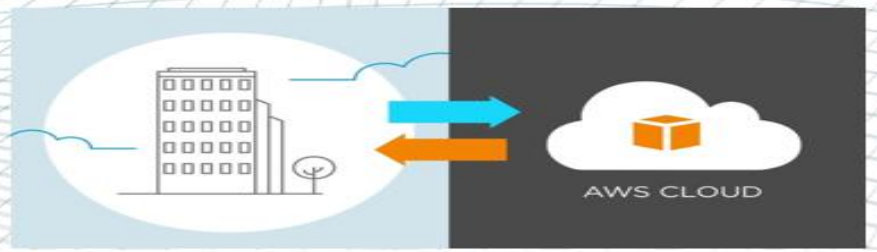
CAPSTONE PROJECT

By – Poonam Sharma
PGCP CC Batch - 12

Capstone Project



By
Poonam Sharma
(PGCP Batch-12)

Application Migration to Cloud Platform

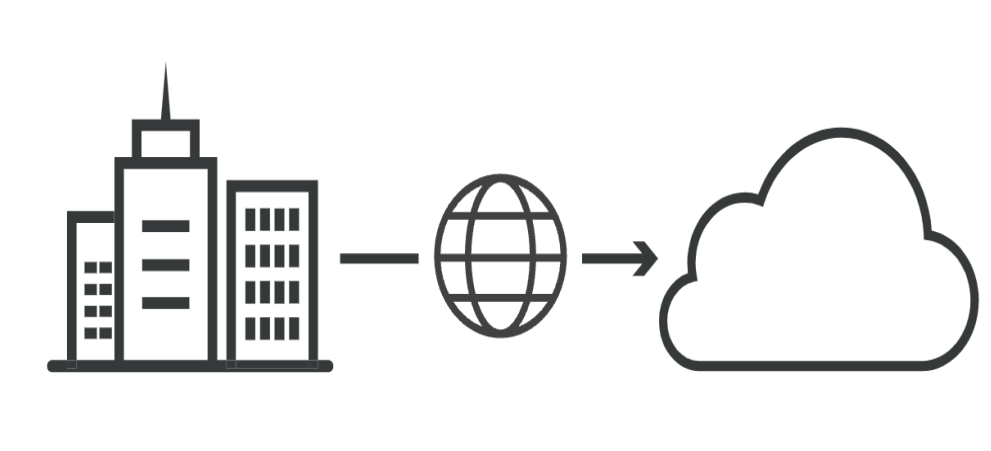
28 September 2021

Submitted to

Manipal Academy of Higher Education & AWS Educate

Application Migration to Cloud Platform

INTRODUCTION



Cloud migration is the process of moving digital business operations into the cloud. Cloud migration is sort of like a physical move, except it involves moving data, applications, and IT processes from some data centres to other data centres, instead of packing up and moving physical goods. Much like a move from a smaller office to a larger one, cloud migration requires quite a lot of preparation and advance work, but usually it ends up being worth the effort, resulting in cost savings and greater flexibility.

About the Organisation

URBANSLOTH is an up-coming unicorn in food delivery apps in India. Due to their ML engines being able to predict on-time delivery of food and service quality they were able to penetrate the Indian market in an almost no time.

Right now, **URBANSLOTH** holds the 3rd position among the top online food delivery apps in India. With the plan in place to expand their presence amongst other market in South Asia with global launch planned for coming years. Headed by Aryan Sethi graduate of UOP IT department along with his infrastructure team headed by Damini Pushkar who has been key person in designing infrastructure for hosting the application on the third party rented infrastructure.

Project Objective:

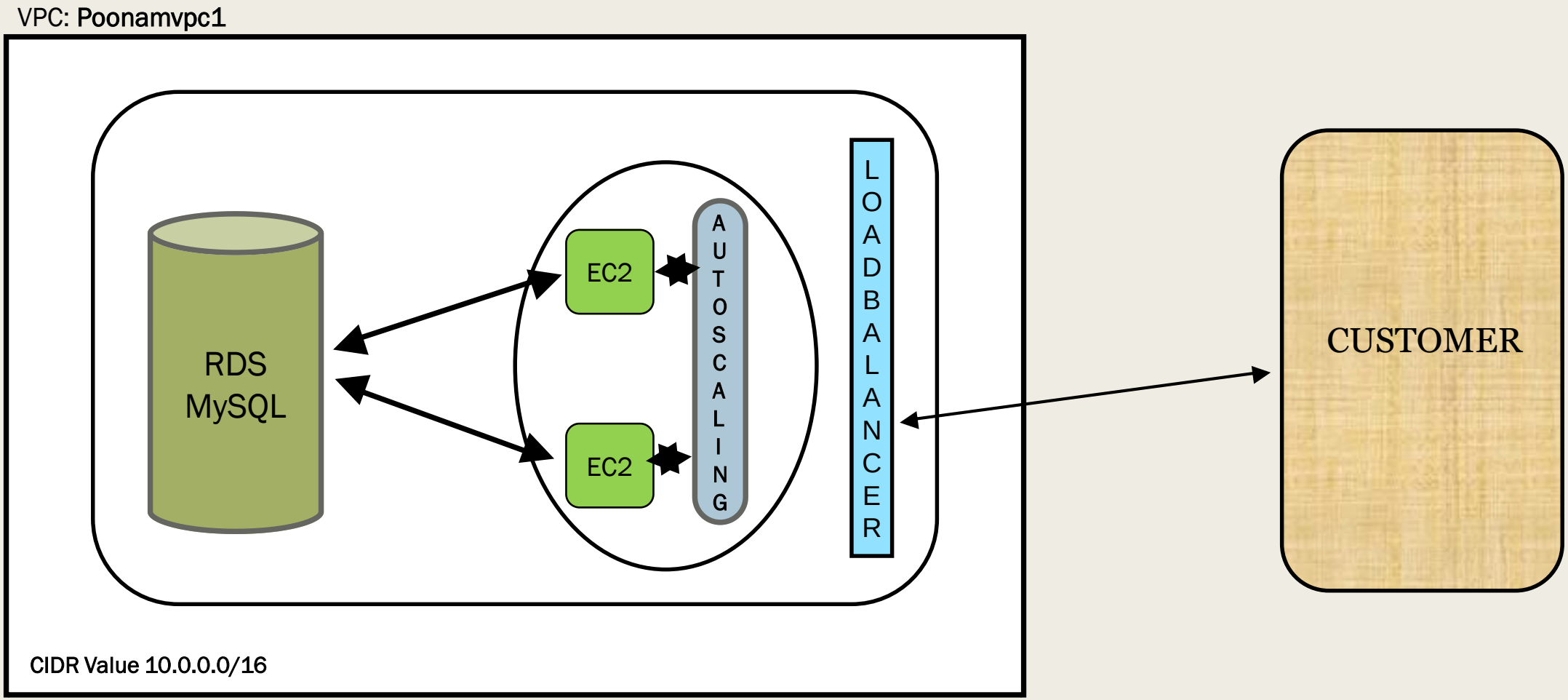
To migrate the on-premise application to the public cloud thereby reducing the load on their application

Understanding of Requirements:

Due to unexpected increase in the load on their application, the infrastructure team of **URBANSLOTH** has started exploring the options to migrate their application to the Public Cloud.

Damini has been tasked with setting up the POC (Proof of Concept) to convince the management that their application is suitable for the cloud and any application can be easily migrated on the cloud without losing any data.

Pictorial Representation:



Tasks Involved:

Below are the high-level task break down for the POC to migrate the application on to the cloud

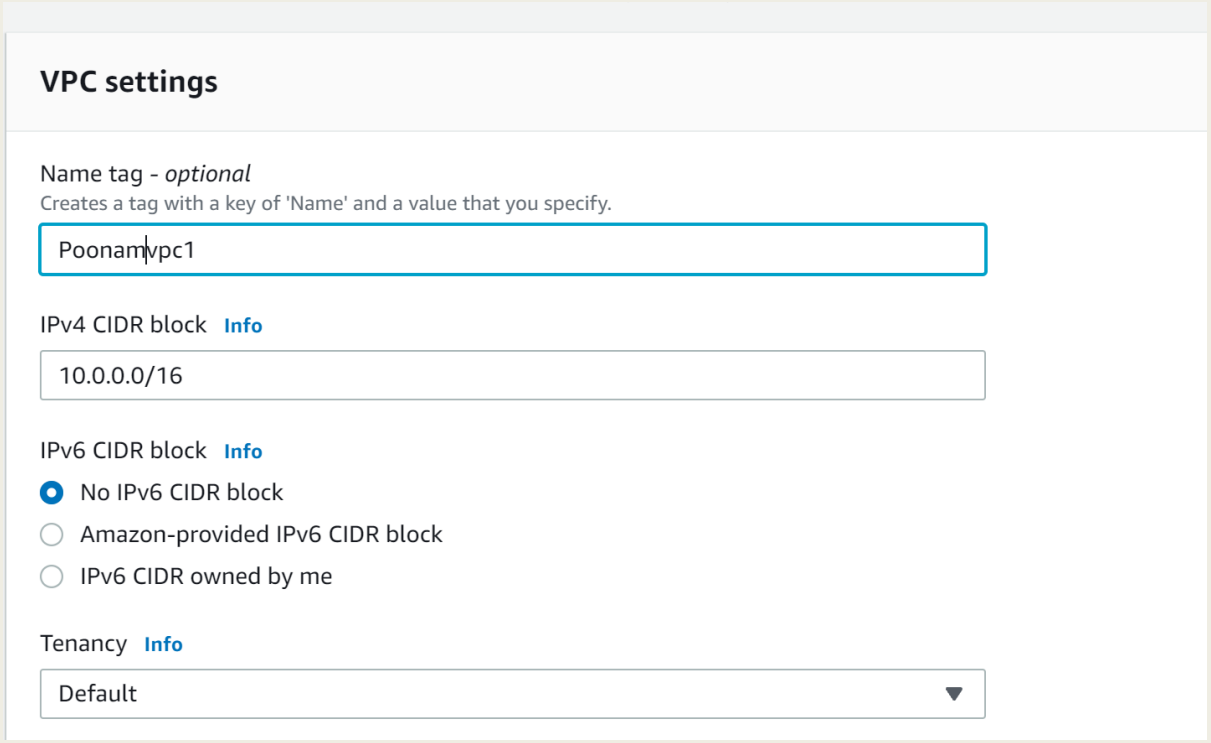
- Migrate the on-premise database onto the database server running on the cloud making sure of secure access to it. That is, for instance it need not be accessible from outside the VPC and should be accessible only from the EC2 instance where the Python application is installed.
- On an EC2 instance, replicate all the dependencies and application libraries and configuration from the on-premise server where the Python application is set up.
- Configure the application to run on normal http port instead of testing & development port 5000.
- Set up auto-scaling with load balancer for the above application configuration.

Project Implementation:

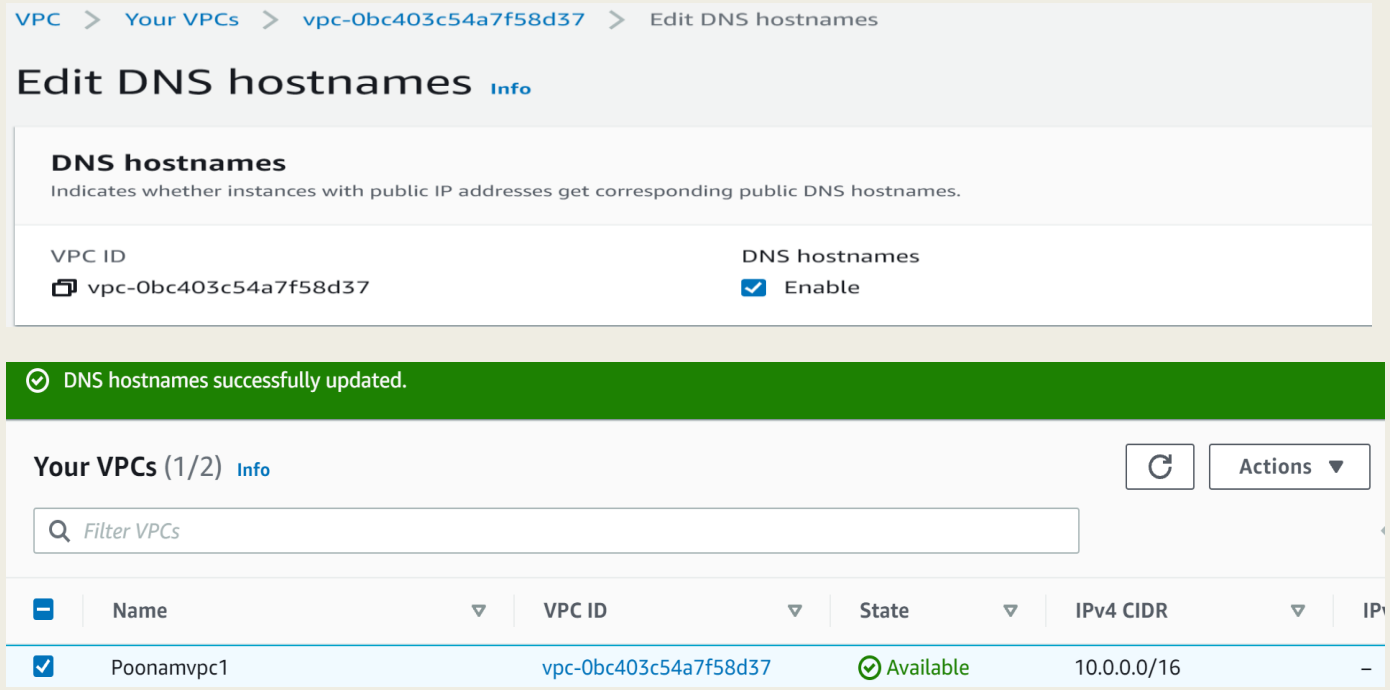
Creating VPC

Task 1 – Setting up a VPC in the Mumbai region with three subnets

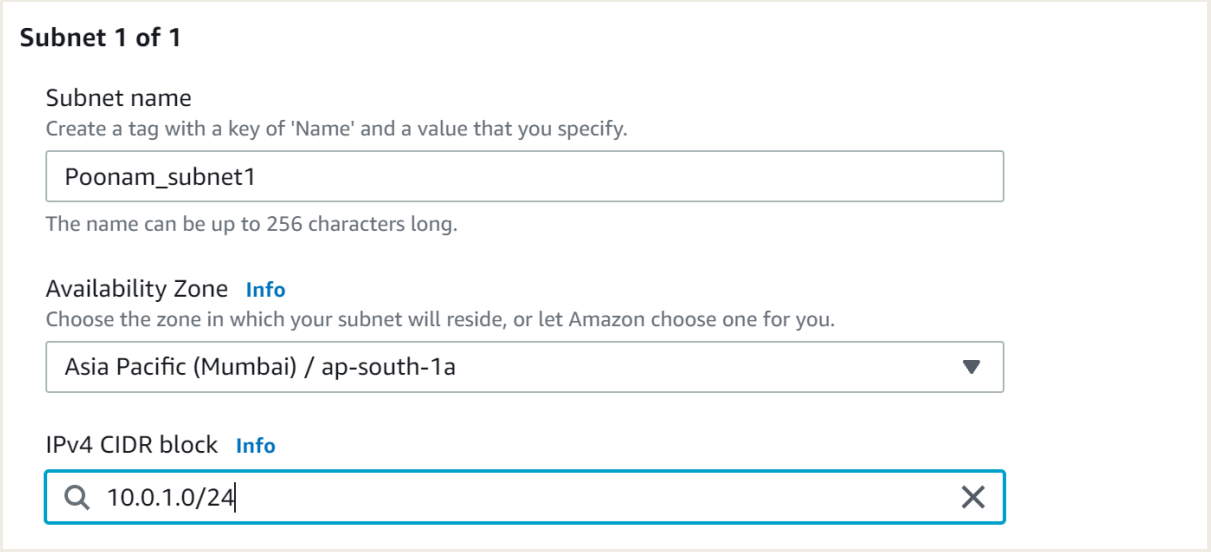
1. Creating VPC **Poonamvpc1**
2. Give CIDR value **10.0.0.0/16**



3. Click on “Edit **DNS Hostname” and** Select **Enable**

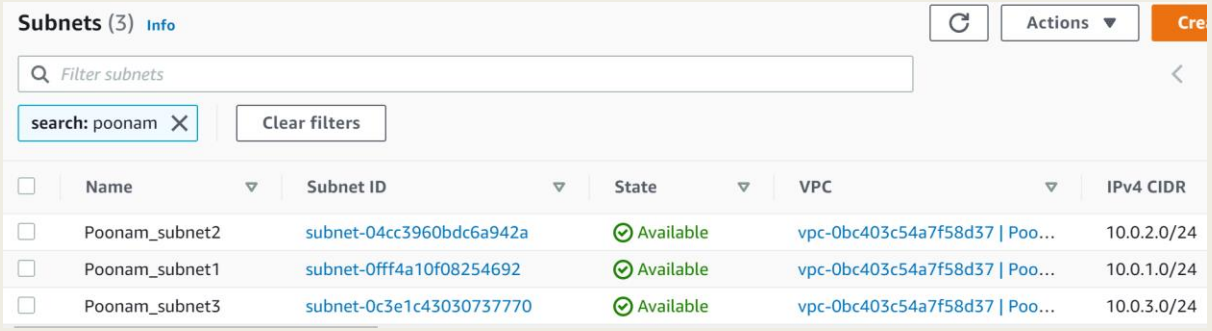


Create Subnets **Poonam_subnet1, Poonam_subnet2, Poonam_subnet3**

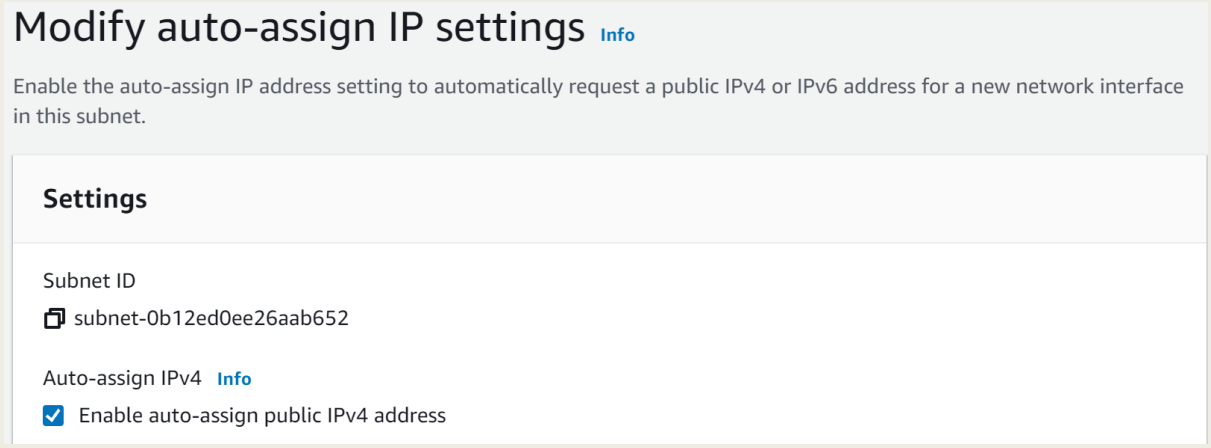


Similarly create 2 more subnets

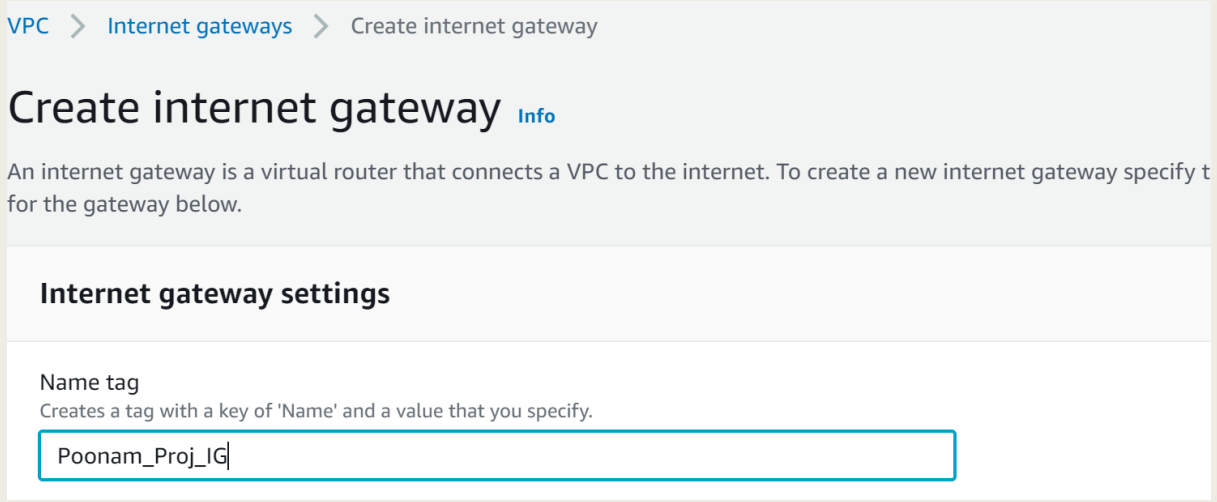
All three Subnets up and running



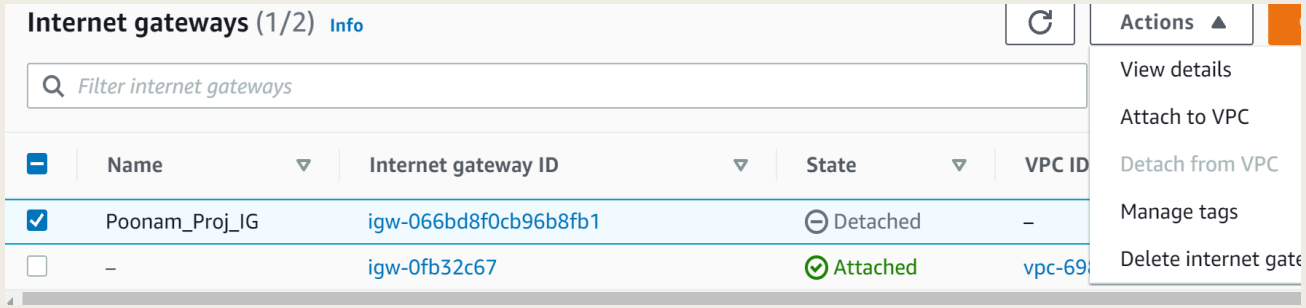
Select **Poonam_Subnet1** and **2**→ Actions → Modify Auto-Assign IP setting → **Enable**



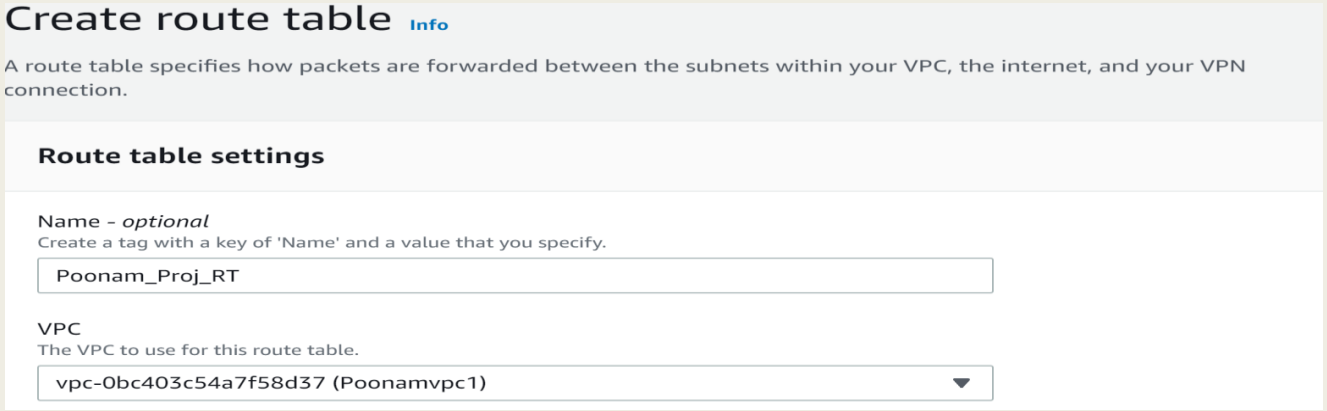
Now, Select **Internet Gateway** → Create Internet Gateway: **Poonam_Proj_IG**



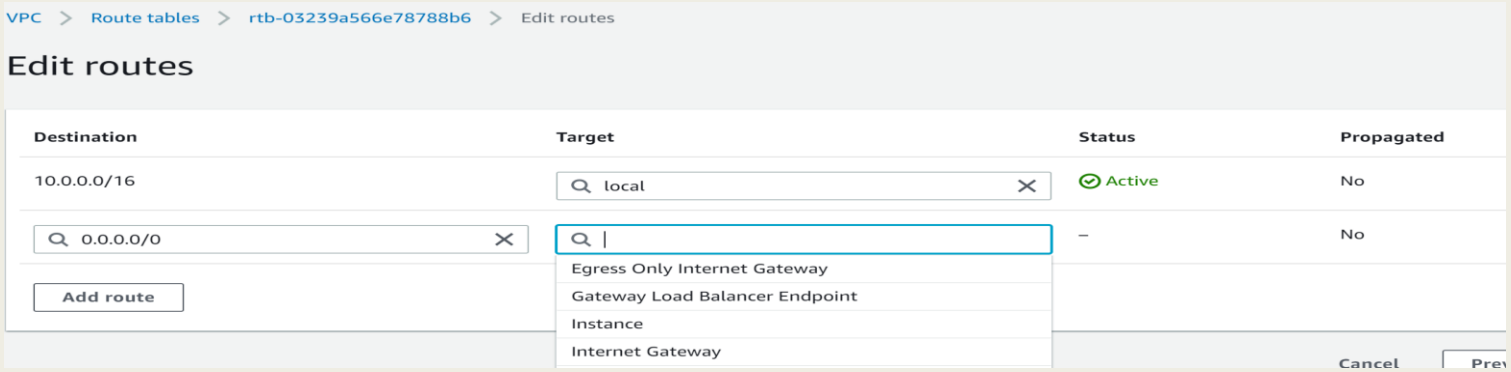
Attach Internet gateway to **Poonamvpc1**

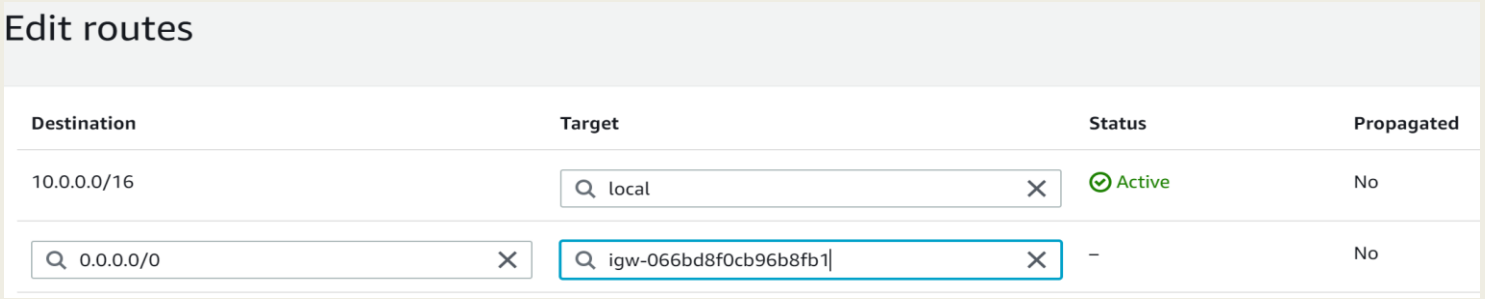


Create **Route Table: Poonam_Proj_RT**

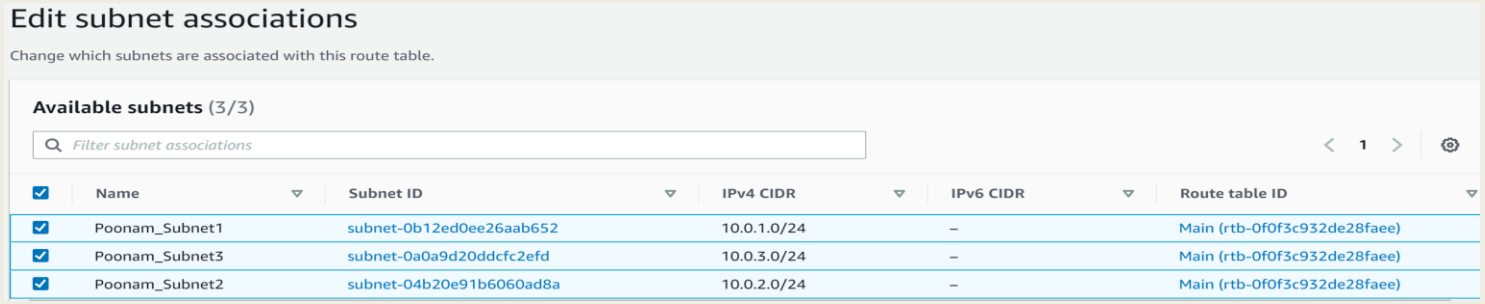


Now, click on **Edit Route,** and Add route **0.0.0.0/0** and select **Internet Gateway**





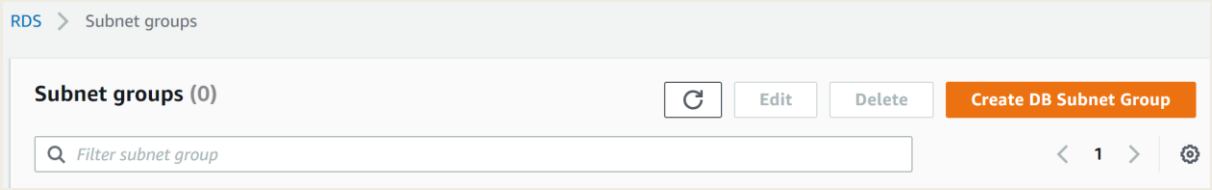
Click on **Edit Subnet Association** and Select all Subnets



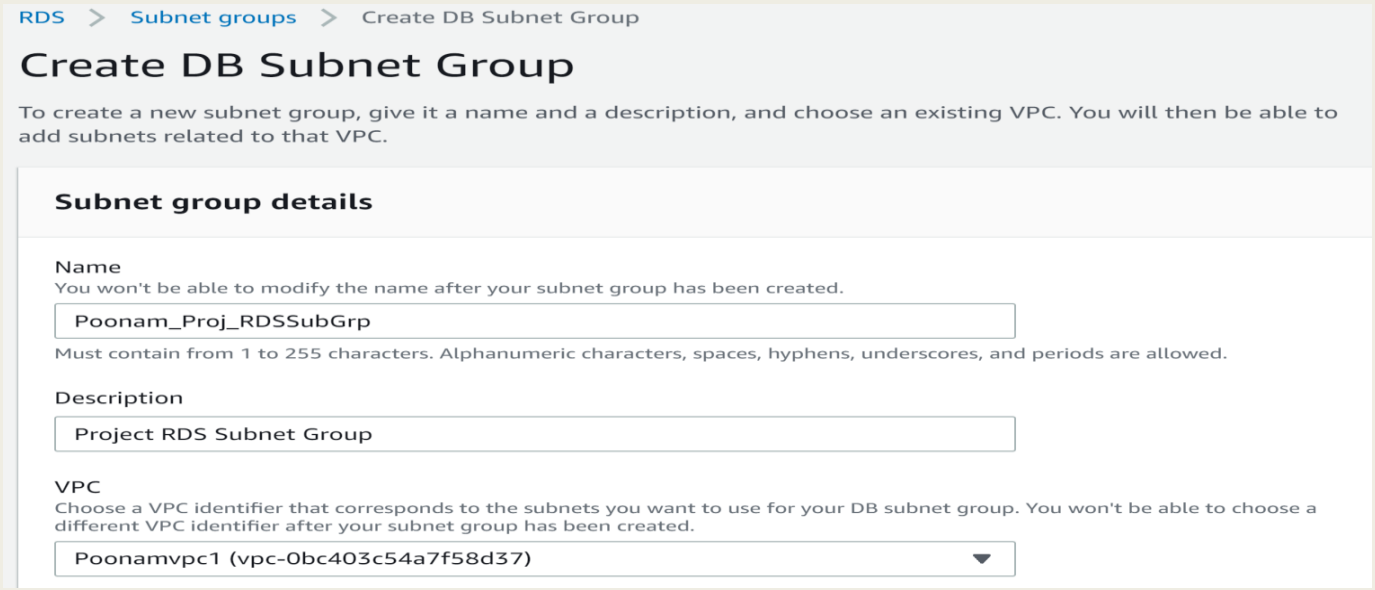
Setting up RDS Database

TASK 1: Create an RDS Subnet Group

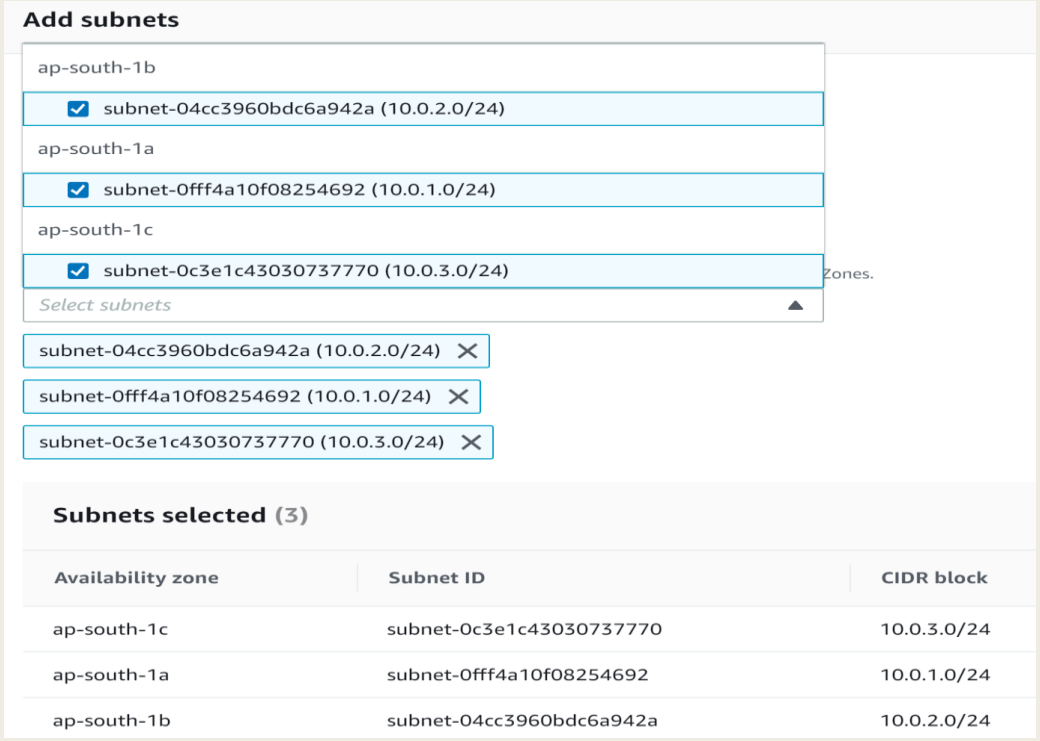
1. Open **Amazon RDS**
2. Click on **Create Subnet Groups**
3. Click on **Create DB Subnet Group**



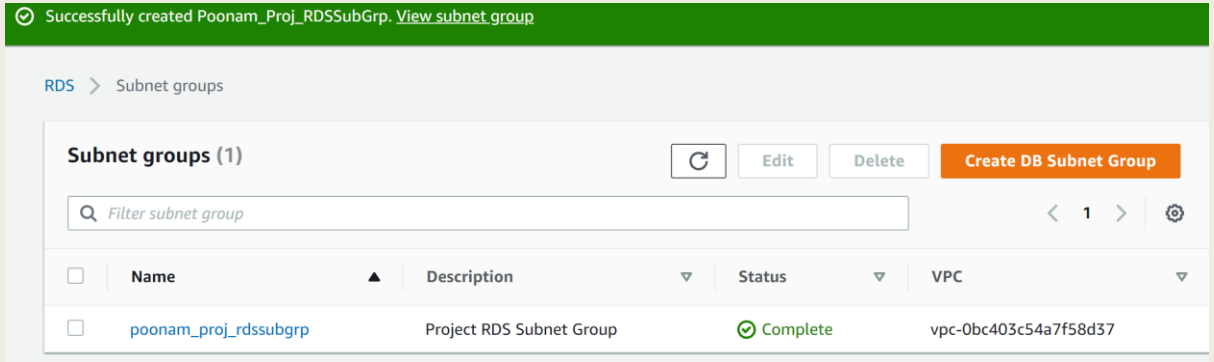
4. Name : **Poonam_Proj_RDSSubGrp**
5. Description: **Project RDS Subnet Group**
6. VPC: Poonam**vpc1**



5. **Add Availability Zones(A-Z) and all the Subnets**



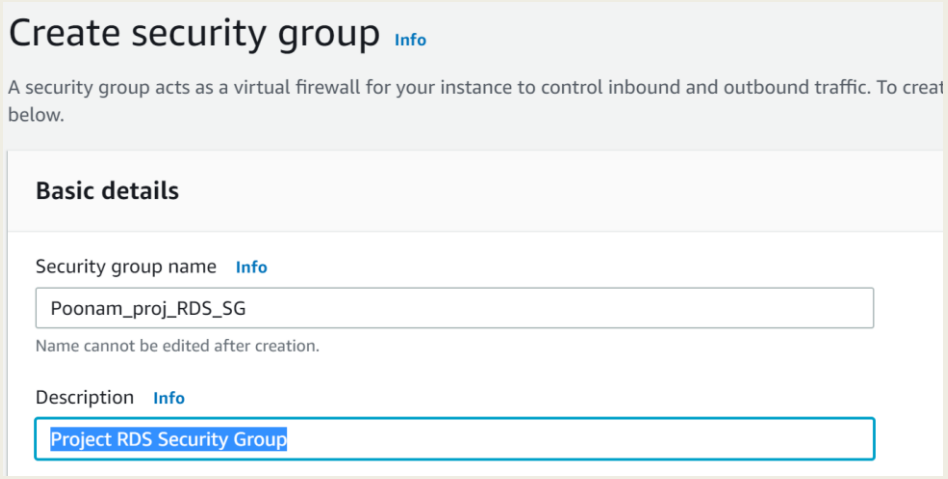
6. Successfully created RDS Subnet group **poonam_proj_rdssubgrp**



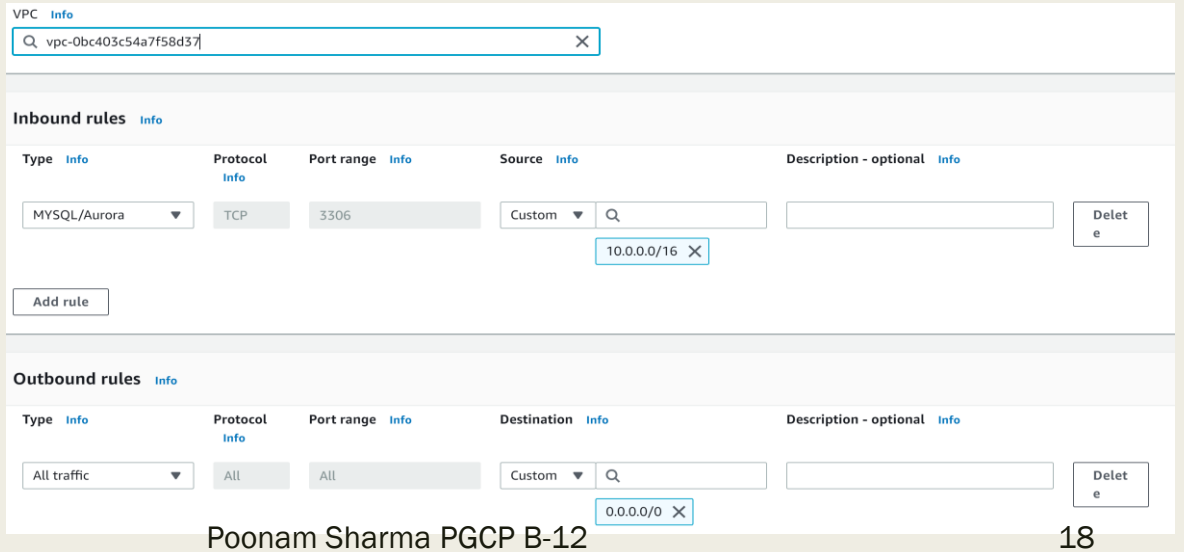
Task 2: Create an RDS Security Group

Use EC2 service → Select MySQL for the incoming traffic

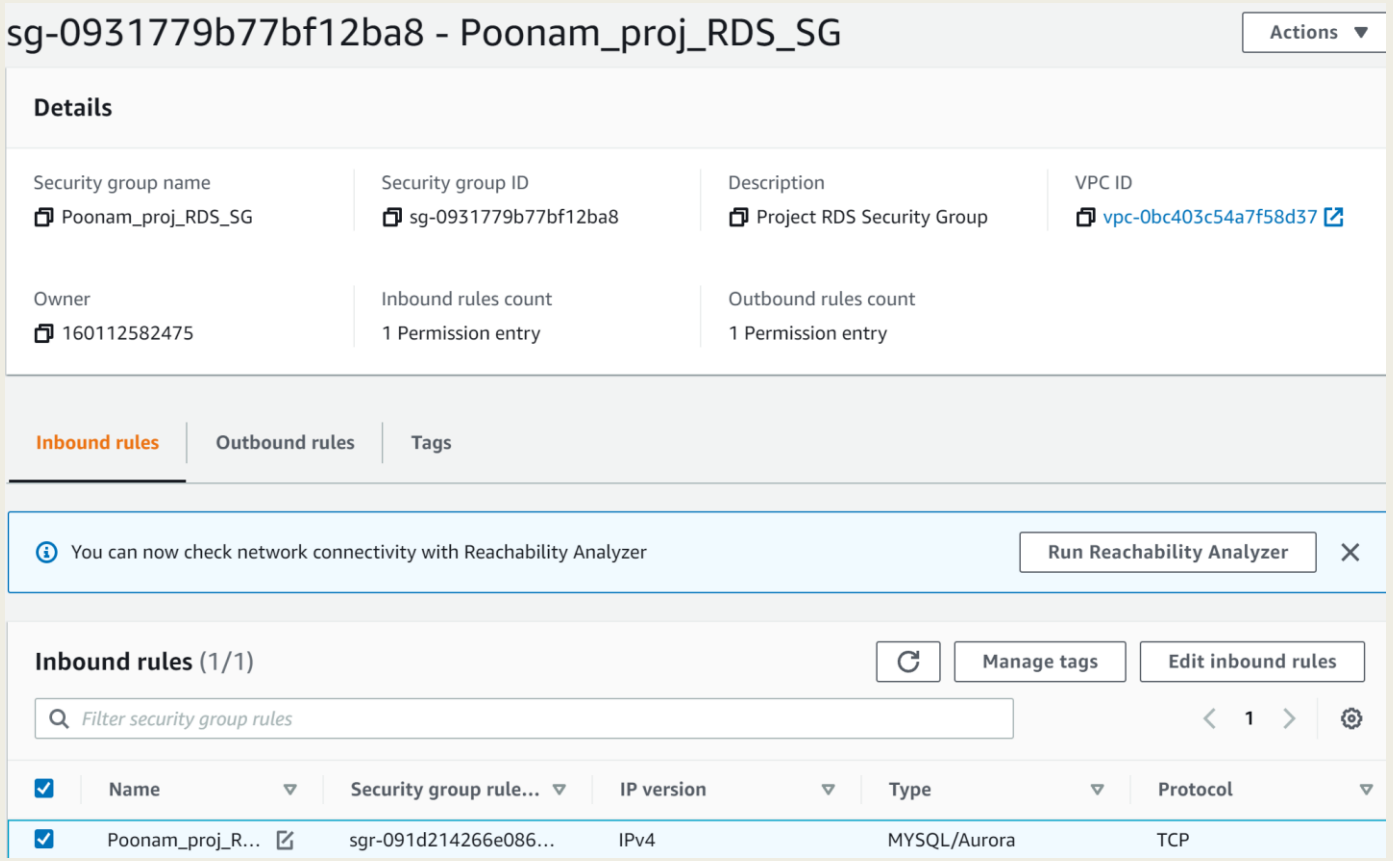
1. Create new Security Group
   Security Group Name: **Poonam_proj_RDS_SG**
   Description: **Project RDS Security Group**



2. Add Inbound rule
   Type: **MYSQL** and Custom it to our VPC IP address i.e.10.0.0.0/16

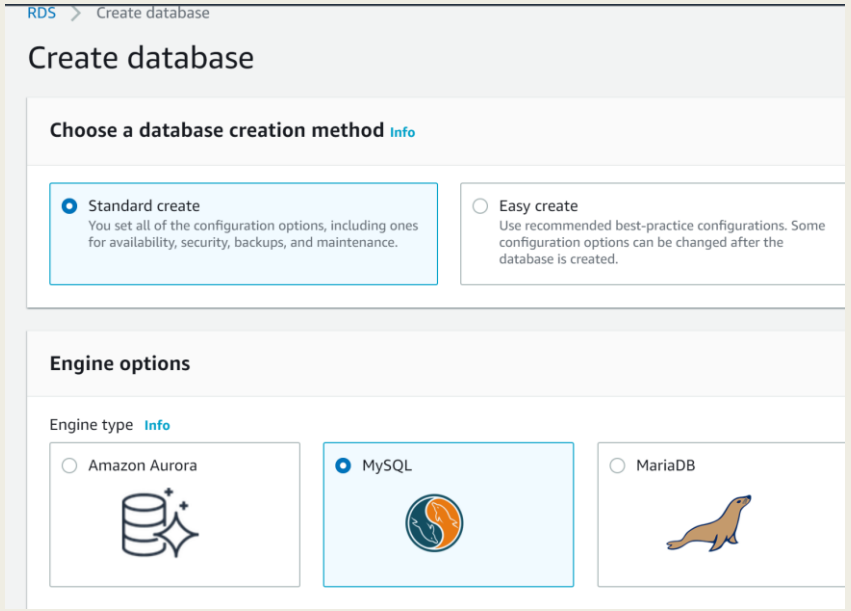


3. Security group **Poonam_proj_RDS_SG** is successfully created

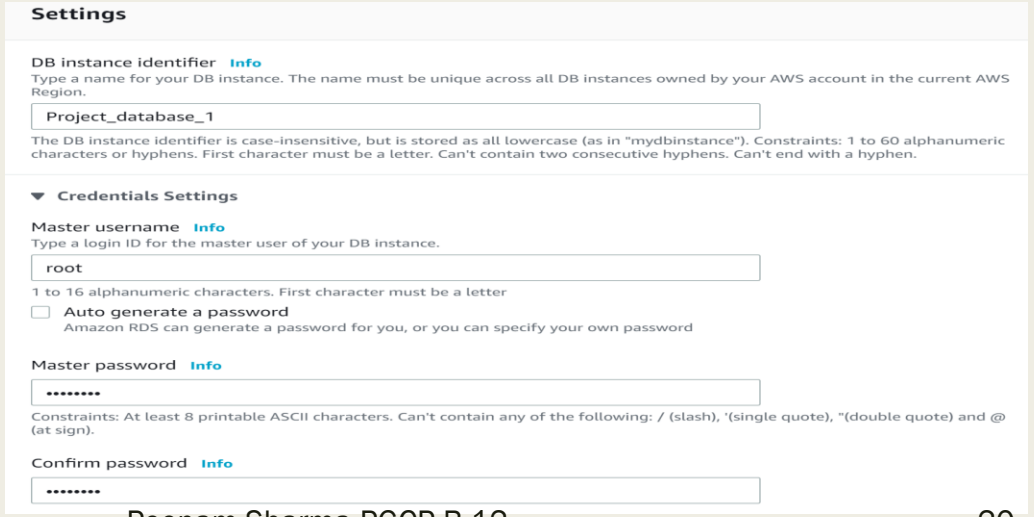


TASK 3: Create Database

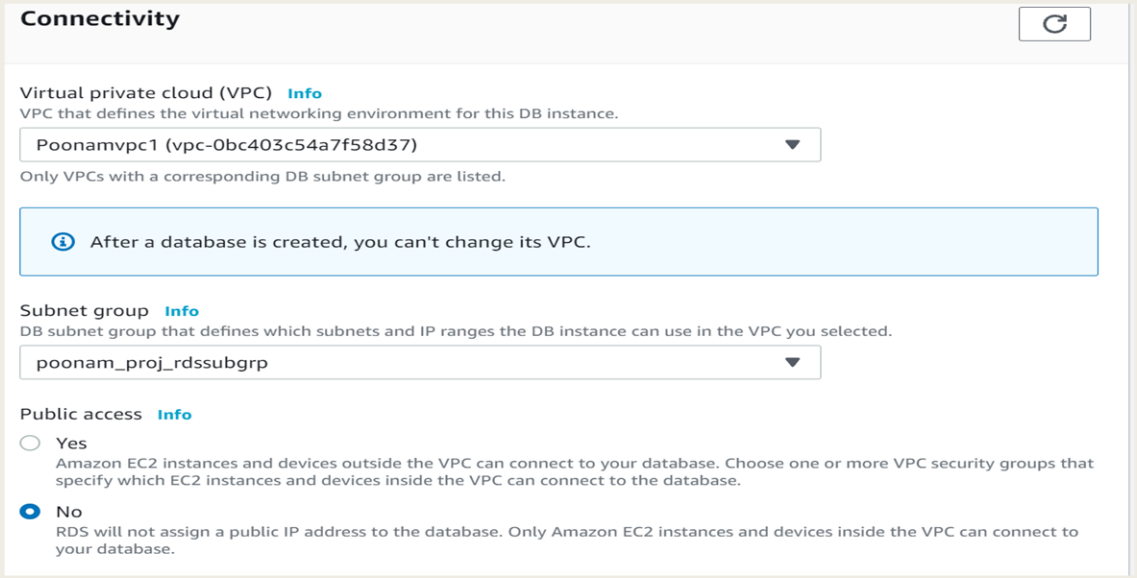
1. Click on **Create Database**
2. Select **Standard create** and then **MySQL Engine**



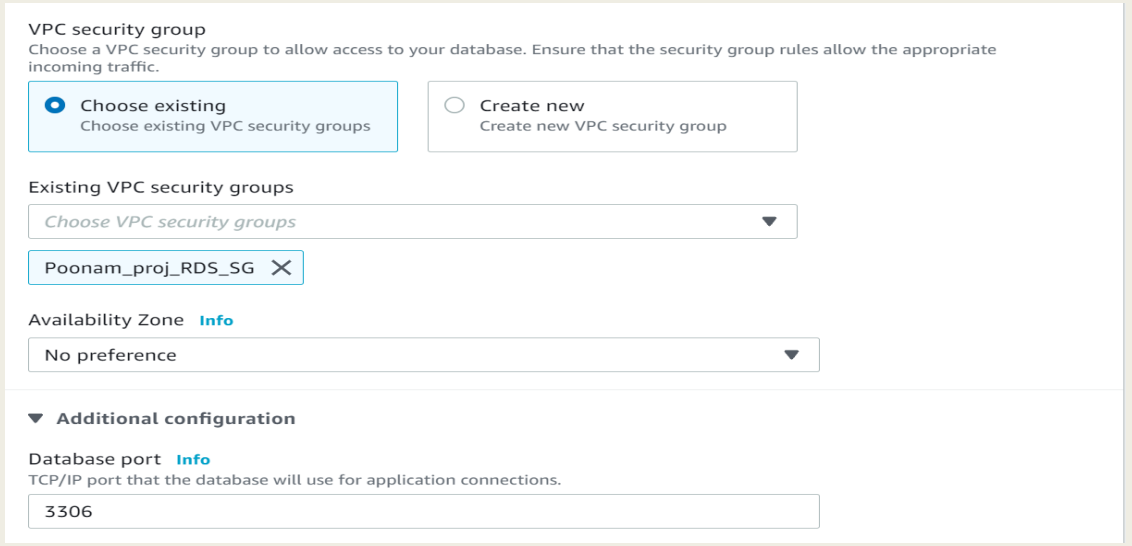
3. Database name: **PoonamProject-Database-1**
   Master Username: **root**
   Master Password: **poonam22**

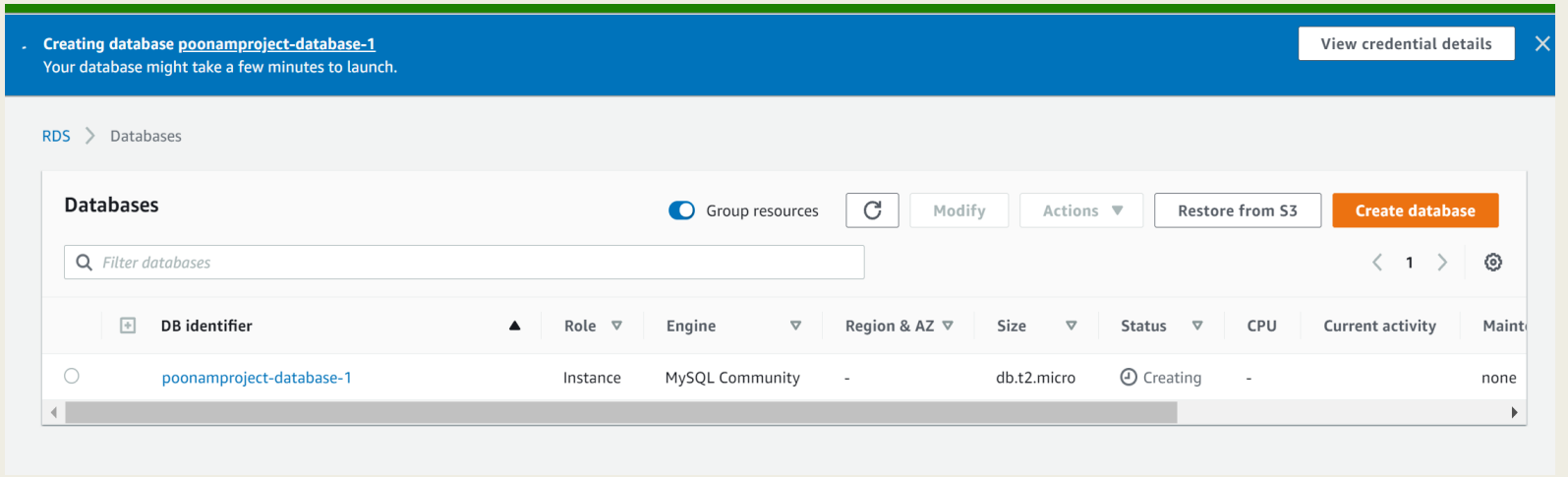


Select VPC and Subnet group

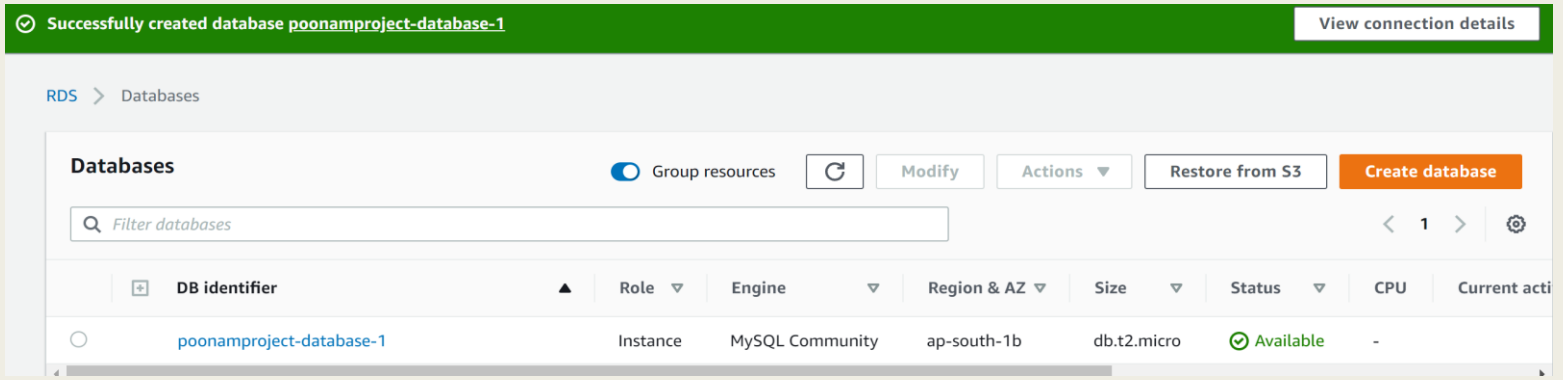


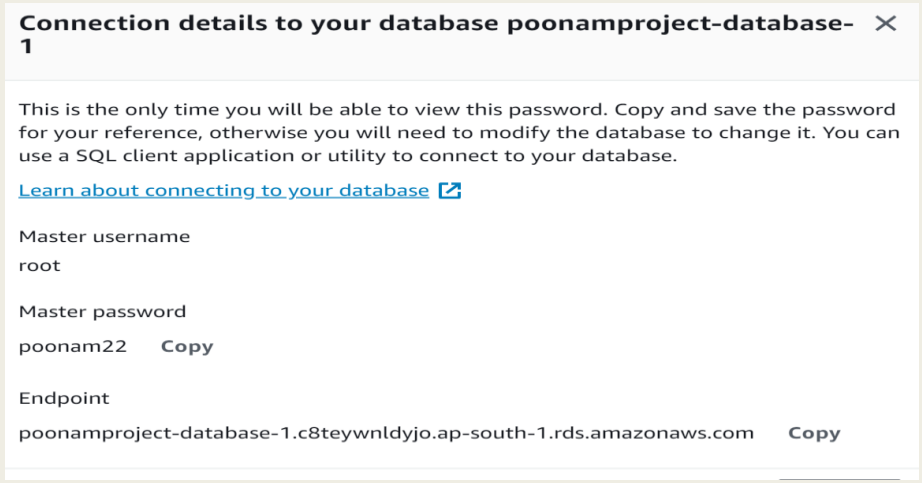
Select VPC Security Group





Now the Database is getting created…….



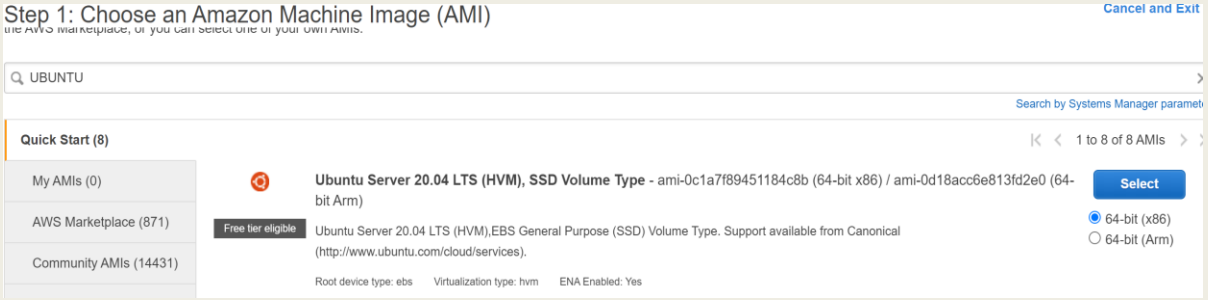


Copy endpoint:
poonamproject-database-1.c8teywnldyjo.ap-south-1.rds.amazonaws.com

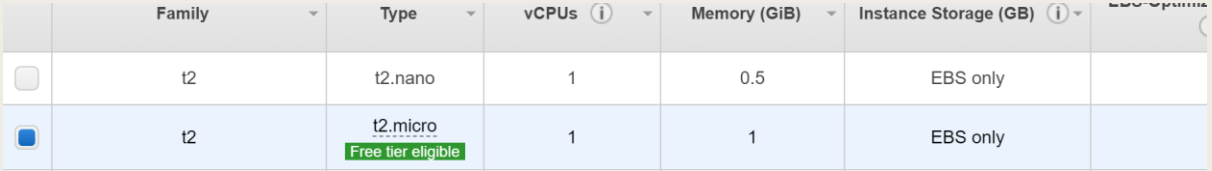
Database is successfully created !!

TASK 4: Launching and Setting up EC2 Instance

1. Launch an Ubuntu EC2 instance

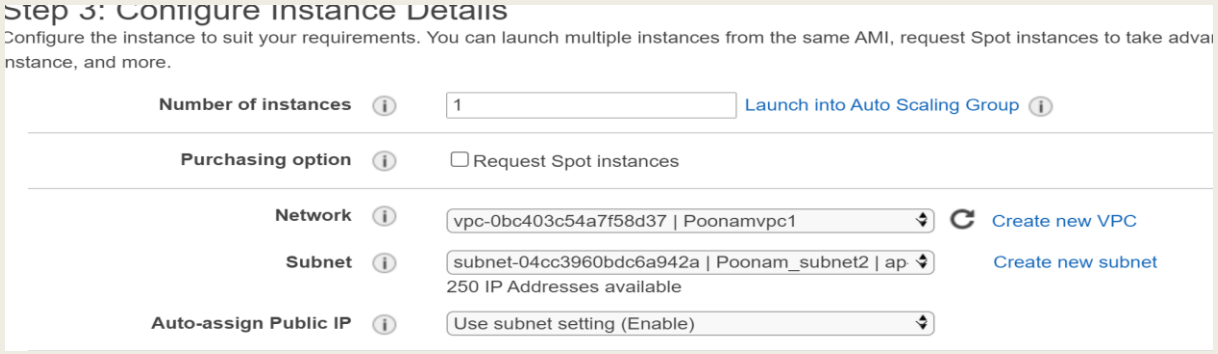


2. Select t2.micro (free tier)



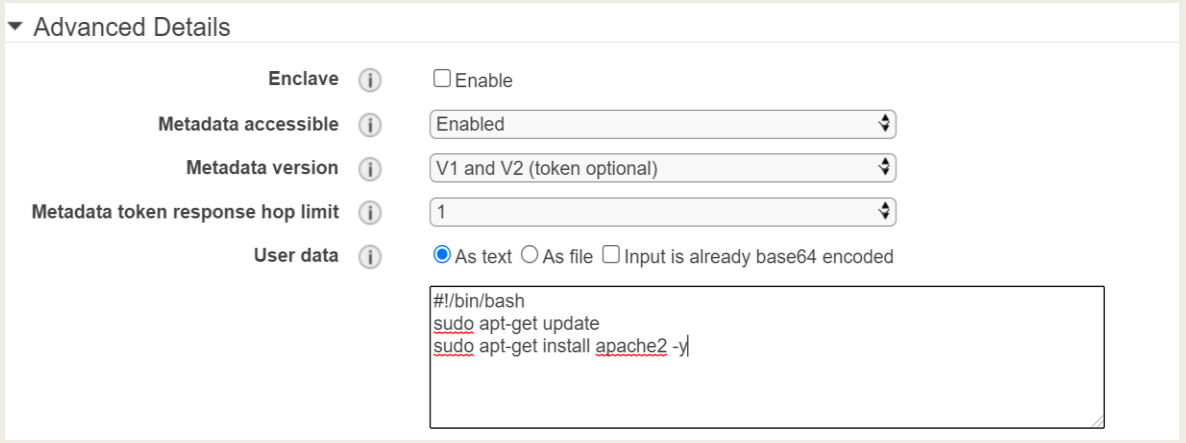
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)
☐	t2	t2.nano	1	0.5	EBS only
☑	t2	t2.micro Free tier eligible	1	1	EBS only

3. Configure instances



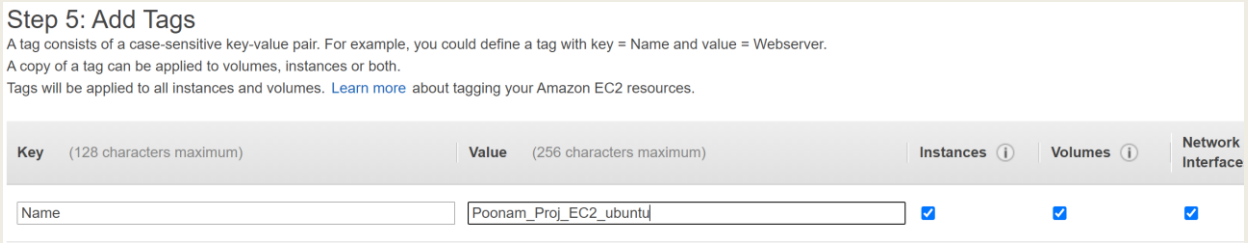
4. Paste below script in “User data”

#!/bin/bash
sudo apt-get update
sudo apt-get install apache2 -y



5. Add Tags

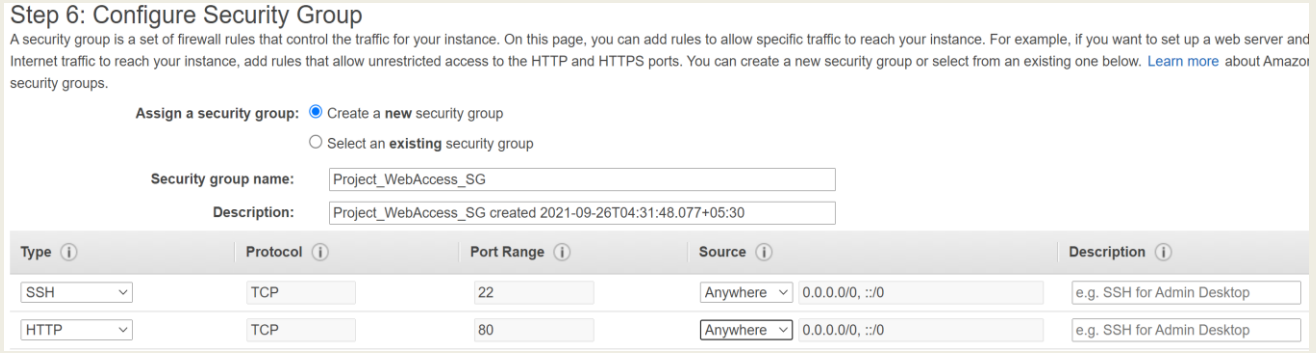
Name: Poonam_Proj_EC2_ubuntu



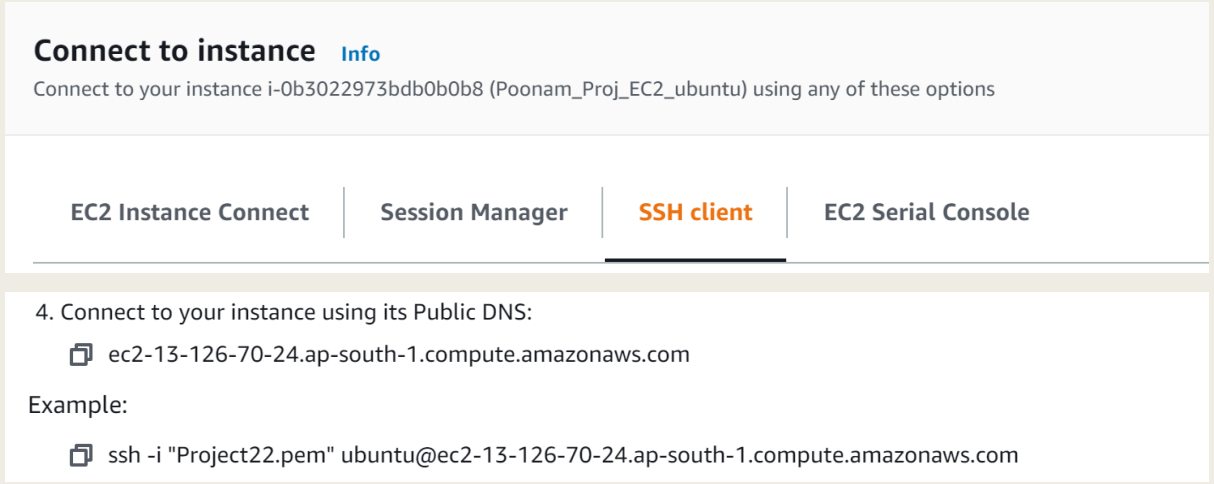
6. Create Security Group: Project_WebAccess_SG

Open Port

SSH	**TCP**	**22**	**Anywhere**
HTTP	**TCP**	**80**	**Anywhere**



7. Using Git Bash terminal , connect Instance just created



8. On the terminal give below command

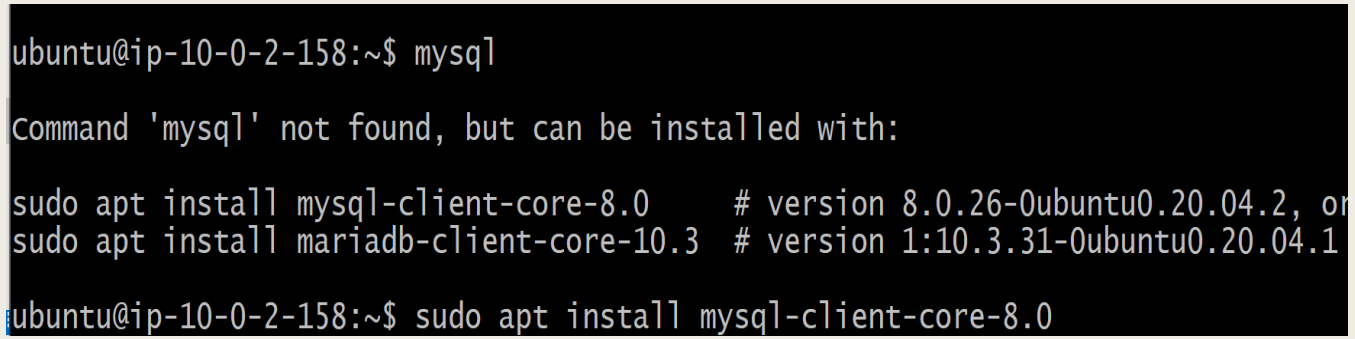
mysql

It will give following error

Command 'mysql' not found, but can be installed with:

sudo apt install mysql-client-core-8.0 # version 8.0.26-0ubuntu0.20.04.2

Copy the highlighted command and paste it again on the terminal



9. Now give command

[Now using mysql -h host from RDS endpoint, -uroot or admin and -P for password, we can connect to the mysql rds instance we created and use it.

$ mysql -h<RDS Endpoint> -u<adminusername> -p<password>]

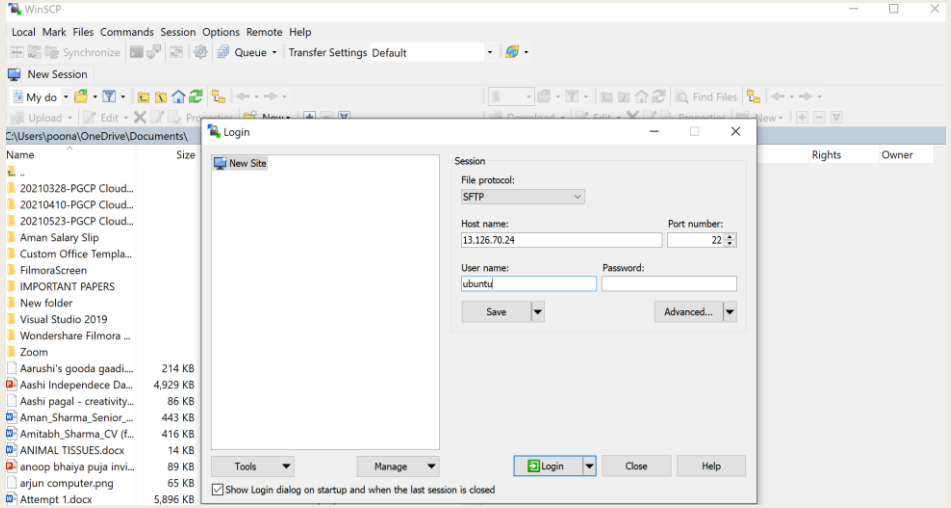
```
mysql -hpoonamproject-database-1.c8teywnldyjo.ap-south-1.rds.amazonaws.com -uroot  -ppoonam22
```

We now land on My SQL prompt

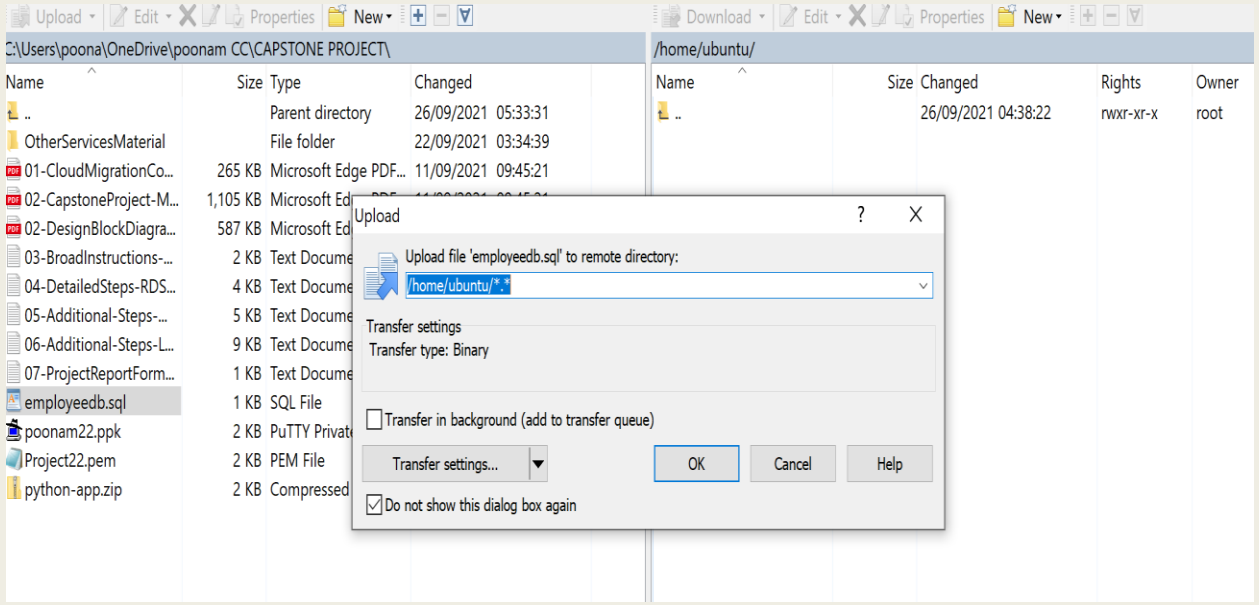
$ mysql> show databases

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input sta

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

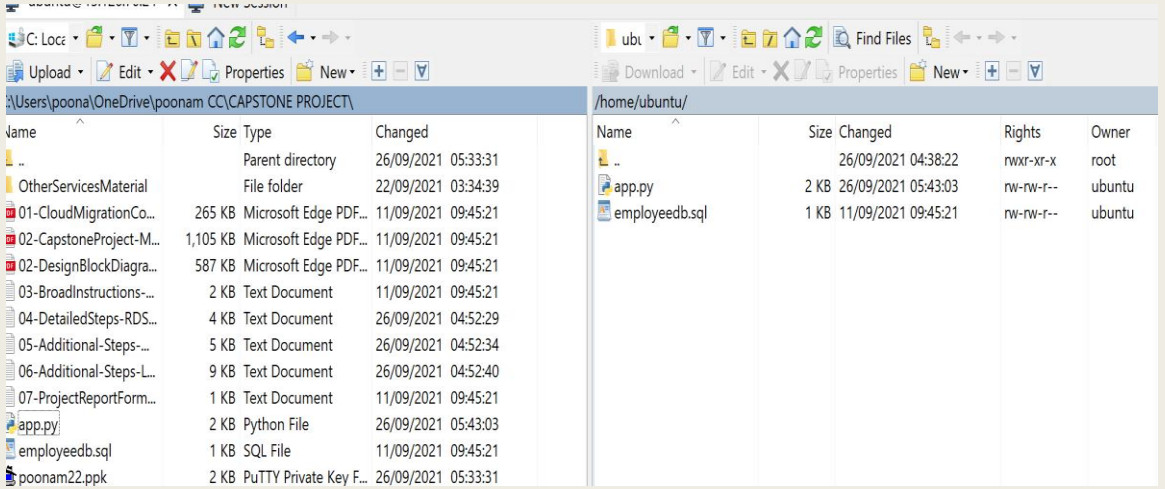
10. Open WinSCP



11. Copy the **employeedb.sql** file in the host machine and drop it to the EC2 machine



12. Copy the **app.py** also



13. Now on the terminal give following command

mysql> show databases; [below is the output]

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input sta

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

14. Mysql> source employeedb.sql

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> source employeedb.sql
Query OK, 1 row affected (0.05 sec)

Database changed
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 22 rows affected (0.01 sec)
Records: 22  Duplicates: 0  Warnings: 0
```

Mysql> show databases;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| employee_db        |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

So now **employeedb** is listed in Database

```
mysql> use employee_db;
Database changed

mysql> show tables;
+-----------------------+
| Tables_in_employee_db |
+-----------------------+
| employees             |
+-----------------------+
1 row in set (0.00 sec)
```

mysql> SELECT * from employees; it will give below output

```
mysql> SELECT * from employees;
+-------------------+
| name              |
+-------------------+
| Murphy Diane      |
| Firrelli Jeff     |
| Patterson William |
| Bondur Gerard     |
| Bow Anthony       |
| Jennings Leslie   |
| Thompson Leslie   |
| Firrelli Julie    |
| Patterson Steve   |
| Tseng Foon Yue    |
| Vanauf George     |
| Bondur Loui       |
| Hernandez Gerard  |
| Castillo Pamela   |
| Bott Larry        |
| Jones Barry       |
| Fixter Andy       |
| Marsh Peter       |
| King Tom          |
| Nishi Mami        |
| Kato Yoshimi      |
| Gerard Martin     |
+-------------------+
22 rows in set (0.00 sec)
```

```
mysql> SELECT * from <databasename>.<tablename>;
mysql> SELECT * from employee_db.employees; +-------------------+
| name              |
+-------------------+
| Murphy Diane      |
| Firrelli Jeff     |
| Patterson William |
| Bondur Gerard     |
| Bow Anthony       |
| Jennings Leslie   |
| Thompson Leslie   |
| Firrelli Julie    |
| Patterson Steve   |
| Tseng Foon Yue    |
| Vanauf George     |
| Bondur Loui       |
| Hernandez Gerard  |
| Castillo Pamela   |
| Bott Larry        |
| Jones Barry       |
| Fixter Andy       |
| Marsh Peter       |
| King Tom          |
| Nishi Mami        |
| Kato Yoshimi      |
| Gerard Martin     |
+-------------------+
22 rows in set (0.00 sec)
```

TASK 5: Configuring and testing Python-Flask application

Normally on the Ubuntu EC2instance python3 is made available. You can check it out by giving the command:

```
$ which python3
$ python3 --version
```

```
mysql> quit
Bye
ubuntu@ip-10-0-2-158:~$
ubuntu@ip-10-0-2-158:~$
ubuntu@ip-10-0-2-158:~$ which python
ubuntu@ip-10-0-2-158:~$ which python3
/usr/bin/python3
ubuntu@ip-10-0-2-158:~$ python3 --version
Python 3.8.5
ubuntu@ip-10-0-2-158:~$
```

A. On the EC2 instance - install pip3 using the following command.
   $ sudo apt-get install python3-pip

B. Install Flask and flask-mysql using the commands below.
   $ sudo pip3 install flask
   $ sudo pip3 install flask-mysql

```
ubuntu@ip-10-0-2-158:~$ sudo pip3 install flask-mysql
Collecting flask-mysql
  Downloading Flask_MySQL-1.5.2-py2.py3-none-any.whl (3.8 kB)
Collecting PyMySQL
  Downloading PyMySQL-1.0.2-py3-none-any.whl (43 kB)
     |████████████████████████████████| 43 kB 985 kB/s
Requirement already satisfied: Flask in /usr/local/lib/python3.8/dist-packages
from flask-mysql) (2.0.1)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.8/di
t-packages (from Flask->flask-mysql) (2.0.1)
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.8/dist-pac
ages (from Flask->flask-mysql) (3.0.1)
Requirement already satisfied: click>=7.1.2 in /usr/local/lib/python3.8/dist-pa
kages (from Flask->flask-mysql) (8.0.1)
Requirement already satisfied: Werkzeug>=2.0 in /usr/local/lib/python3.8/dist-p
ckages (from Flask->flask-mysql) (2.0.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.8/dist
packages (from Jinja2>=3.0->Flask->flask-mysql) (2.0.1)
Installing collected packages: PyMySQL, flask-mysql
Successfully installed PyMySQL-1.0.2 flask-mysql-1.5.2
ubuntu@ip-10-0-2-158:~$
```

1. Give command ls -l to see the list

```
ubuntu@ip-10-0-2-158:~$ pwd
/home/ubuntu
ubuntu@ip-10-0-2-158:~$ ls -l
total 12
-rw-rw-r-- 1 ubuntu ubuntu 1052 Sep 26 00:13 app.py
-rw-rw-r-- 1 ubuntu ubuntu  738 Sep 11 04:15 employeedb.sql
-rw-rw-r-- 1 ubuntu ubuntu 1243 Sep 11 04:15 python-app.zip
ubuntu@ip-10-0-2-158:~$
```

2. By using nano editor(which is same as vi or vim editor)
   Nano app.py

```
 GNU nano 4.8                                  app.py
import os
from flask import Flask
from flaskext.mysql import MySQL       # For newer versions of flask-mysql
# from flask.ext.mysql import MySQL    # For older versions of flask-mysql
app = Flask(__name__)

mysql = MySQL()

mysql_database_host = 'MYSQL_DATABASE_HOST' in os.environ and os.environ['MYSQL>

# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'db_user'
app.config['MYSQL_DATABASE_PASSWORD'] = 'PasswOrd'
app.config['MYSQL_DATABASE_DB'] = 'employee_db'
app.config['MYSQL_DATABASE_HOST'] = mysql_database_host
mysql.init_app(app)

conn = mysql.connect()

cursor = conn.cursor()
                        [ Read 42 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

3. Make changes in USERNAME, PASSWORD & DATABASE HOSTNAME

A sample Python-Flask applicaton is given in the file app.py. Make sure to change the following parameters in the code to reflect your RDS db instance details.

app.config['MYSQL_DATABASE_USER'] = '<your admin username>'
app.config['MYSQL_DATABASE_PASSWORD'] = '<your admin user password>'
app.config['MYSQL_DATABASE_DB'] = 'employee_db'
app.config['MYSQL_DATABASE_HOST'] = '<your RDS instance end point>'

```
import os
from flask import Flask
from flaskext.mysql import MySQL       # For newer versions of flask-mysql
# from flask.ext.mysql import MySQL    # For older versions of flask-mysql
app = Flask(__name__)

mysql = MySQL()

mysql_database_host = 'MYSQL_DATABASE_HOST' in os.environ and os.environ['MYSQL_DATABASE_HOST'] or  'localho

# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = 'Poonam22'
app.config['MYSQL_DATABASE_DB'] = 'employee_db'
app.config['MYSQL_DATABASE_HOST'] = 'poonamproject-database-1.c8teywnldyjo.ap-south-1.rds.amazonaws.com'
mysql.init_app(app)

conn = mysql.connect()

cursor = conn.cursor()
```

4. In Database host name we need to provide the endpoint of our RDS instance
   'poonamproject-database-1.c8teywnldyjo.ap-south-1.rds.amazonaws.com'
   Ctrl + X -→ to quit
   Save changes → Yes
   In app.py → Yes

5. Give below command

Cat app.py and see the output

```
ubuntu@ip-10-0-2-158:~$ cat app.py
import os
from flask import Flask
from flaskext.mysql import MySQL      # For newer versions of flask-mysql
# from flask.ext.mysql import MySQL   # For older versions of flask-mysql
app = Flask(__name__)

mysql = MySQL()

mysql_database_host = 'MYSQL_DATABASE_HOST' in os.environ and os.environ['MYSQL_DATABASE_HOST'] or  'localhost'

# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = 'poonam22'
app.config['MYSQL_DATABASE_DB'] = 'employee_db'
app.config['MYSQL_DATABASE_HOST'] = 'poonamproject-database-1.c8teywnldyjo.ap-south-1.rds.amazonaws.com'
mysql.init_app(app)

conn = mysql.connect()

cursor = conn.cursor()

@app.route("/")
def main():
    return "Welcome!"

@app.route('/how are you')
def hello():
    return 'I am good, how about you?'

@app.route('/read from database')
def read():
    cursor.execute("SELECT * FROM employees")
    row = cursor.fetchone()
    result = []
    while row is not None:
      result.append(row[0])
      row = cursor.fetchone()

    return ",".join(result)

if __name__ == "__main__":
    app.run()
```

6. Now test the python application by running it with the command below.
$ FLASK_APP=app.py flask run --host=0.0.0.0

```
ubuntu@ip-10-0-2-158:~$ FLASK_APP=app.py flask run --host=0.0.0.0
 * Serving Flask app 'app.py' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://10.0.2.158:5000/ (Press CTRL+C to quit)
```

So now Flask app is up and running

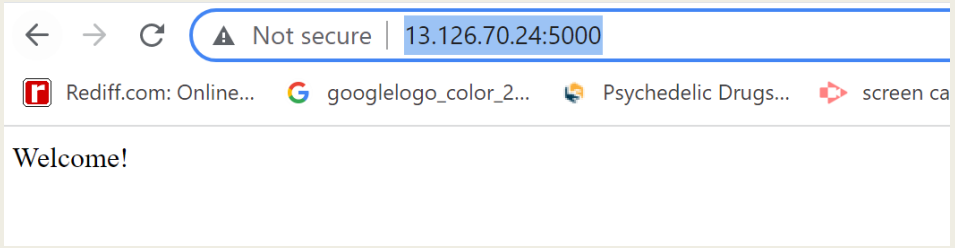
7. But when we copy the Public IP of our host followed by :5000 i.e. http://13.126.70.24:5000/
We will get an error

8. Since it uses port number 5000 you may get an error page. So make sure you edit the security group of you EC2 instance to allow inbound traffic on Custom TCP on port number 5000.

Add rules

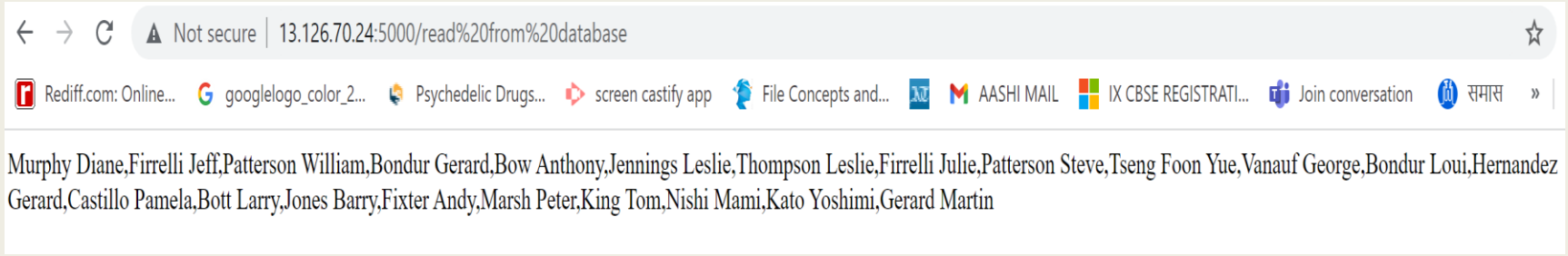
Custom TCP	**5000**	**Anywhere IPv4**
Custom TCP	**5000**	**Anywhere IPv6**

9. Now paste this in the browser
   a. http://13.126.70.24:5000/



   b. 13.126.70.24/5000/how are you

c. 13.126.70.24/5000/read from database



So our Python app is working properly now !!

Flask is installed with the prerequisites like pip3 tool as mentioned in the notes for the earlier steps

A) Install and Enable mod_wsgi (Web Server Gateway Interface Module)

$ sudo apt-get install libapache2-mod-wsgi-py3 python-dev

Module wsgi gets enabled by default after installation. You can give the following command to make sure Module wsgi is enabled.

$ sudo a2enmod wsgi

```
ubuntu@ip-10-0-2-158:~$ sudo a2enmod wsgi
Module wsgi already enabled
ubuntu@ip-10-0-2-158:~$ ls -l /var/www
total 4
drwxr-xr-x 2 root root 4096 Sep 25 23:08 html
ubuntu@ip-10-0-2-158:~$
```

B) Set up a sample Flask Application

List the contents of the default web pages directory.

$ ls -l /var/www

```
ubuntu@ip-10-0-2-158:~$ ls -l  /var/www
total 8
drwxr-xr-x 3 root root 4096 Sep 26 02:06 FlaskApp
drwxr-xr-x 2 root root 4096 Sep 25 23:08 html
ubuntu@ip-10-0-2-158:~$
```

A) Create the required sub-directories in the above.

$ sudo mkdir -p /var/www/FlaskApp/FlaskApp

```
drwxr-xr-x 3 root root 4096 Sep 26 02:06 FlaskApp
drwxr-xr-x 2 root root 4096 Sep 25 23:08 html
ubuntu@ip-10-0-2-158:~$ ls -l  /var/www/FlaskApp/
total 4
drwxr-xr-x 2 root root 4096 Sep 26 02:06 FlaskApp
ubuntu@ip-10-0-2-158:~$ ls -l  /var/www/FlaskApp/FlaskApp/
total 0
ubuntu@ip-10-0-2-158:~$
```

A) Create a sample test python application.

$ sudo nano /var/www/FlaskApp/FlaskApp/__init__.py

OR

$ sudo cp app.py /var/www/FlaskApp/FlaskApp/__init__.py

Using $ sudo nano /var/www/FlaskApp/FlaskApp/__init__.py , will open an empty file, where we will copy the code and save it and come out of nano editor.

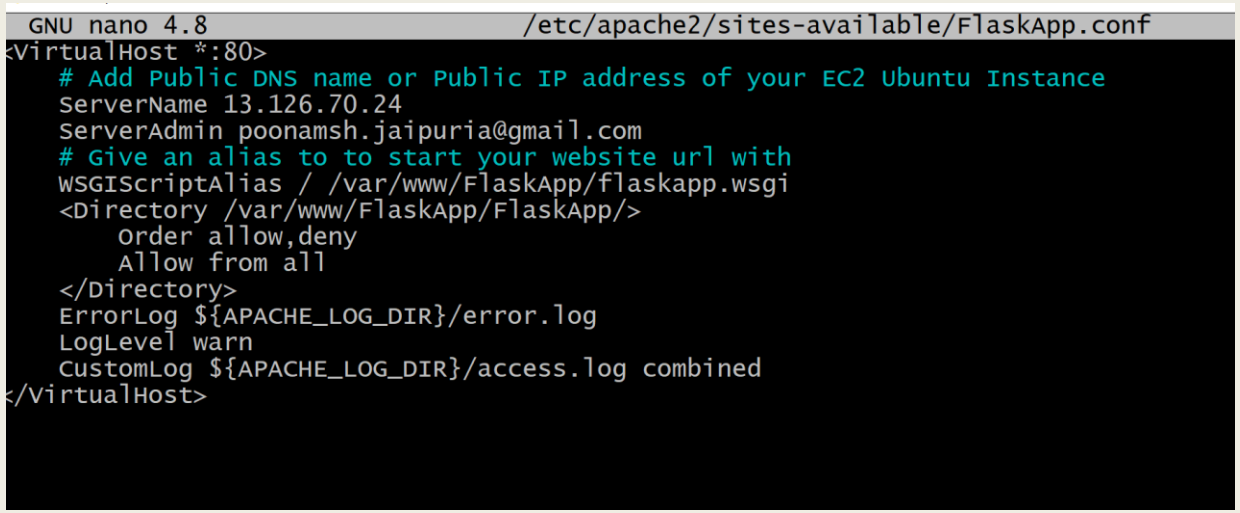
CONFIGURE AND ENABLE A NEW VIRTUAL HOST AT PORT NUMBER 80

1. **Create the configuration file for the FlaskApp.**

$ sudo nano /etc/apache2/sites-available/FlaskApp.conf

Copy paste the following lines.

```
-->
<VirtualHost *:80>
    # Add Public DNS name or Public IP address of your EC2 Ubuntu Instance
    ServerName 13.126.70.24
    ServerAdmin poonamsh.jaipuria@gmail.com
    # Give an alias to to start your website url with
    WSGIScriptAlias / /var/www/FlaskApp/flaskapp.wsgi
    <Directory /var/www/FlaskApp/FlaskApp/>
        Order allow,deny
        Allow from all
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
<--
```



1. Enable the virtual host with the comand below

   $ sudo a2ensite FlaskApp

   Then run the following command:

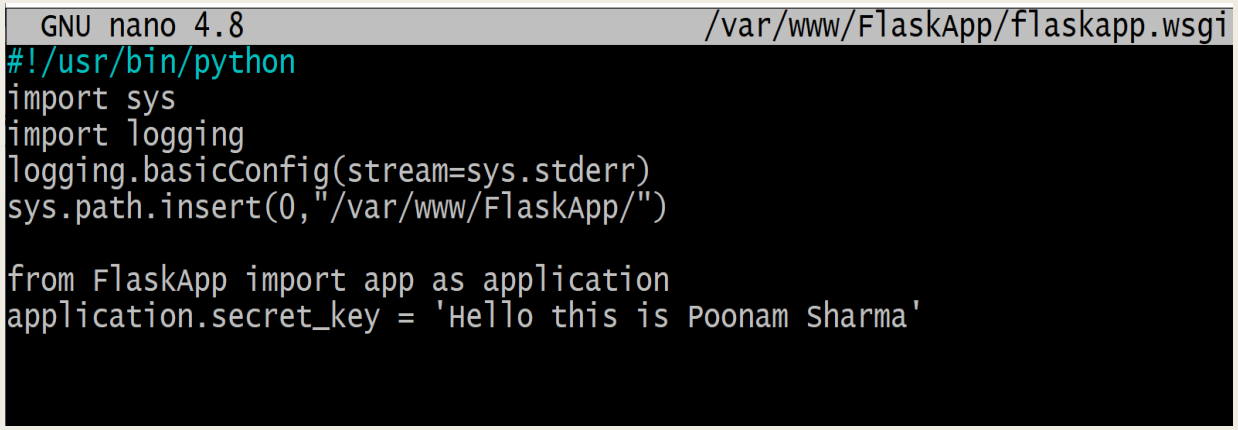
   $ sudo systemctl reload apache2

2. Create the .wsgi File

   Run the following command and create a configuration file flaskapp.wsgi in the FlaskApp directory

   $ sudo nano /var/www/FlaskApp/flaskapp.wsgi

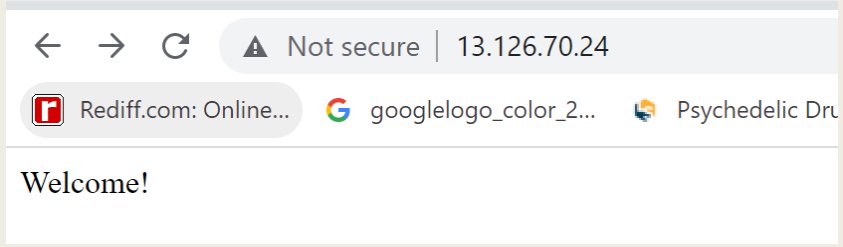
```
#!/usr/bin/python
import sys
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0,"/var/www/FlaskApp/")

from FlaskApp import app as application
application.secret_key = 'Hello this is Poonam Sharma'
```



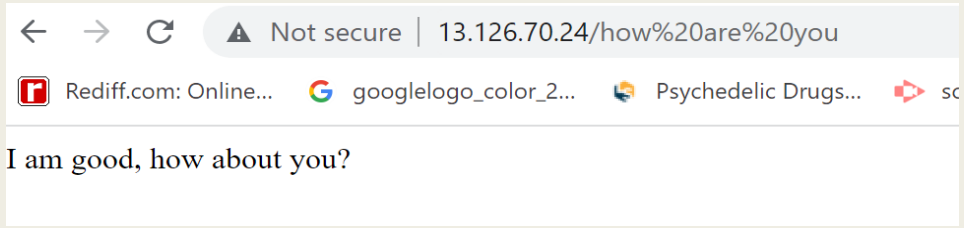
3. Finally restart apache2 server.

   $ sudo service apache2 restart

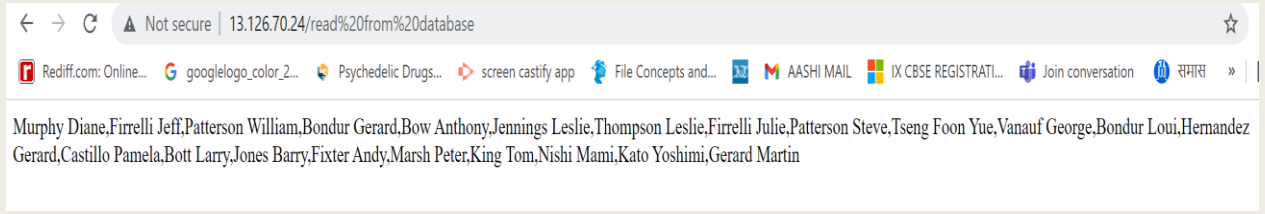
a) Copy public IP of EC2 instance and paste on browser, we will get this output



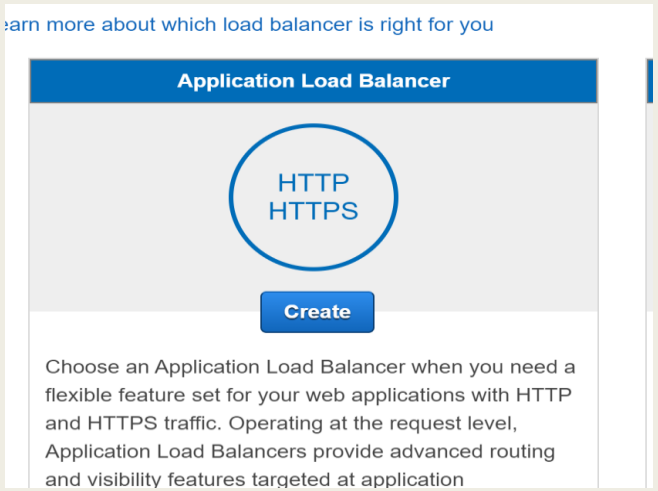
b) Public_IP/how are you will give below output

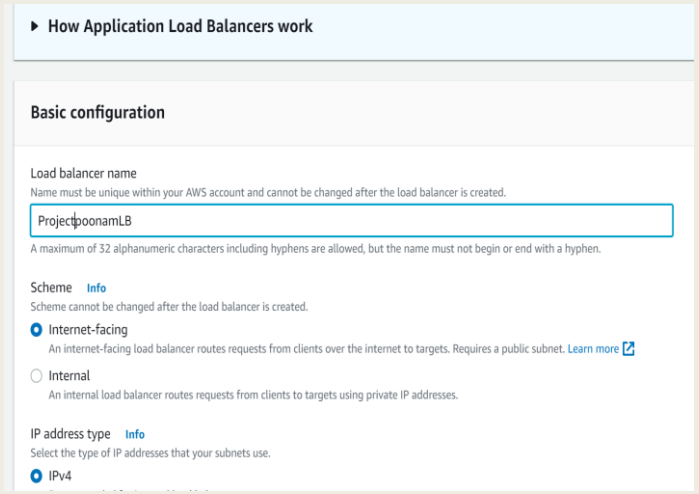


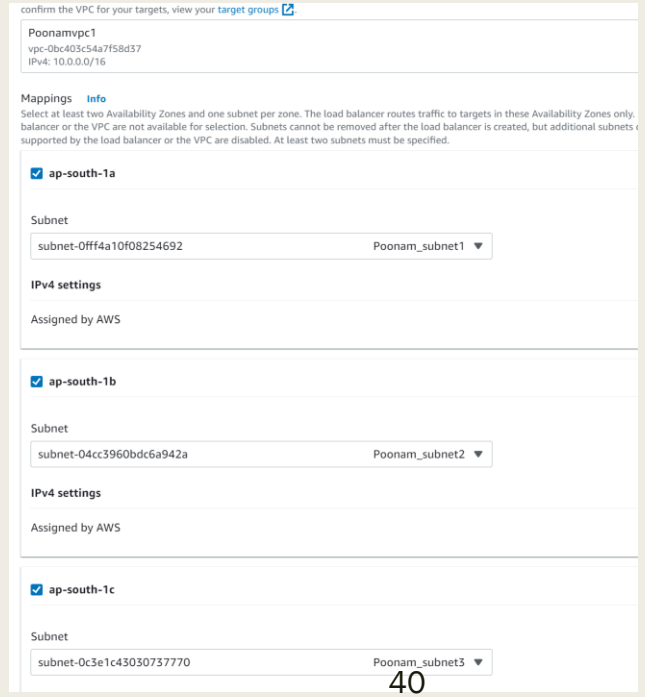
c) Public_IP/read from database



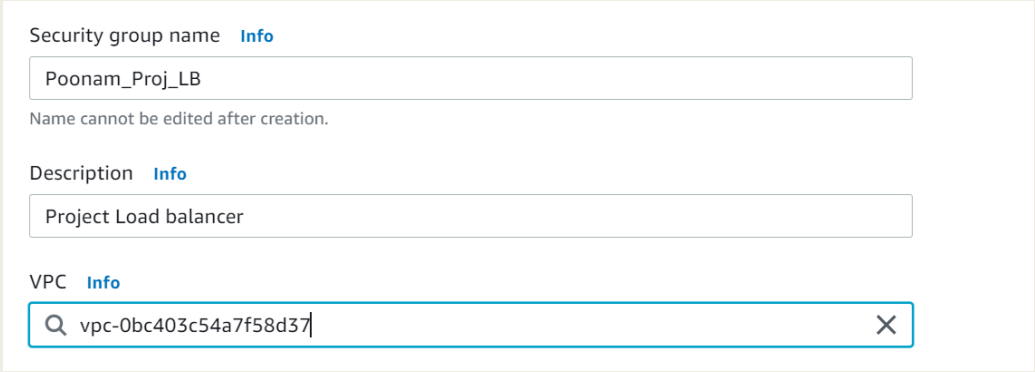
1. Create a Load Balancer (Application Load Balancer)

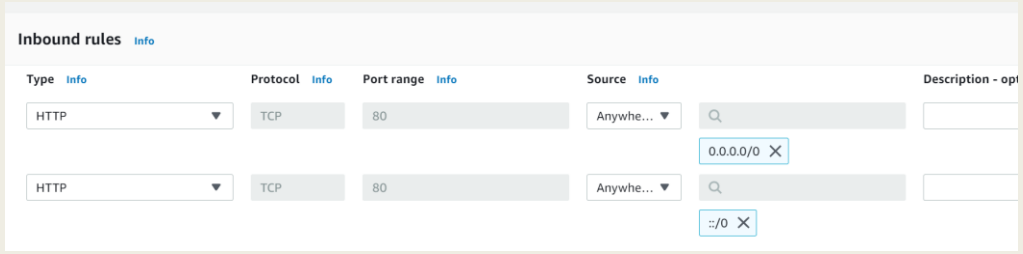


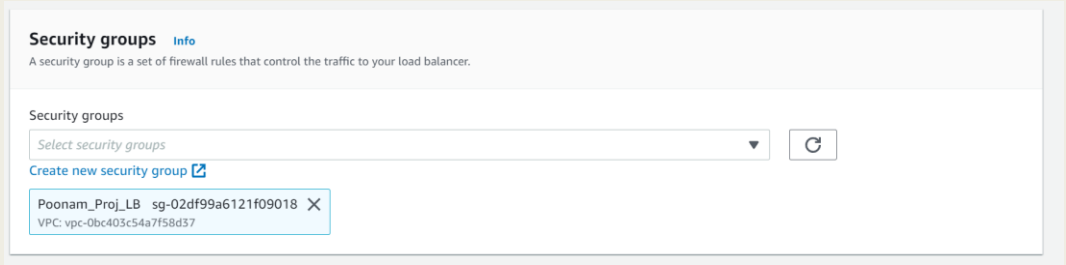




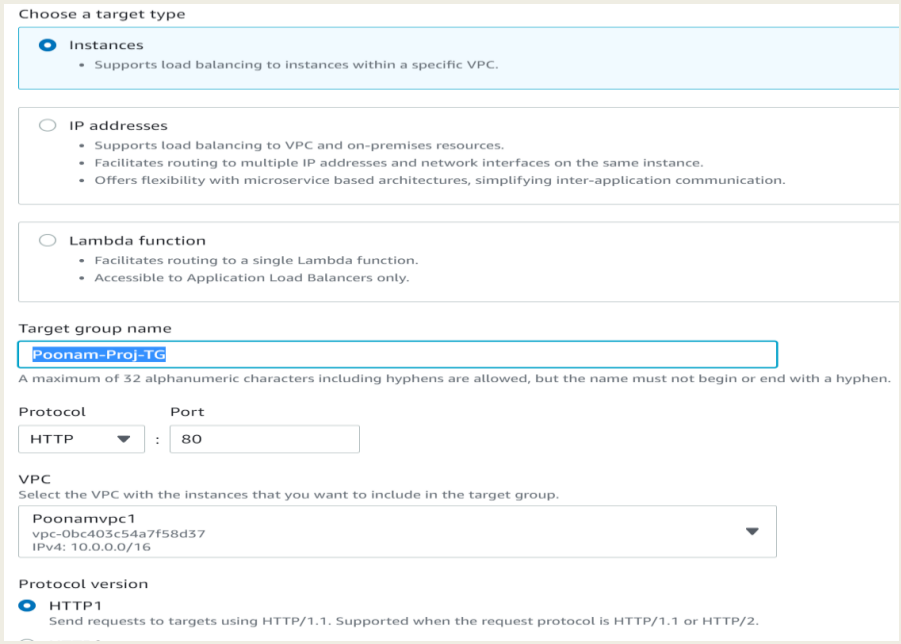
Security group name : Poonam_Proj_LB
Description: Project Load balancer



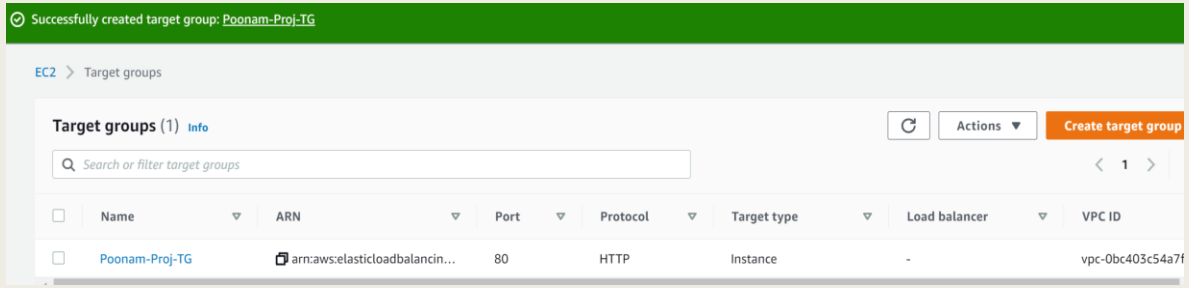




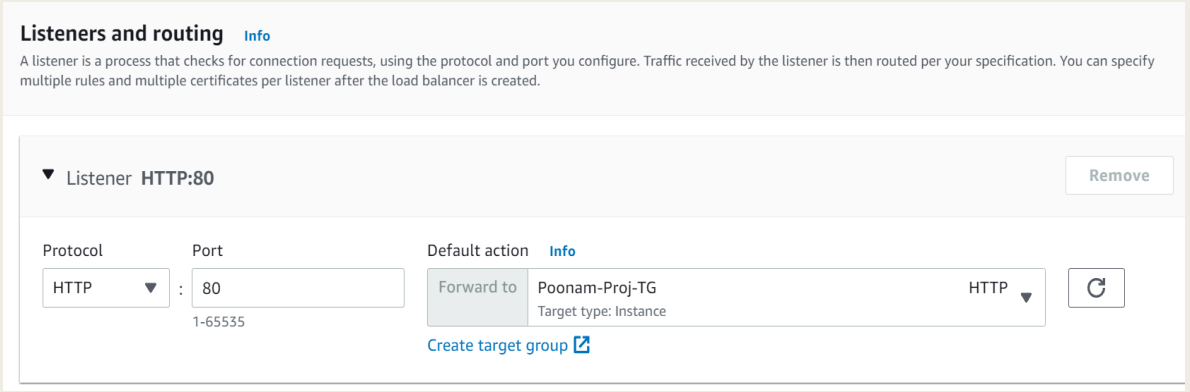
Click Next: Configure Routing
Select "New target group" and give it a name such as **Poonam-Proj-TG**.
Leave the target type as **Instance**, the protocol as HTTP on port 80.

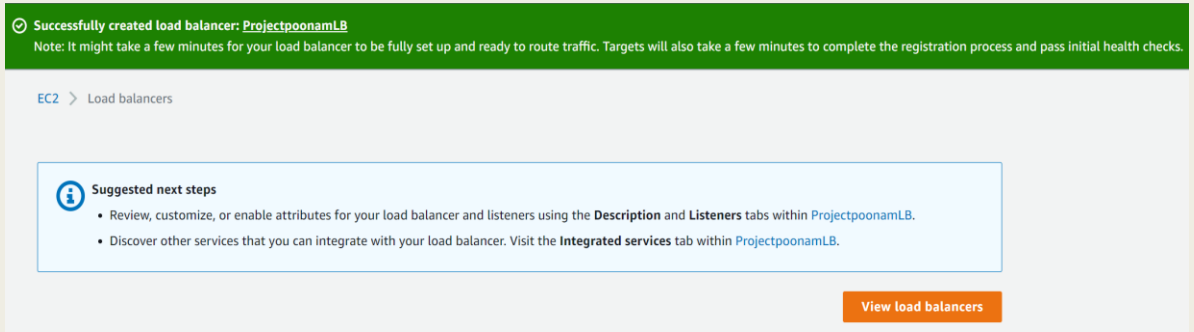


No need to select Instance, just click on **Create Target Group**

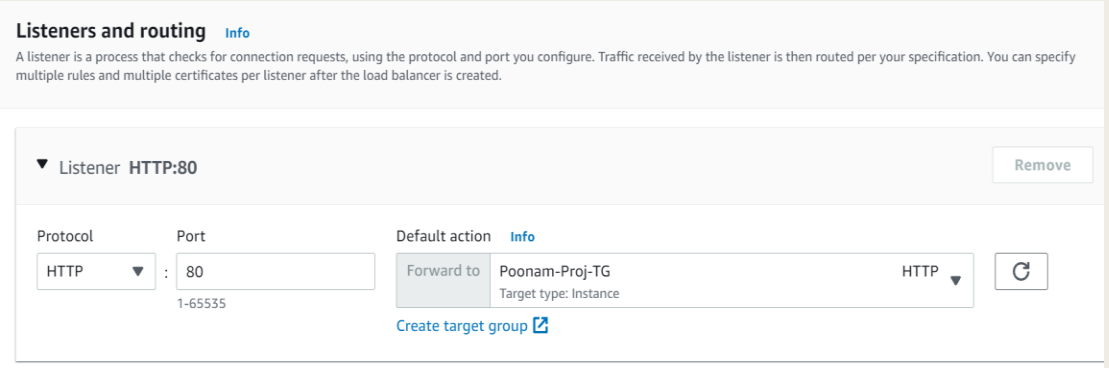


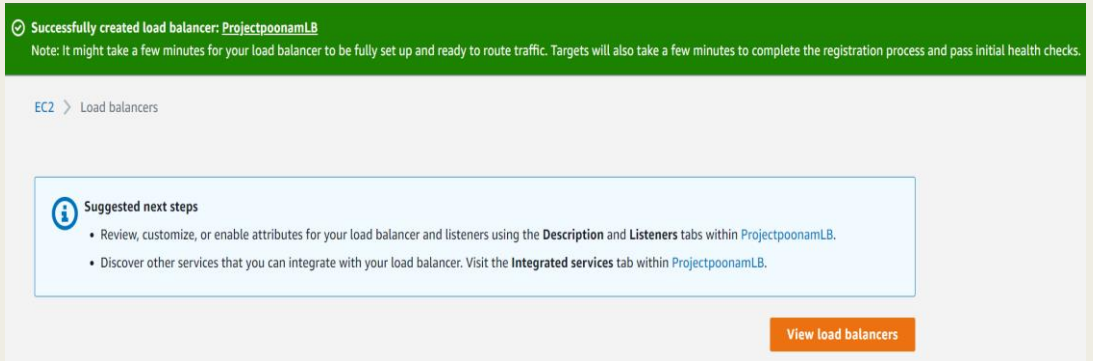
Once Target group is created refresh the target group box in load balancer page

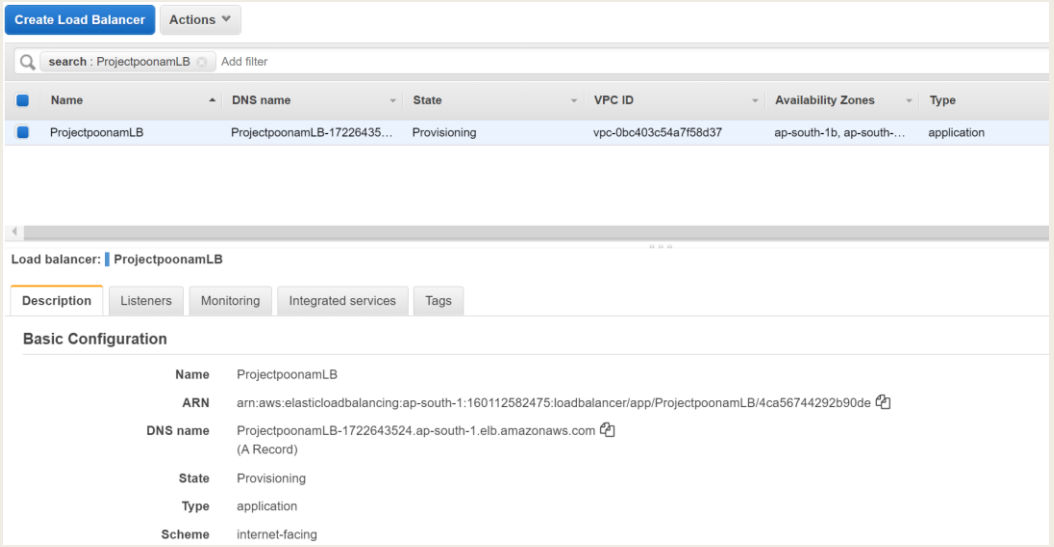




Once Target group is created refresh the target group box in load balancer page







Copy the DNS name of Load Balancer

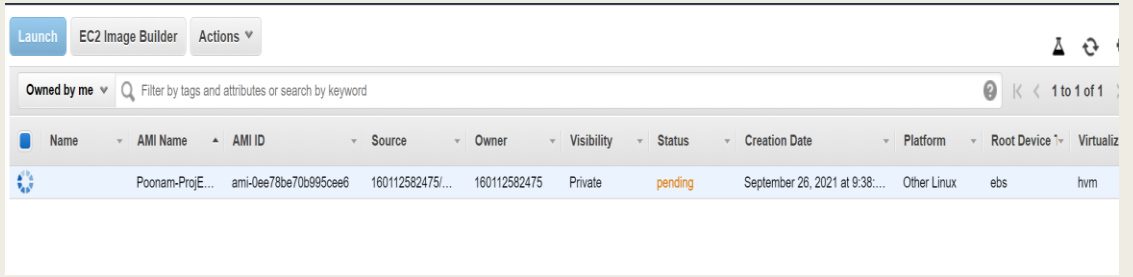
ProjectpoonamLB-1722643524.ap-south-1.elb.amazonaws.com

2. CREATE AN EC2 UBUNTU INSTANCE AND ITS AMI

On your Ubuntu EC2instancnce -
Update the configuration file for the FlaskApp.
$ sudo nano /etc/apache2/sites-available/FlaskApp.conf

```
  GNU nano 4.8                  /etc/apache2/sites-available/FlaskApp.conf                  Modified
<VirtualHost *:80>
    # Add Public DNS name or Public IP address of your EC2 Ubuntu Instance
    ServerName ProjectpoonamLB-1722643524.ap-south-1.elb.amazonaws.com
    ServerAdmin poonamsh.jaipuria@gmail.com
    # Give an alias to to start your website url with
    WSGIScriptAlias / /var/www/FlaskApp/flaskapp.wsgi
    <Directory /var/www/FlaskApp/FlaskApp/>
        Order allow,deny
        Allow from all
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

Go to the EC2 Instance ,
Select the instance → Actions → Image and Templates → Create Image
Image name: **Poonam-ProjEC2-IMG**
Image description: Project EC2 Image



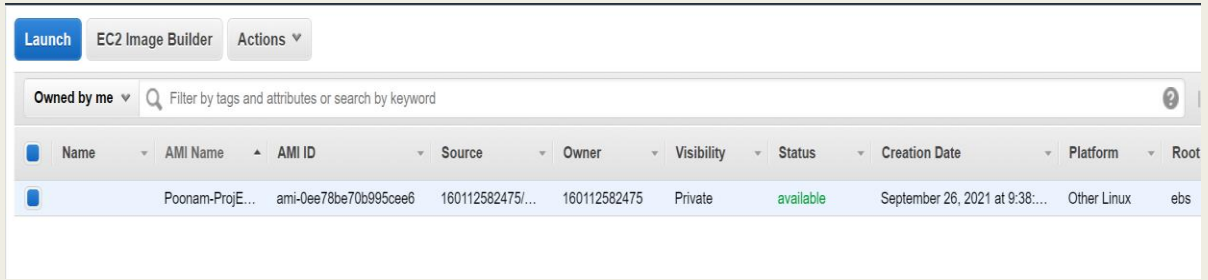
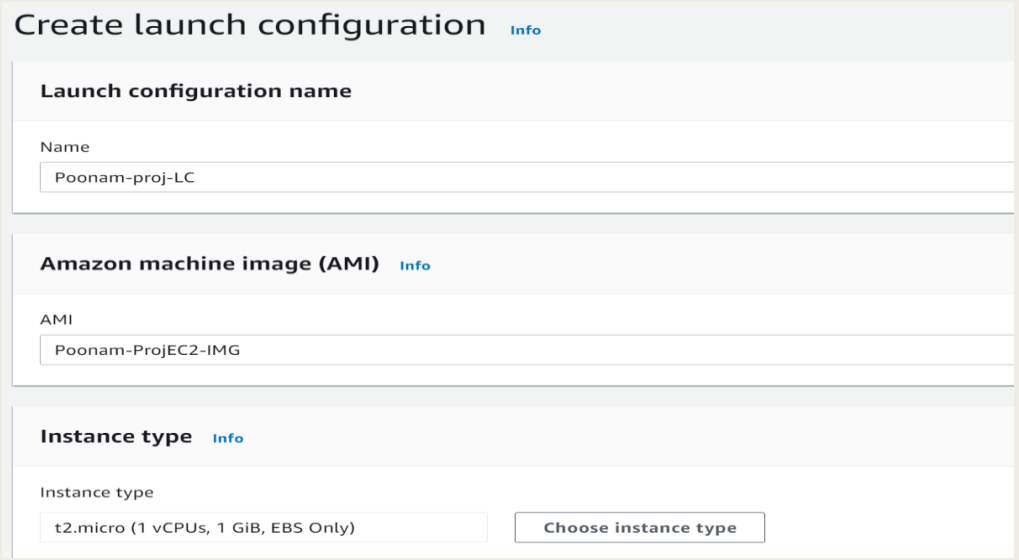


Image Poonam-ProjEC2-IMG is launched !!

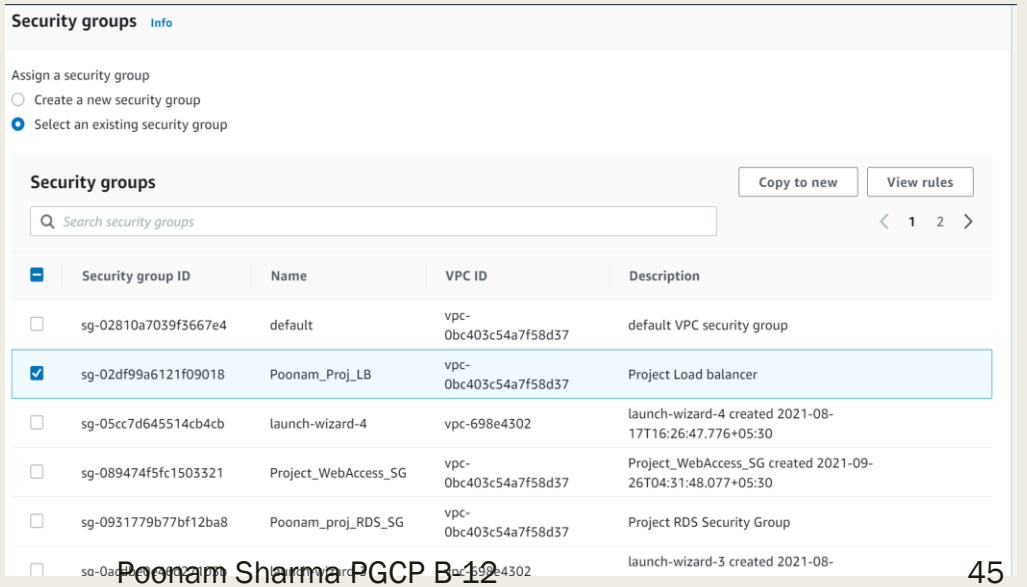
3. CREATE A LAUNCH CONFIGURATION

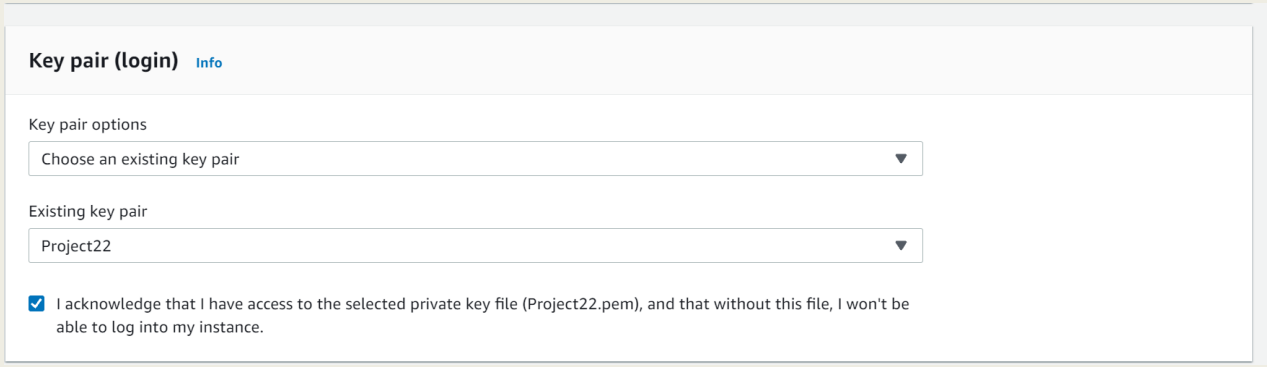
On the navigation pane, under AUTO SCALING, choose Launch Configurations.
Choose Create launch configuration, and enter a name for the launch configuration such as

Name: Poonam-proj-LC

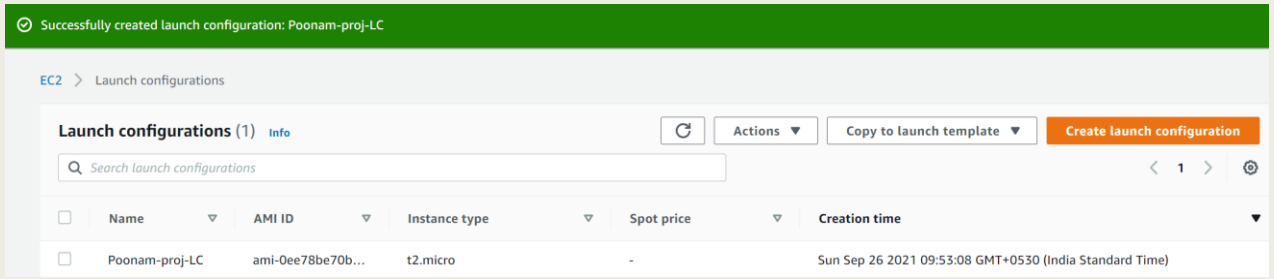


And select the Security Group of the Elastic Load Balancer





Select the acknowledgment check box, and then choose Create launch configuration.



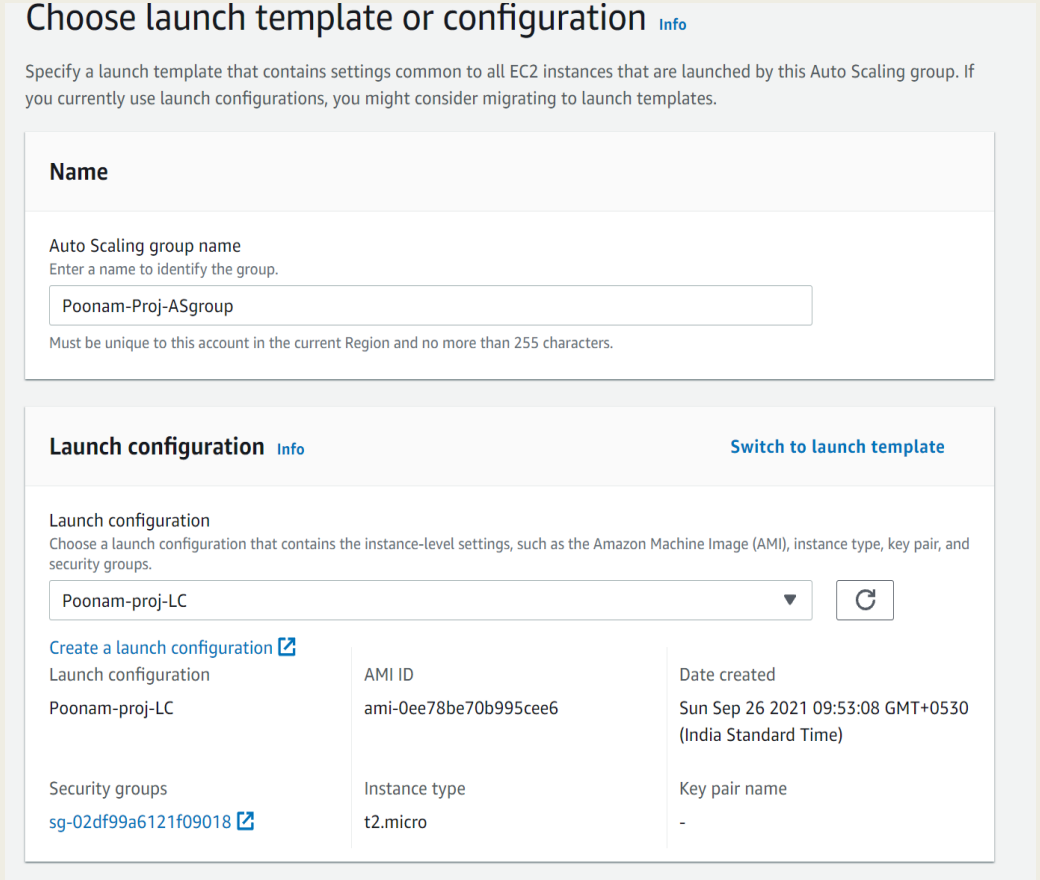
Launch configuration is successfully launched !!

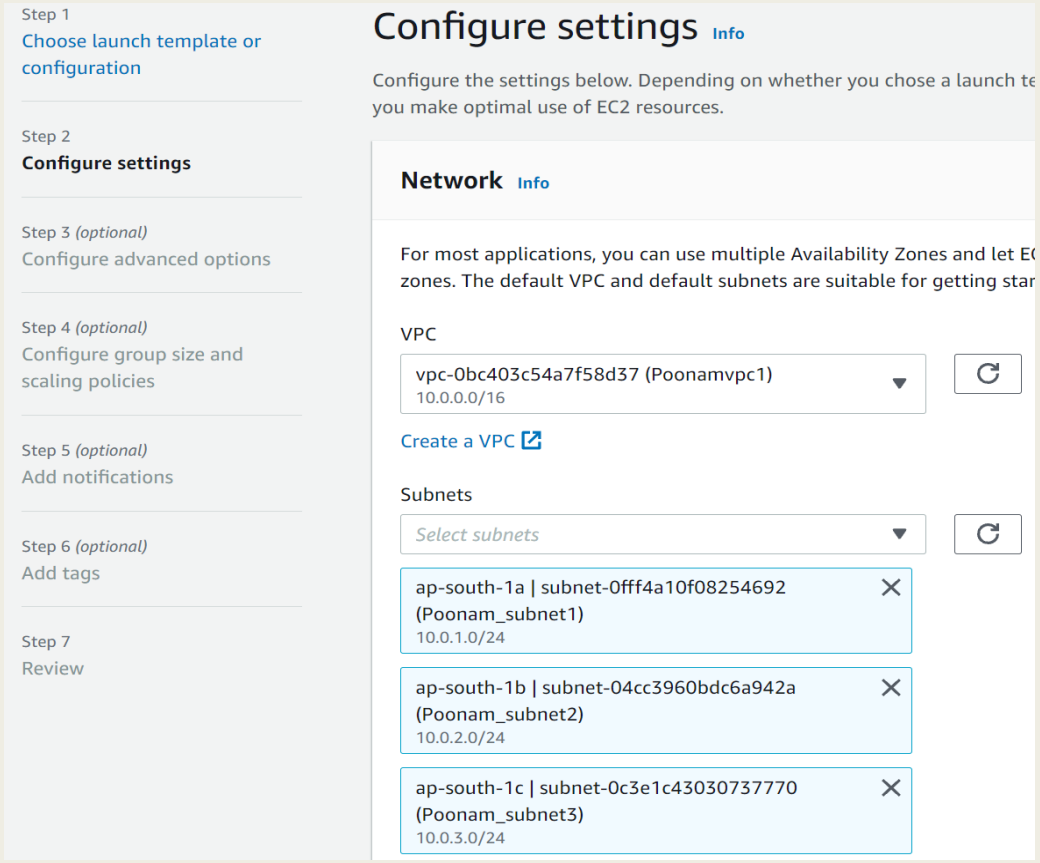
4. CREATE AUTO SCALING GROUP

On EC2 services page - Choose Create an Auto Scaling group on left navigation panel.

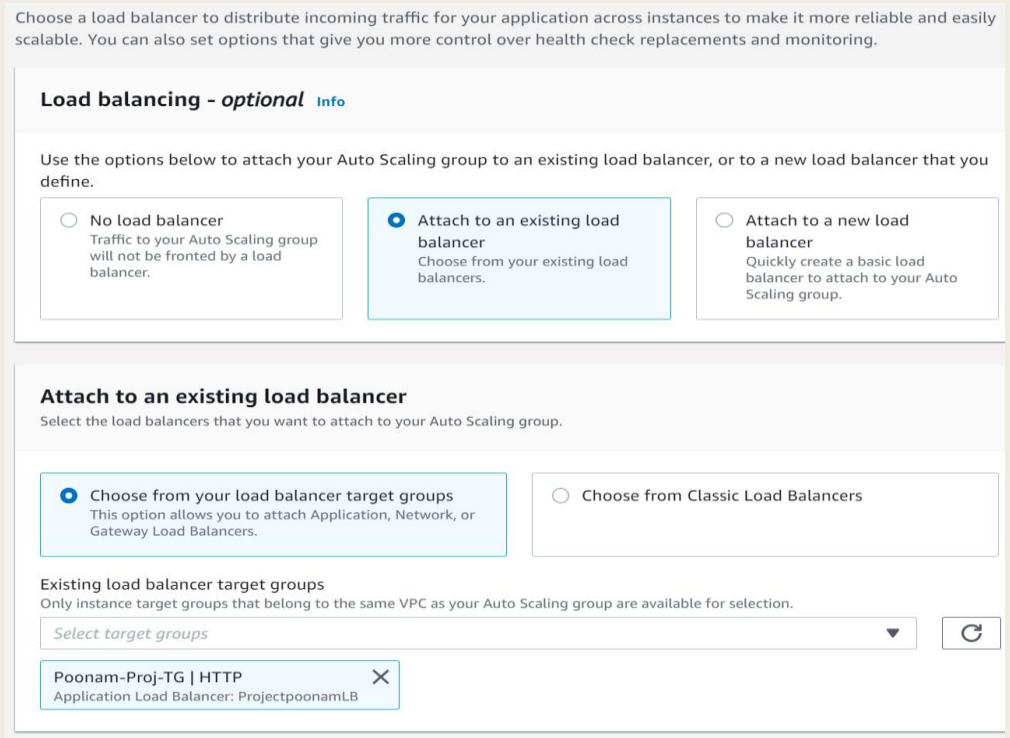
On the page - Choose launch template or configuration page, for Auto Scaling group name, enter a name for your Auto Scaling group like Poonam-Proj-ASgroup

Select 'Switch to launch'

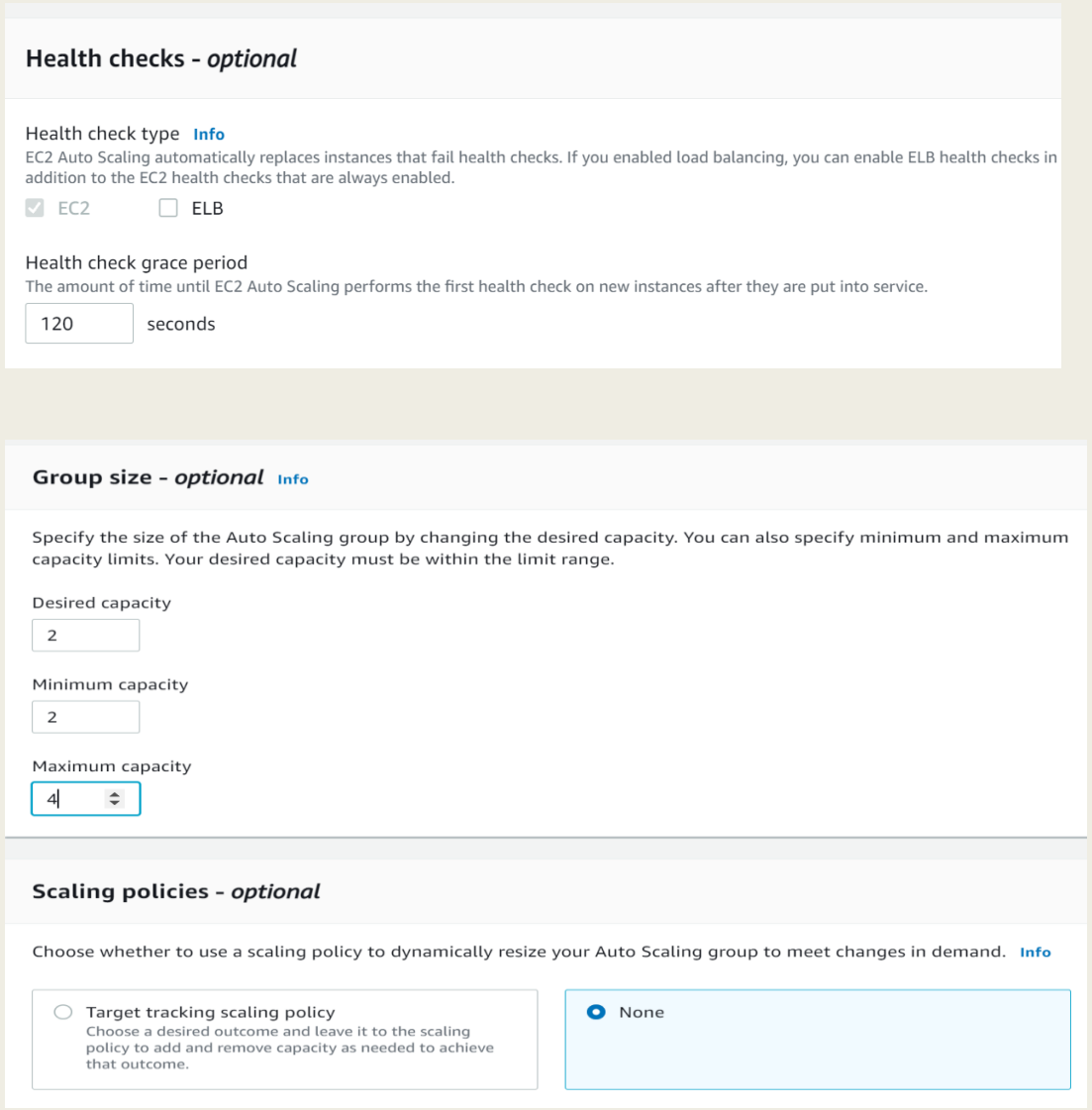




Attach existing Load Balancer



Reduce health check duration to 120seconds



Review Info

Step 1: Choose launch template or configuration

Edit

Group details

Auto Scaling group name

Poonam-Proj-ASgroup

Launch configuration

Poonam-proj-LC

Step 2: Configure settings

Edit

Network

Network

VPC

vpc-0bc403c54a7f58d37

Availability Zone	Subnet	
ap-south-1a	subnet-0fff4a10f08254692	10.0.1.0/24
ap-south-1b	subnet-04cc3960bdc6a942a	10.0.2.0/24
ap-south-1c	subnet-0c3e1c43030737770	10.0.3.0/24

Step 3: Configure advanced options

Edit

Load balancing

Load balancer 1

Name	Type	Target group
ProjectpoonamLB	Application/HTTP	Poonam-Proj-TG

Health checks

Health check type	Health check grace period
EC2	120 seconds

Additional settings

Monitoring

Disabled

Step 4: Configure group size and scaling policies — Edit

Group size

Desired capacity	Minimum capacity	Maximum capacity
2	2	4

Scaling policy

No scaling policy

Instance scale-in protection

Instance scale-in protection

☐ Enable instance protection from scale in

Step 5: Add notifications — Edit

Notifications

No notifications

Step 5: Add notifications — Edit

Notifications

No notifications

Step 6: Add tags — Edit

Tags (1)

Key	Value	Tag new instances
Name	Proj22	Yes

Cancel | **Create Auto Scaling group**

Poonam-Proj-ASgroup created successfully

EC2 > Auto Scaling groups

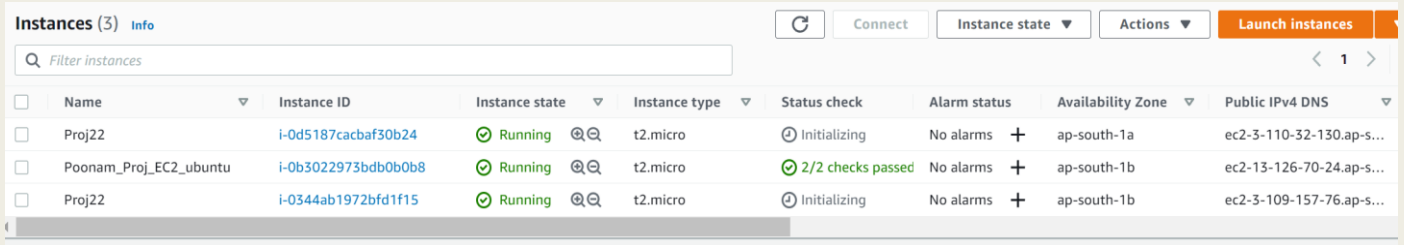
Auto Scaling groups (1) — Edit | Delete | Create an Auto Scaling group

Search your Auto Scaling groups

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
Poonam-Proj-ASgroup	Poonam-proj-LC	0	Updating capacity	2	2	4	ap-south-1c, ap-south-1...

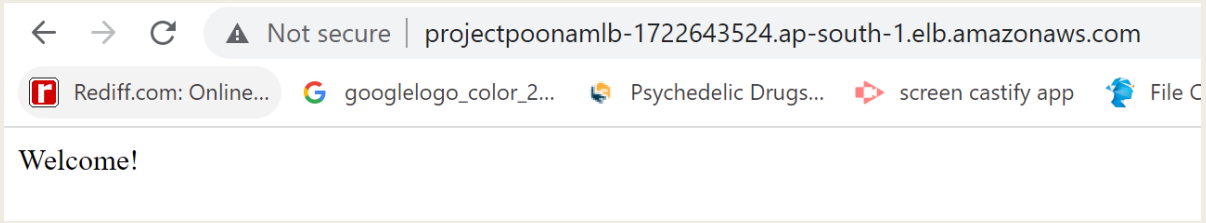
Successfully created Autoscaling group.....!!!

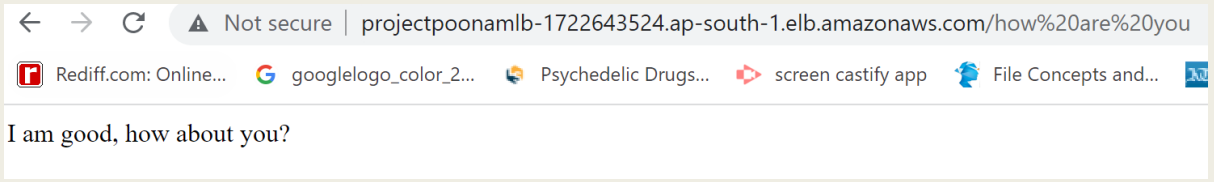
Go to the EC2 Instances, we will find two new instance created

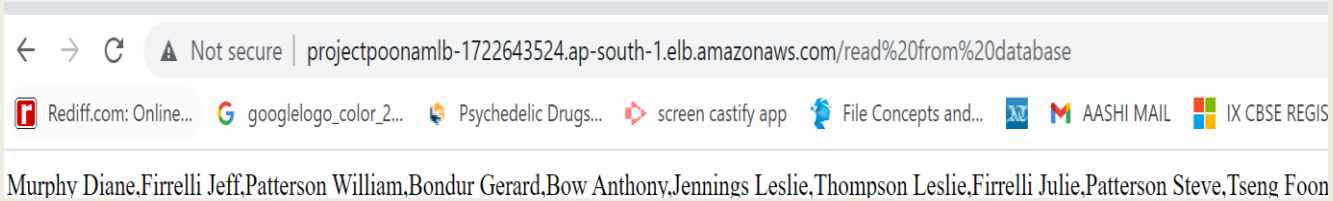


Terminated the first instance created

Now copy the DNS name of the Load Balancer, and paste in the browser, the output will be

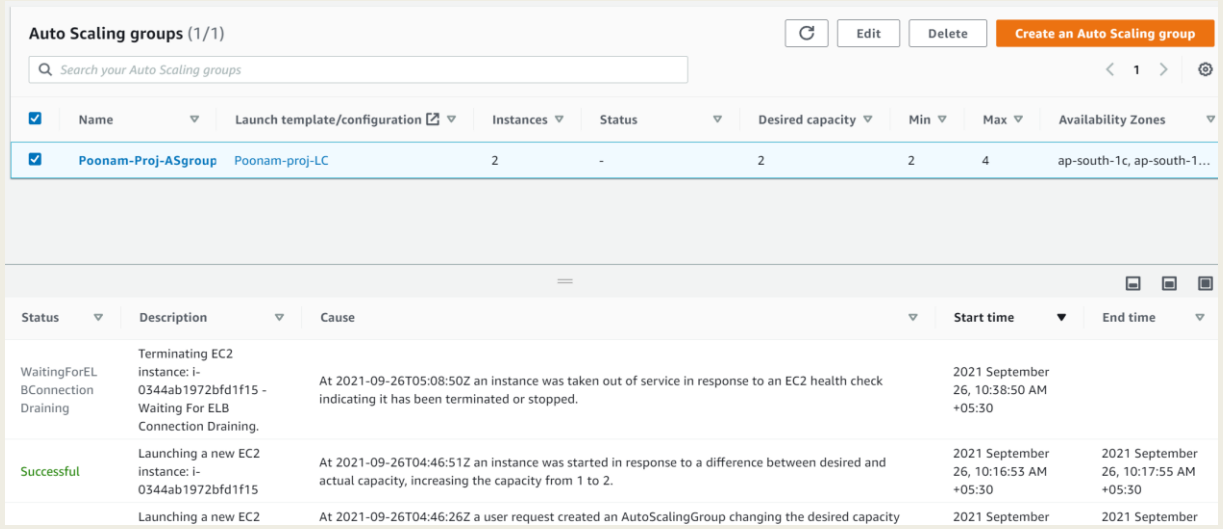




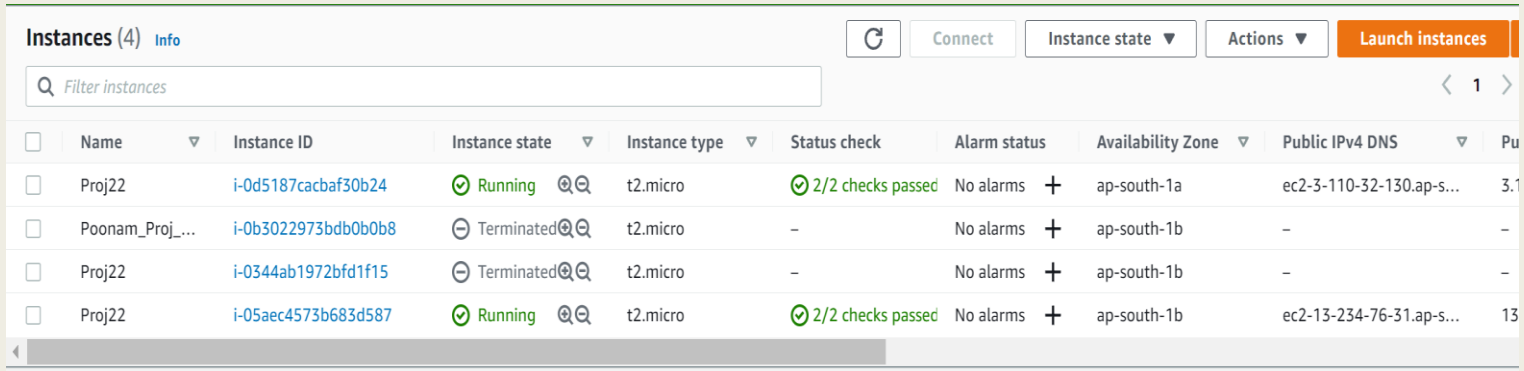


ProjectpoonamLB-1722643524.ap-south-1.elb.amazonaws.com	Welcome!
ProjectpoonamLB-1722643524.ap-south-1.elb.amazonaws.com/how are you	I am good, how about you?
ProjectpoonamLB-1722643524.ap-south-1.elb.amazonaws.com/read from database	<List of employee names>

Delete one more instance then when we will check the Auto scaling group, it will show the status as below:



We observe that one more EC2 instance is created due to Auto Scaling, below is the output



This indicates that the application is properly configured and the application works correctly accessing the database.

**********END OF IMPLEMENTATION**********

General Queries by Management

A. How can you make sure that the RDS is accessible only from any EC2 instance and not from your laptop or other IP addresses?

Ans: RDS will only be accessible from any instance within the VPC as while giving the inbound rule, I have hardcore the VPC CIDR value i.e. **10.0.0.0/16**

B. What are the dependencies you need to install for the Flask Python application to run?

Ans: Dependencies that was needed to install for the Flask Python Application to run was WinSCP