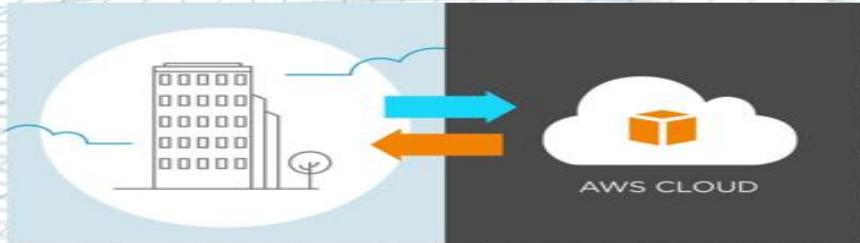


# **CAPSTONE PROJECT**

By – Poonam Sharma

PGCP CC Batch - 12

# Capstone Project



*By*  
**Poonam Sharma**  
*(PGCP Batch-12)*

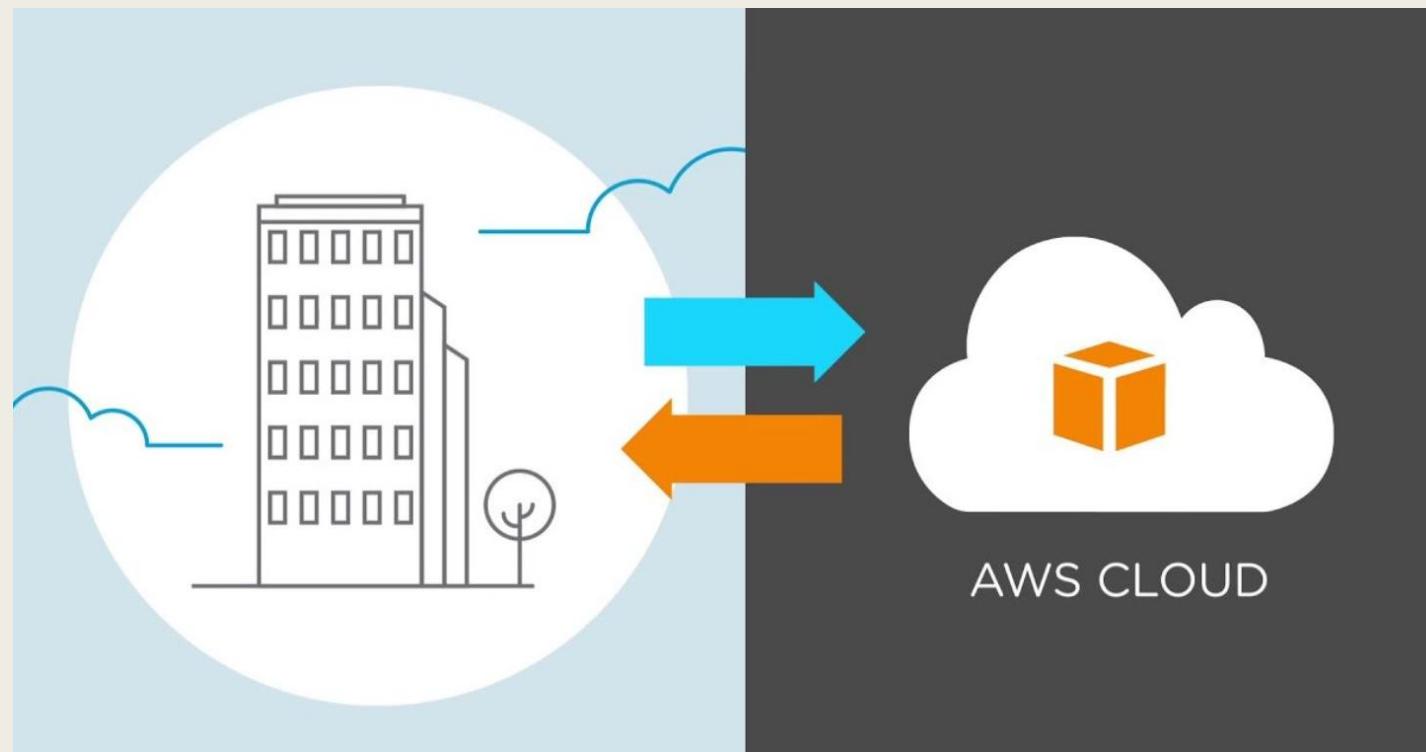
## *Application Migration to Cloud Platform*

28 September 2021

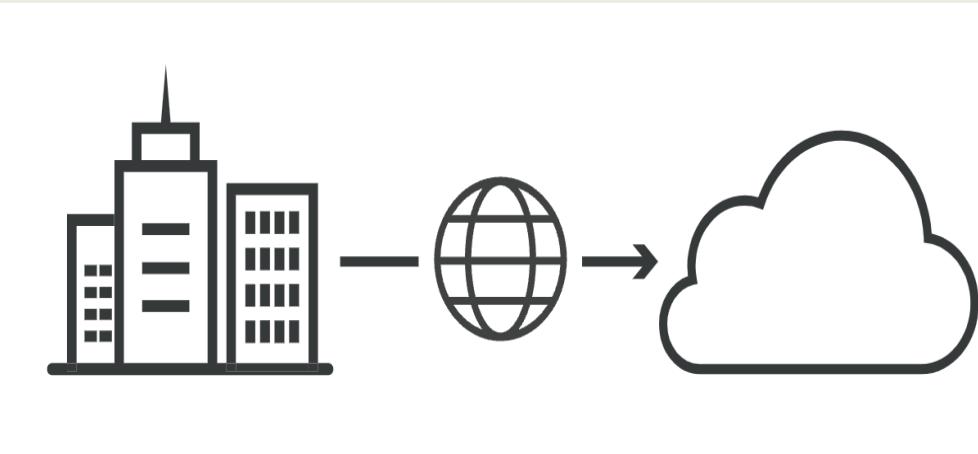
Submitted to

**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:**

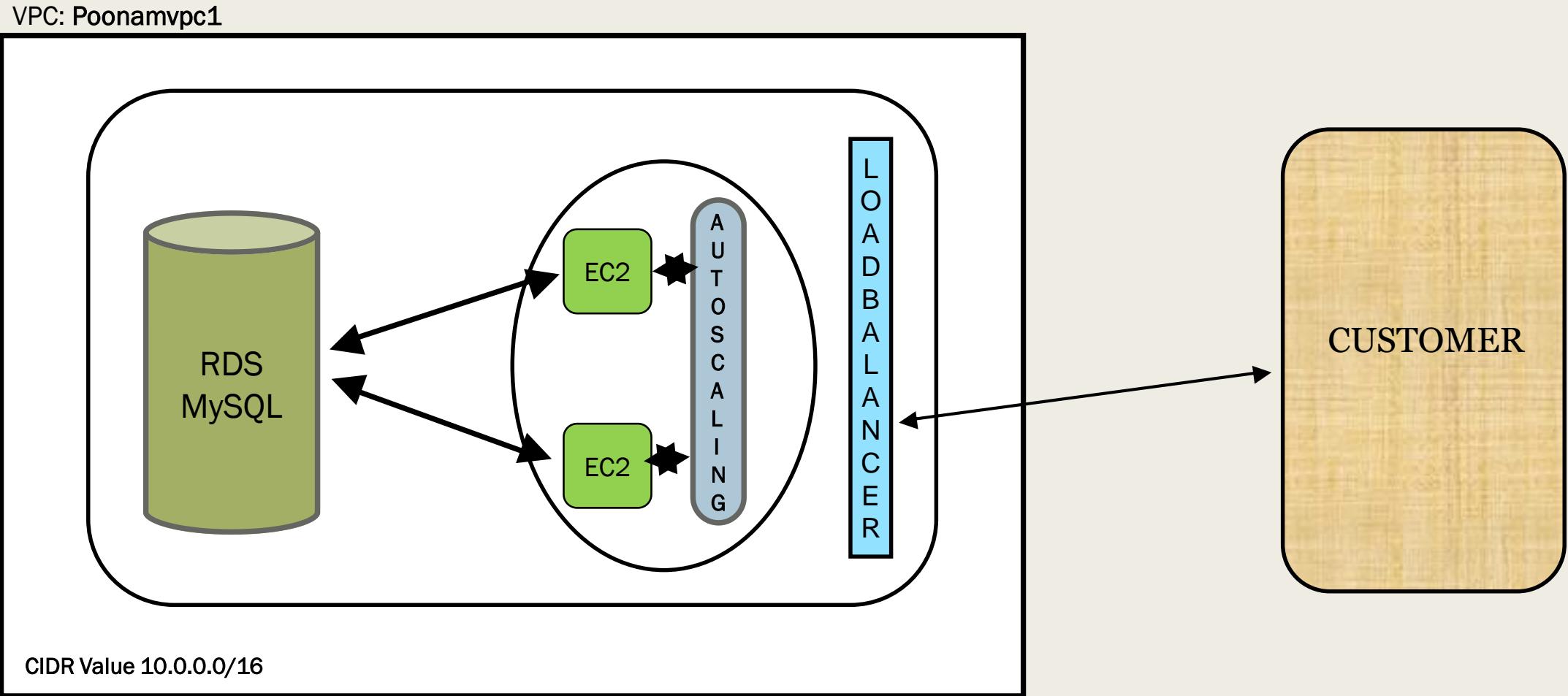
To migrate the on-premise application to the public cloud thereby reducing the load on their application

## **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**

**VPC settings**

Name tag - *optional*  
Creates a tag with a key of 'Name' and a value that you specify.

Poonamvpc1

IPv4 CIDR block [Info](#)

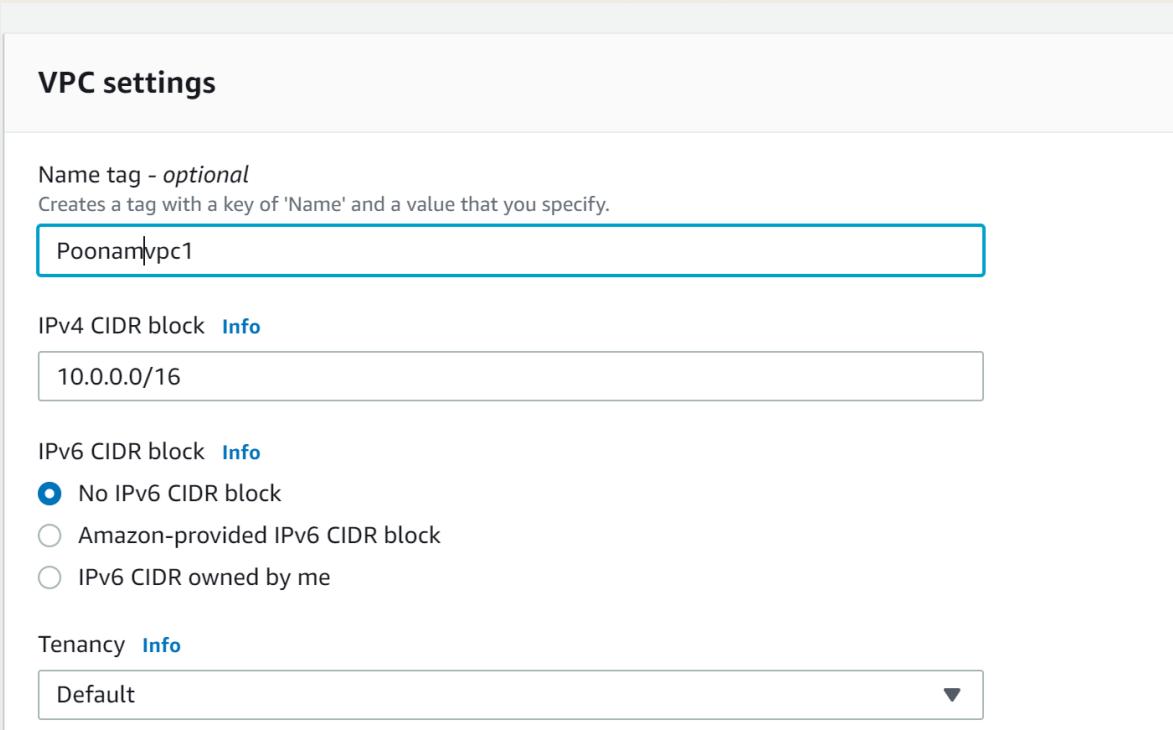
10.0.0.0/16

IPv6 CIDR block [Info](#)

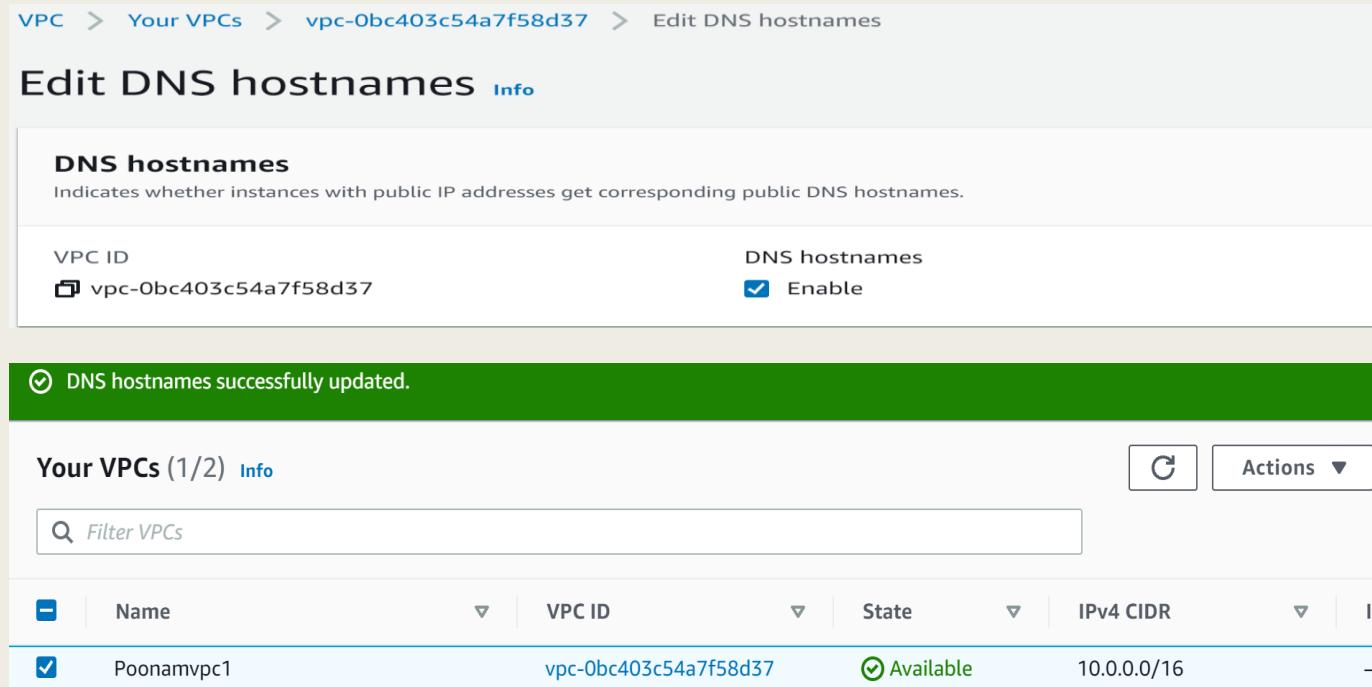
No IPv6 CIDR block  
 Amazon-provided IPv6 CIDR block  
 IPv6 CIDR owned by me

Tenancy [Info](#)

Default ▾



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



## Create Subnets **Poonam\_subnet1, Poonam\_subnet2, Poonam\_subnet3**

**Subnet 1 of 1**

**Subnet name**  
Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

**Availability Zone** [Info](#)  
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

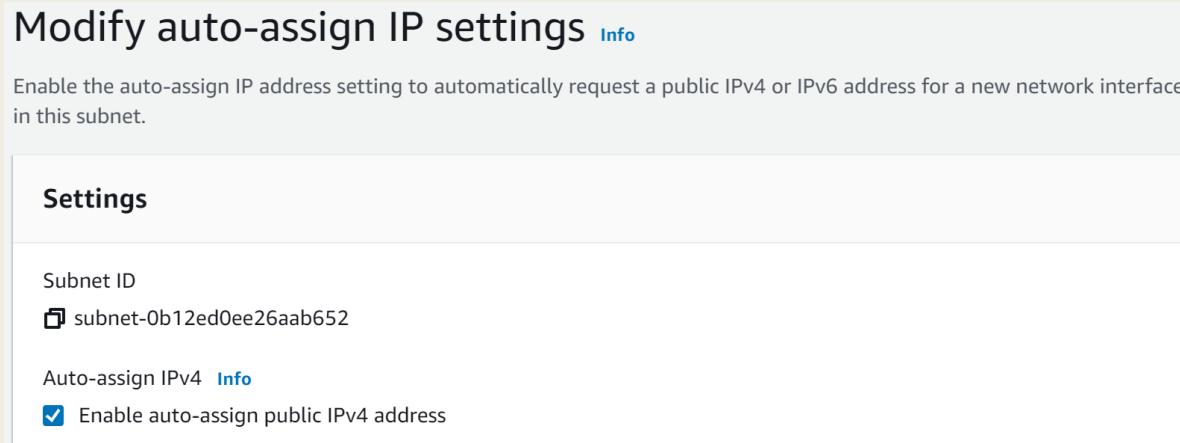
**IPv4 CIDR block** [Info](#)

Similarly create 2 more subnets

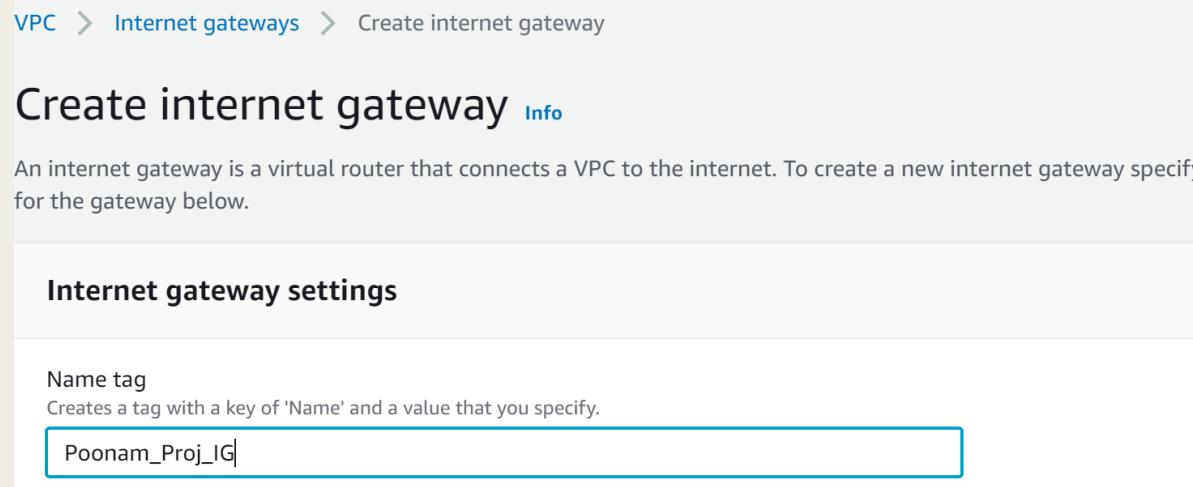
All three Subnets up and running

Subnets (3) <a href="#">Info</a>						
<input type="text"/> Filter subnets						
<input type="text" value="search: poonam"/> <a href="#">X</a> <a href="#">Clear filters</a>						
<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR	
<input type="checkbox"/>	Poonam_subnet2	subnet-04cc3960bdc6a942a	<span>Available</span>	vpc-0bc403c54a7f58d37   Poo...	10.0.2.0/24	
<input type="checkbox"/>	Poonam_subnet1	subnet-0fff4a10f08254692	<span>Available</span>	vpc-0bc403c54a7f58d37   Poo...	10.0.1.0/24	
<input type="checkbox"/>	Poonam_subnet3	subnet-0c3e1c43030737770	<span>Available</span>	vpc-0bc403c54a7f58d37   Poo...	10.0.3.0/24	

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**

Name	Internet gateway ID	State	VPC ID
Poonam_Proj_IG	igw-066bd8f0cb96b8fb1	Detached	-
-	igw-0fb32c67	Attached	vpc-69123456

## Create Route Table: **Poonam\_Proj\_RT**

**Create route table** Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

**Route table settings**

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.

Poonam\_Proj\_RT

**VPC**  
The VPC to use for this route table.

vpc-0bc403c54a7f58d37 (Poonamvpc1)

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

The screenshot shows two consecutive steps in the AWS VPC console for editing route tables.

**Step 1:** The user is adding a new route. In the "Destination" field, they type "0.0.0.0/0". In the "Target" field, they type "local" and then click on the dropdown menu to search for a target. The dropdown shows "Egress Only Internet Gateway", "Gateway Load Balancer Endpoint", "Instance", and "Internet Gateway".

**Step 2:** The user has selected "Internet Gateway" from the dropdown. The "Target" field now contains "igw-066bd8f0cb96b8fb1".

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

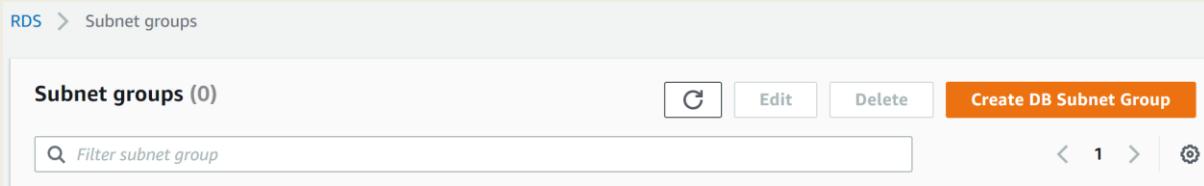
The screenshot shows the "Edit subnet associations" interface. The user has selected all three available subnets for association with the route table.

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
Poonam_Subnet1	subnet-0b12ed0ee26aab652	10.0.1.0/24	-	Main (rtb-0f0f3c932de28faee)
Poonam_Subnet3	subnet-0a0a9d20ddfc2efd	10.0.3.0/24	-	Main (rtb-0f0f3c932de28faee)
Poonam_Subnet2	subnet-04b20e91b6060ad8a	10.0.2.0/24	-	Main (rtb-0f0f3c932de28faee)

# 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: **Poonamvpc1**

A screenshot of the 'Create DB Subnet Group' wizard. The title is 'Create DB Subnet Group'. Below it, a note says: 'To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.' The 'Subnet group details' section contains fields for 'Name' (set to 'Poonam\_Proj\_RDSSubGrp'), 'Description' (set to 'Project RDS Subnet Group'), and 'VPC' (set to 'Poonamvpc1 (vpc-0bc403c54a7f58d37)').

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

The screenshot shows the 'Add subnets' section of the AWS Subnet Selection interface. It lists subnets grouped by availability zone:

- ap-south-1b: subnet-04cc3960bdc6a942a (10.0.2.0/24)
- ap-south-1a: subnet-0fff4a10f08254692 (10.0.1.0/24)
- ap-south-1c: subnet-0c3e1c43030737770 (10.0.3.0/24)

A dropdown menu labeled 'Select subnets' is open, showing the same three subnets again, each with a delete icon (X). Below this is a table titled 'Subnets selected (3)' with columns: Availability zone, Subnet ID, and CIDR block.

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

## 6. Successfully created RDS Subnet group **poonam\_proj\_rdssubgrp**

The screenshot shows the 'Subnet groups' page in the AWS RDS console. A green success message at the top says 'Successfully created Poonam\_Proj\_RDSSubGrp. View subnet group'. The main table displays one subnet group:

Name	Description	Status	VPC
poonam_proj_rdssubgrp	Project RDS Subnet Group	Complete	vpc-0bc403c54a7f58d37

## 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**

### Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create below.

#### Basic details

Security group name Info

Poonam\_proj\_RDS\_SG

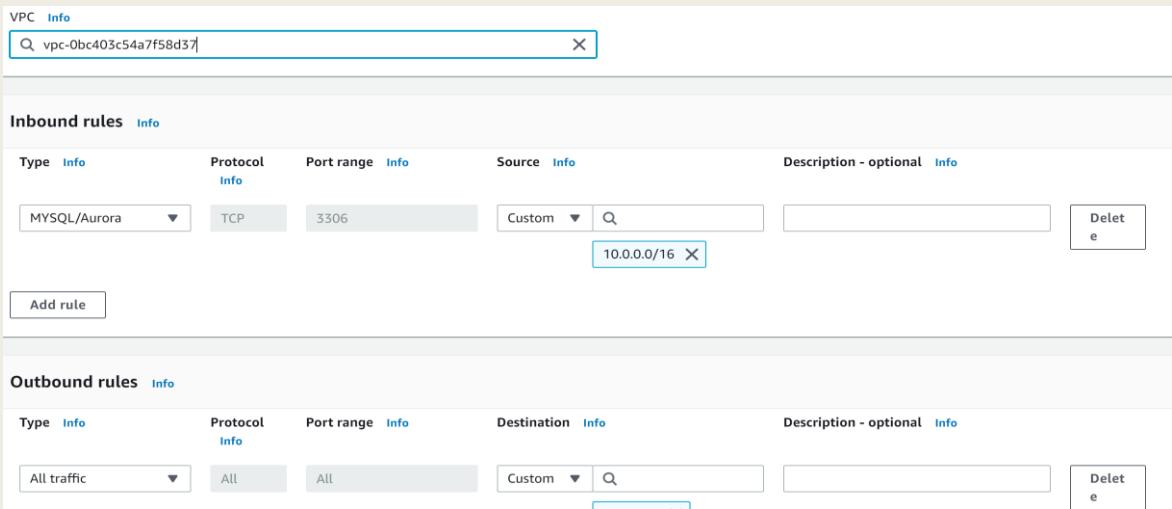
Name cannot be edited after creation.

Description Info

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



The screenshot shows the AWS VPC Security Group Inbound Rules configuration page. At the top, there's a search bar with the value "vpc-0bc403c54a7f58d37". Below the search bar, the title "Inbound rules" is followed by a table header with columns: Type, Protocol, Port range, Source, and Description - optional. A single rule is listed: "Type: MYSQL/Aurora, Protocol: TCP, Port range: 3306, Source: Custom 10.0.0.0/16, Description: optional". There are "Add rule" and "Delete" buttons at the bottom of the table. Below the table, another section titled "Outbound rules" is partially visible.

### 3. Security group Poonam\_proj\_RDS\_SG is successfully created

sg-0931779b77bf12ba8 - Poonam\_proj\_RDS\_SG Actions ▾

Details			
Security group name <a href="#">Poonam_proj_RDS_SG</a>	Security group ID <a href="#">sg-0931779b77bf12ba8</a>	Description <a href="#">Project RDS Security Group</a>	VPC ID <a href="#">vpc-0bc403c54a7f58d37</a>
Owner <a href="#">160112582475</a>	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

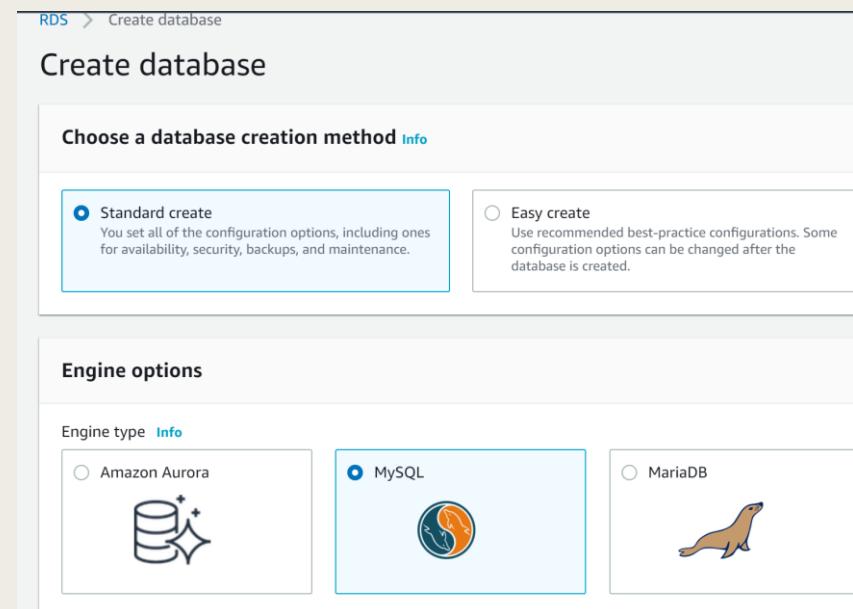
[Inbound rules](#) [Outbound rules](#) [Tags](#)

i You can now check network connectivity with Reachability Analyzer [Run Reachability Analyzer](#) X

Inbound rules (1/1)					
<input type="button" value="C"/>	<a href="#">Manage tags</a>	<a href="#">Edit inbound rules</a>			
<input type="text" value="Filter security group rules"/> <span style="float: right;">&lt; 1 &gt; <span style="font-size: small;">@</span></span>					
<input checked="" type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol
<input checked="" type="checkbox"/>	Poonam_proj_R...	<a href="#">sgr-091d214266e086...</a>	IPv4	MYSQL/Aurora	TCP

## 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**

The screenshot shows the 'Settings' step of the 'Create database' wizard. It includes fields for the DB instance identifier (set to 'Project\_database\_1'), master username ('root'), master password ('poonam22'), and confirm password ('poonam22'). The master password field has a note indicating it must be at least 8 printable ASCII characters and cannot contain specific punctuation.

DB instance identifier	<input type="text" value="Project_database_1"/>
Master username	<input type="text" value="root"/>
Master password	<input type="password" value="poonam22"/>
Confirm password	<input type="password" value="poonam22"/>

## Select VPC and Subnet group

**Connectivity**

**Virtual private cloud (VPC) [Info](#)**  
VPC that defines the virtual networking environment for this DB instance.

Poonamvpc1 (vpc-0bc403c54a7f58d37) ▾  
Only VPCs with a corresponding DB subnet group are listed.

**After a database is created, you can't change its VPC.**

**Subnet group [Info](#)**  
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

poomam\_proj\_rdssubgrp ▾

**Public access [Info](#)**

Yes  
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

No  
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

## Select VPC Security Group

**VPC security group**  
Choose a VPC security group to allow access to your database. Ensure 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 VPC security groups ▾  
Poonam\_proj\_RDS\_SG X

**Availability Zone [Info](#)**  
No preference ▾

**Additional configuration**

**Database port [Info](#)**  
TCP/IP port that the database will use for application connections.  
3306

A screenshot of the AWS RDS Databases page. At the top, a blue header bar displays a message: "Creating database **poonamproject-database-1**. Your database might take a few minutes to launch." To the right of the message are "View credential details" and a close button ("X"). Below the header, the page title is "RDS > Databases". The main content area is titled "Databases" and contains a table with the following columns: DB identifier, Role, Engine, Region & AZ, Size, Status, CPU, Current activity, and Maint. A search bar labeled "Filter databases" is positioned above the table. The table shows one row for the database "poonamproject-database-1", which is listed as an "Instance" using "MySQL Community" engine, located in "Region & AZ: -" with "Size: db.t2.micro", "Status: Creating", and "CPU: -". The "Maint." column shows "none".

**Now the Database is getting created.....**

A screenshot of the AWS RDS Databases page. At the top, a green header bar displays a message: "Successfully created database **poonamproject-database-1**". To the right of the message are "View connection details" and a close button ("X"). Below the header, the page title is "RDS > Databases". The main content area is titled "Databases" and contains a table with the same columns as the previous screenshot. The table now shows the database "poonamproject-database-1" in a different state: it is listed as an "Instance" using "MySQL Community" engine, located in "Region & AZ: ap-south-1b" with "Size: db.t2.micro", "Status: Available", and "CPU: -".

**Connection details to your database poonamproject-database- 1**

This is the only time you will be able to view this password. Copy and save the password for your reference, otherwise you will need to modify the database to change it. You can use a SQL client application or utility to connect to your database.

[Learn about connecting to your database](#)

Master username  
root

Master password  
poonam22 [Copy](#)

Endpoint  
poonamproject-database-1.c8teywnldyjo.ap-south-1.rds.amazonaws.com [Copy](#)

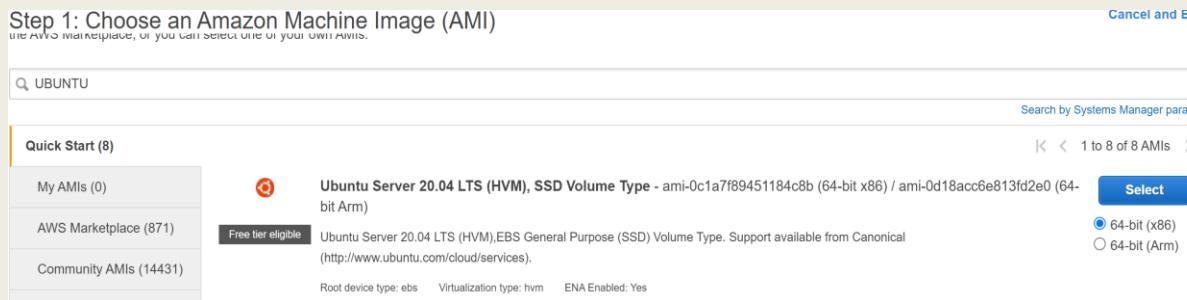
Copy endpoint:

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)	EBS Optimized
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	
<input checked="" type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	

### 3. Configure instances

Step 3: Configure instance details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of lower prices, or choose a different instance type.

Number of instances	<input type="text" value="1"/>	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-0bc403c54a7f58d37   Poonamvpc1	<input type="button" value="Create new VPC"/>
Subnet	subnet-04cc3960bdc6a942a   Poonam_subnet2   ap-south-1	<input type="button" value="Create new subnet"/> 250 IP Addresses available
Auto-assign Public IP	<input type="button" value="Use subnet setting (Enable)"/>	

#### 4. Paste below script in “User data”

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

Advanced Details

Enclave	<input type="checkbox"/> Enable
Metadata accessible	Enabled
Metadata version	V1 and V2 (token optional)
Metadata token response hop limit	1
User data	<input checked="" type="radio"/> As text <input type="radio"/> As file <input type="checkbox"/> Input is already base64 encoded
<pre>#!/bin/bash sudo apt-get update sudo apt-get install apache2 -y</pre>	

#### 5. Add Tags

Name: Poonam\_Proj\_EC2\_ubuntu

##### Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	(128 characters maximum)	Value	(256 characters maximum)	Instances	Volumes	Network Interface
Name		Poonam_Proj_EC2_ubuntu		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

#### 6. Create Security Group: Project\_WebAccess\_SG

Open Port

SSH	TCP	22	Anywhere
HTTP	TCP	80	Anywhere

##### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon security groups.

Assign a security group:  Create a new security group

Select an existing security group

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere	0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere	0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop

## 7. Using Git Bash terminal , connect Instance just created

The screenshot shows the 'Connect to instance' page for an EC2 instance. At the top, it says 'Connect to your instance i-0b3022973bdb0b0b8 (Poonam\_Proj\_EC2\_ubuntu) using any of these options'. Below this, there are four tabs: 'EC2 Instance Connect', 'Session Manager', 'SSH client' (which is highlighted in orange), and 'EC2 Serial Console'. Under the 'SSH client' section, it says '4. Connect to your instance using its Public DNS:' followed by a copy icon and the IP address 'ec2-13-126-70-24.ap-south-1.compute.amazonaws.com'. Below that, it says 'Example:' followed by another copy icon and the command 'ssh -i "Project22.pem" ubuntu@ec2-13-126-70-24.ap-south-1.compute.amazonaws.com'.

## 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

The terminal window shows the following output:

```
ubuntu@ip-10-0-2-158:~$ mysql
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, or
sudo apt install mariadb-client-core-10.3  # version 1:10.3.31-0ubuntu0.20.04.1
ubuntu@ip-10-0-2-158:~$ sudo apt install mysql-client-core-8.0
```

## 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.c8teywn1dyjo.ap-south-1.rds.amazonaws.com -uroot -ppoonam22
```

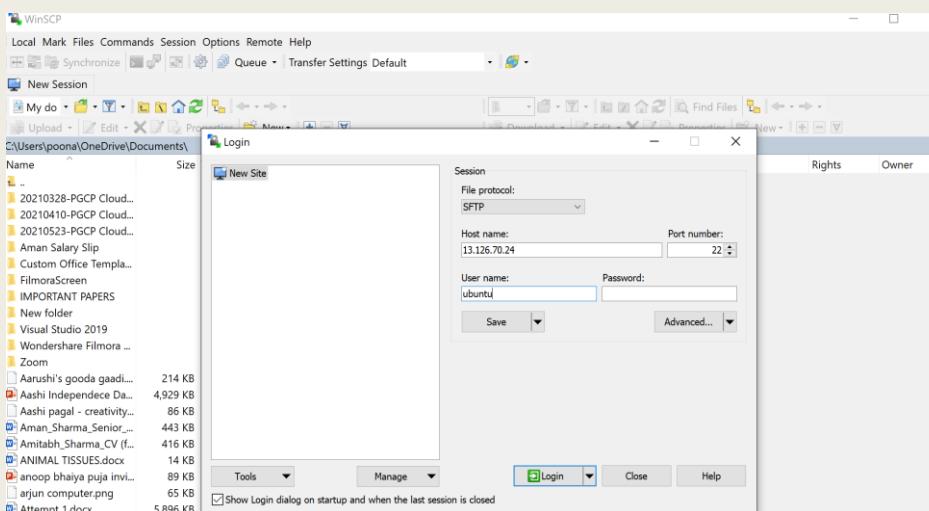
We now land on MySQL prompt

```
$ mysql> show databases
```

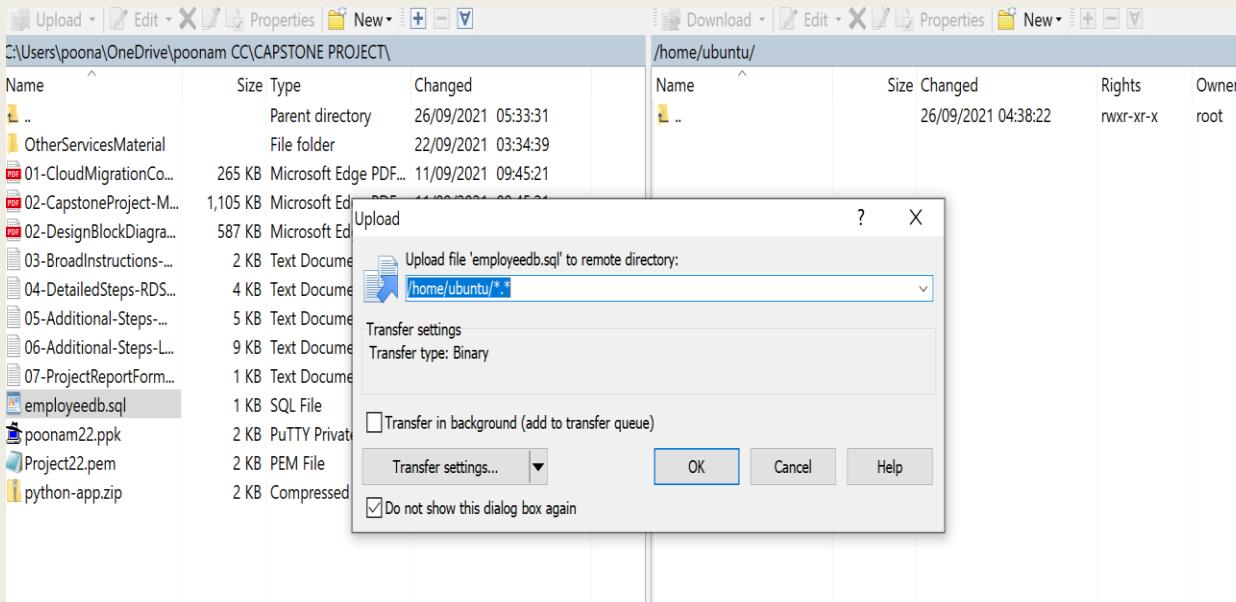
```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.00 sec)

mysql> |
```

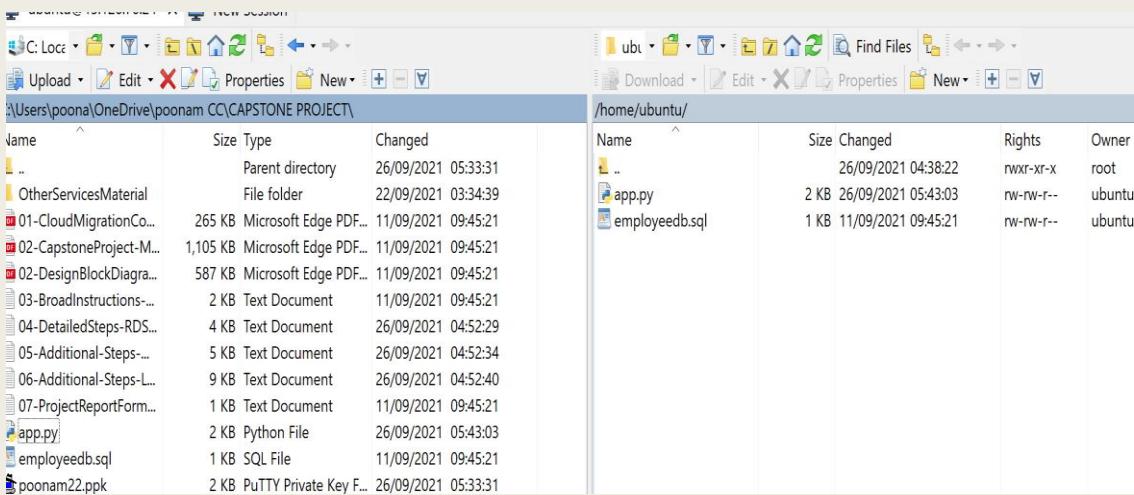
## 10. Open WinSCP



## 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 statement
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 <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)
```

```
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)
```

## TASK 5: Configuring and testing Python-Flask application

Normally on the Ubuntu EC2 instance 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/dist-packages  
(from Flask->flask-mysql) (2.0.1)  
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.8/dist-packages  
(from Flask->flask-mysql) (3.0.1)  
Requirement already satisfied: click>=7.1.2 in /usr/local/lib/python3.8/dist-packages  
(from Flask->flask-mysql) (8.0.1)  
Requirement already satisfied: werkzeug>=2.0 in /usr/local/lib/python3.8/dist-packages  
(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)

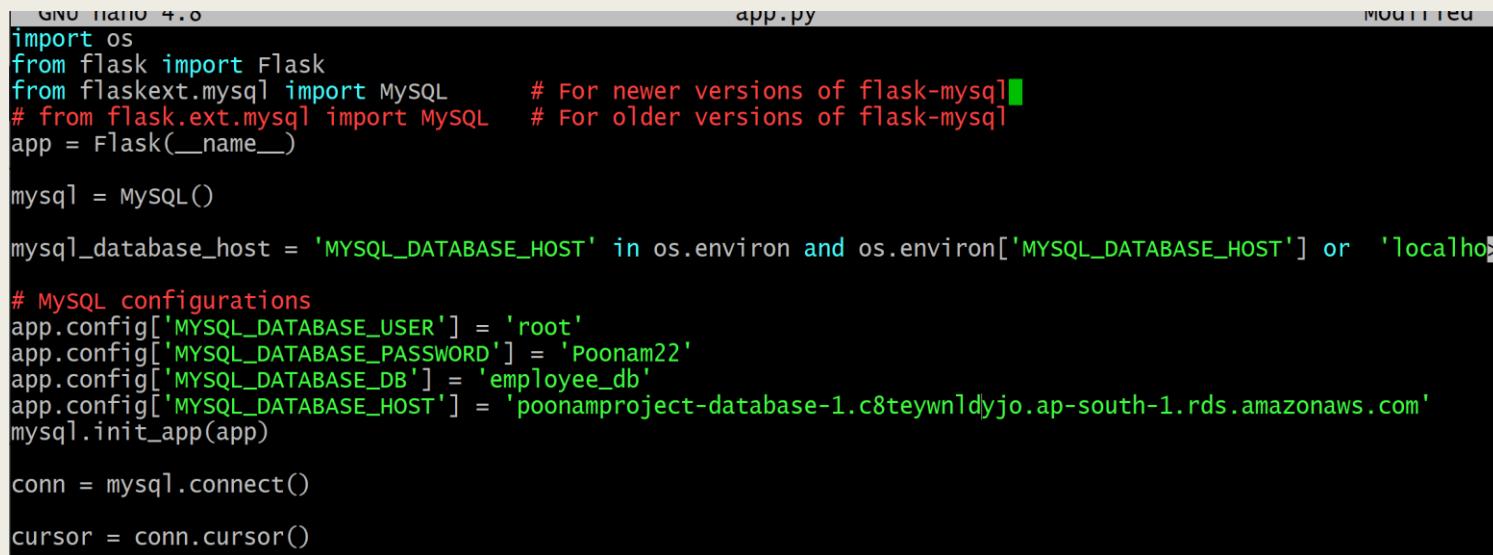
### Nano app.py

```
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 application 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>'
```



```
GNU nano 4.0          app.py          MOUNTED  
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()
```

### 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.c8teywn1dyjo.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 inbound traffic on Custom TCP on port number 5000.

you EC2 instance to allow

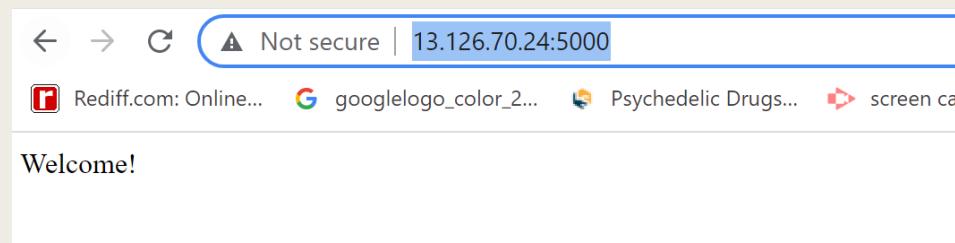
#### Add rules

Custom TCP	5000
Custom TCP	5000

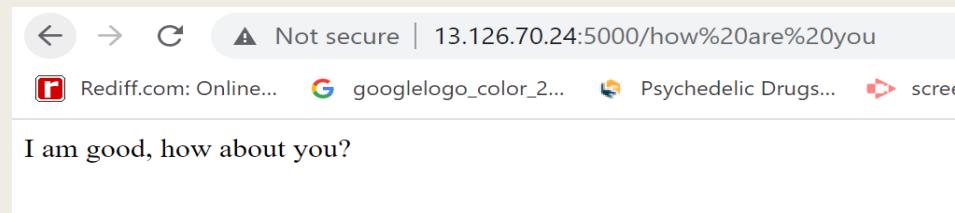
Anywhere IPv4
Anywhere IPv6

9. Now paste this in the browser

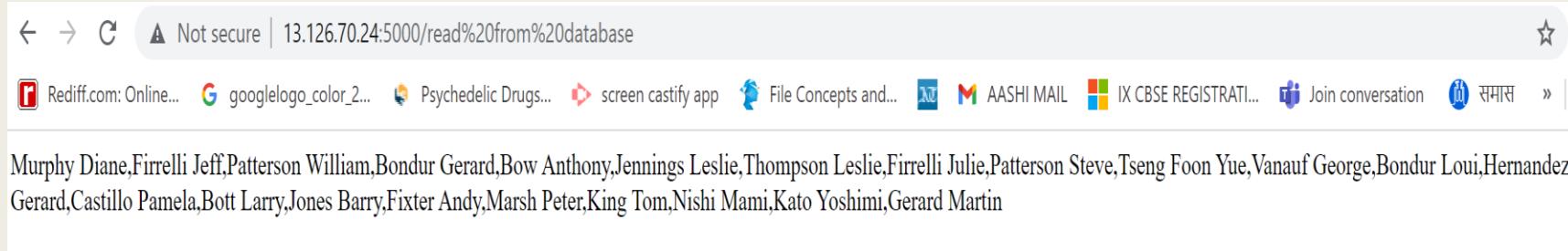
a. <http://13.126.70.24:5000/>



b. [13.126.70.24:5000/how%20are%20you](http://13.126.70.24:5000/how%20are%20you)



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
```

- 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:~$ 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:~$ |
```

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

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

```
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:~$ |
```

```
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 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>
<-->
```

```
GNU nano 4.8                               /etc/apache2/sites-available/FlaskApp.conf
<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 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 command 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'
```

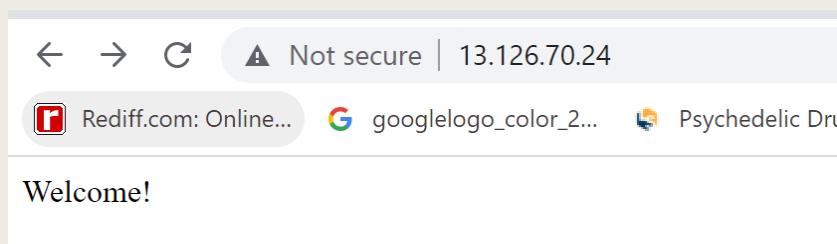
```
GNU nano 4.8                               /var/www/FlaskApp/flaskapp.wsgi
#!/usr/bin/python
import sys
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0,"/var/www/FlaskApp/")

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

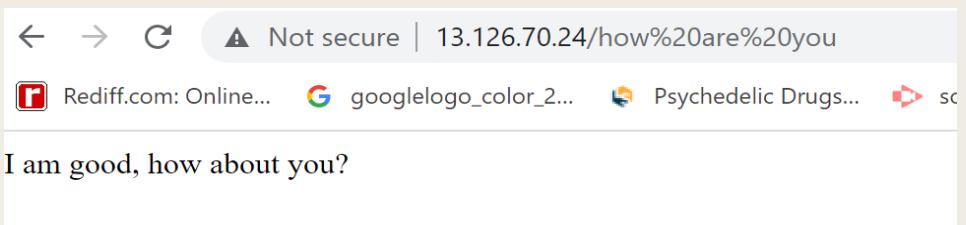
3. Finally restart apache2 server.

```
$ sudo service apache2 restart
```

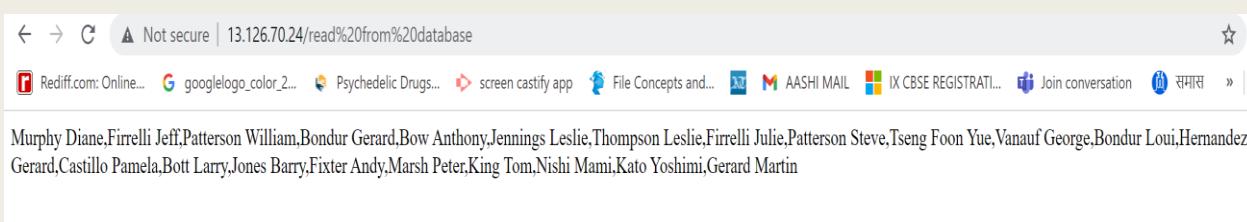
- a) Copy public IP of EC2 instance and paste on browser, we will get this output



b) Public\_IP/how are you will give below output



c) Public\_IP/read from database



## 1. Create a Load Balancer (Application Load Balancer)

Learn more about which load balancer is right for you

### Application Load Balancer

Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application

**Create**

▶ How Application Load Balancers work

#### Basic configuration

Load balancer name  
Name must be unique within your AWS account and cannot be changed after the load balancer is created.

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info  
Scheme cannot be changed after the load balancer is created.

Internet-facing  
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more

Internal  
An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type Info  
Select the type of IP addresses that your subnets use.

IPv4

confirm the VPC for your targets, view your [target groups](#)

Poonamvpc1  
vpc-0bc403c54a7f58d37  
IPv4: 10.0.0.0/16

Mappings Info  
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. The load balancer or the VPC are not available for selection. Subnets cannot be removed after the load balancer is created, but additional subnets can be added if supported by the load balancer or the VPC are disabled. At least two subnets must be specified.

ap-south-1a

Subnet  
subnet-0ff4a10f08254692 Poonam\_subnet1

ap-south-1b

Subnet  
subnet-04cc3960bdc6a942a Poonam\_subnet2

ap-south-1c

Subnet  
subnet-0c3e1c43030737770 Poonam\_subnet3

**Security group name : Poonam\_Proj\_LB**  
Description: Project Load balancer

Security group name [Info](#)  
Poonam\_Proj\_LB  
Name cannot be edited after creation.

Description [Info](#)  
Project Load balancer

VPC [Info](#)  
 X

Inbound rules [Info](#)

Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional
HTTP	TCP	80	Anywhere... <input type="text" value="0.0.0.0"/> <span>X</span>	
HTTP	TCP	80	Anywhere... <input type="text" value="::/0"/> <span>X</span>	

Security groups [Info](#)  
A security group is a set of firewall rules that control the traffic to your load balancer.

Security groups  
 G

Create new security group [Create new security group](#)

Poonam\_Proj\_LB sg-02df99a6121f09018 X  
VPC: vpc-0bc403c54a7f58d37

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.

Choose a target type

Instances

- Supports load balancing to instances within a specific VPC.

IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.

Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Target group name  
  
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol Port  
 :

VPC  
Select the VPC with the instances that you want to include in the target group.  
 ▼

Protocol version  
 HTTP1  
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

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

The screenshot shows the 'Target groups' section of the AWS EC2 console. A green header bar at the top indicates 'Successfully created target group: Poonam-Proj-TG'. Below this, the 'Target groups (1) info' table lists one item:

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
Poonam-Proj-TG	arn:aws:elasticloadbalancing:... 80	HTTP	Instance	-	vpc-0bc403c54a7f	

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

The screenshot shows the 'Listeners and routing' section of the AWS Load Balancers console. It displays a single listener configuration for 'Listener HTTP:80':

Protocol	Port	Default action
HTTP	80	Forward to Poonam-Proj-TG Target type: Instance

A blue link labeled 'Create target group' is visible below the listener configuration.

The screenshot shows the 'Load balancers' section of the AWS Load Balancers console. A green header bar at the top indicates 'Successfully created load balancer: ProjectpoonamLB'. Below this, the 'Load balancers' table lists one item:

Name	Type	Status
ProjectpoonamLB	Application Load Balancer	Active

A blue box titled 'Suggested next steps' contains two items:

- Review, customize, or enable attributes for your load balancer and listeners using the **Description** and **Listeners** tabs within **ProjectpoonamLB**.
- Discover other services that you can integrate with your load balancer. Visit the **Integrated services** tab within **ProjectpoonamLB**.

A blue button labeled 'View load balancers' is located at the bottom right.

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

The screenshot shows the 'Listeners and routing' section of the AWS Load Balancer configuration. It displays a single listener named 'Listener HTTP:80'. The configuration includes:

- Protocol: HTTP
- Port: 80
- Default action: Info
- Forward to: Poonam-Proj-TG
- Target type: Instance
- HTTP
- Create target group

The screenshot shows the 'Load balancers' page. A success message at the top states: 'Successfully created load balancer: ProjectpoonamLB'. Below it, a note says: 'Note: It might take a few minutes for your load balancer to be fully set up and ready to route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.' The page lists one load balancer entry:

Name	DNS name	State	VPC ID	Availability Zones	Type
ProjectpoonamLB	ProjectpoonamLB-1722643524.ap-south-1.elb.amazonaws.com	Provisioning	vpc-0bc403c54a7f58d37	ap-south-1b, ap-south-1c	application

Suggested next steps:

- Review, customize, or enable attributes for your load balancer and listeners using the Description and Listeners tabs within ProjectpoonamLB.
- Discover other services that you can integrate with your load balancer. Visit the Integrated services tab within ProjectpoonamLB.

View load balancers

The screenshot shows the 'Basic Configuration' section of the load balancer configuration page for 'ProjectpoonamLB'. The details are:

Name	ProjectpoonamLB
ARN	arn:aws:elasticloadbalancing:ap-south-1:160112582475:loadbalancer/app/ProjectpoonamLB/4ca56744292b90de
DNS name	ProjectpoonamLB-1722643524.ap-south-1.elb.amazonaws.com (A Record)
State	Provisioning
Type	application
Scheme	internet-facing

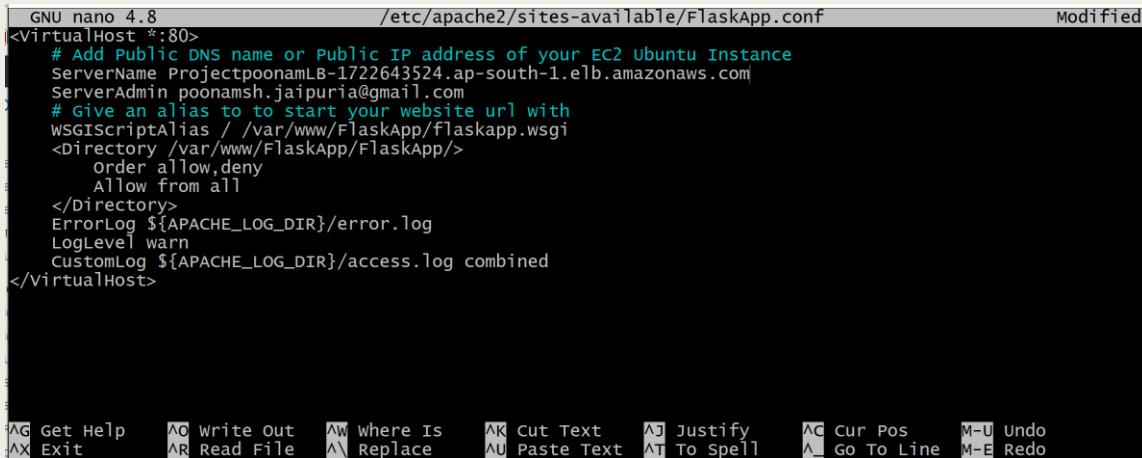
Copy the DNS name of Load Balancer  
ProjectpoonamLB-1722643524.ap-south-1.elb.amazonaws.com

## 2. CREATE AN EC2 UBUNTU INSTANCE AND ITS AMI

On your Ubuntu EC2 instance -

Update the configuration file for the FlaskApp.

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



The screenshot shows a terminal window with the command `sudo nano /etc/apache2/sites-available/FlaskApp.conf` running. The file content is displayed in the nano editor, which includes a VirtualHost block for port 80. The configuration specifies a ServerName and ServerAdmin, and defines a WSGIScriptAlias for the Flask application. The nano editor interface is visible at the bottom, showing various keyboard shortcuts.

```
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 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>
```

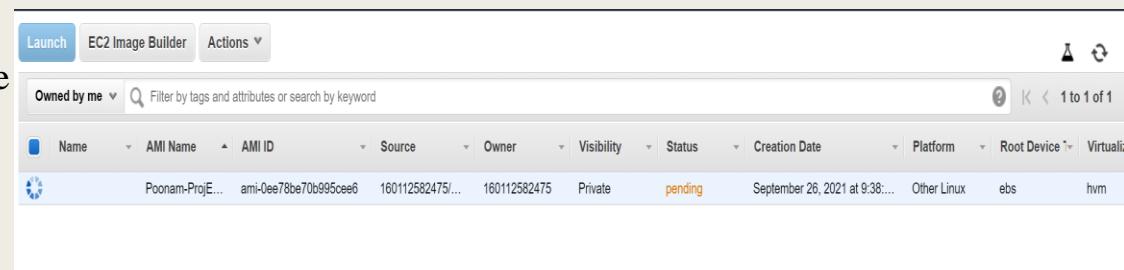
AG Get Help AO Write Out AW Where Is AK Cut Text AJ Justify AC Cur Pos M-U Undo  
AX Exit AR Read File A\ Replace AU Paste Text AT To Spell A Go To Line M-E Redo

Go to the EC2 Instance ,

Select the instance → Actions → Image and Templates → Create Image

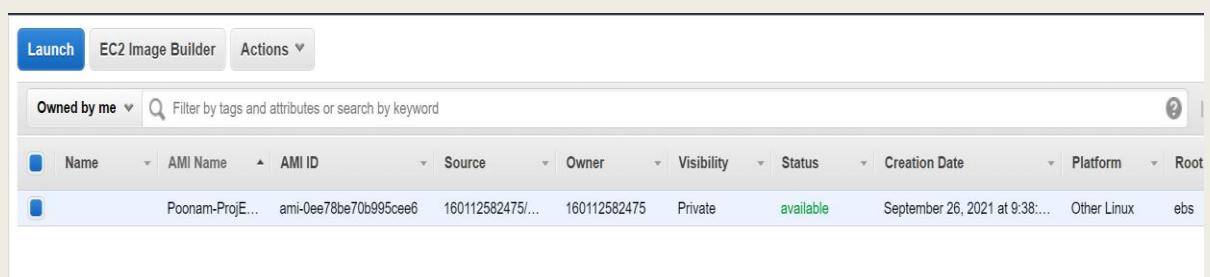
Image name: **Poonam-ProjEC2-IMG**

Image description: Project EC2 Image



The screenshot shows the AWS EC2 console with the 'Actions' tab selected. Under 'Image and Templates', the 'Create Image' option is chosen. A new AMI is being created with the following details:

Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date	Platform	Root Device	Virtualization
	Poonam-ProjE...	ami-0ee78be70b995cee6	160112582475...	160112582475	Private	pending	September 26, 2021 at 9:38...	Other Linux	ebs	hvm



The screenshot shows the AWS EC2 console with the 'Actions' tab selected. Under 'Image and Templates', the 'AMI' section is shown. The previously created AMI is now listed as available:

Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date	Platform	Root
	Poonam-ProjE...	ami-0ee78be70b995cee6	160112582475...	160112582475	Private	available	September 26, 2021 at 9:38...	Other Linux	ebs

**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

Create launch configuration [Info](#)

**Launch configuration name**

Name  
Poonam-proj-LC

**Amazon machine image (AMI) [Info](#)**

AMI  
Poonam-ProjEC2-IMG

**Instance type [Info](#)**

Instance type  
t2.micro (1 vCPUs, 1 GiB, EBS Only) [Choose instance type](#)

**Security groups [Info](#)**

Assign a security group

Create a new security group  
 Select an existing security group

**Security groups** [Copy to new](#) [View rules](#)

Search security groups [Q](#)

	Security group ID	Name	VPC ID	Description
<input type="checkbox"/>	sg-02810a7039f3667e4	default	vpc-0bc403c54a7f58d37	default VPC security group
<input checked="" type="checkbox"/>	sg-02df99a6121f09018	Poonam_Proj_LB	vpc-0bc403c54a7f58d37	Project Load balancer
<input type="checkbox"/>	sg-05cc7d645514cb4cb	launch-wizard-4	vpc-698e4302	launch-wizard-4 created 2021-08-17T16:26:47.776+05:30
<input type="checkbox"/>	sg-089474f5fc1503321	Project_WebAccess_SG	vpc-0bc403c54a7f58d37	Project_WebAccess_SG created 2021-09-26T04:31:48.077+05:30
<input type="checkbox"/>	sg-0931779b77bf12ba8	Poonam_proj_RDS_SG	vpc-0bc403c54a7f58d37	Project RDS Security Group
<input type="checkbox"/>	sg-0ad333333333333333333333333333333	Poonam_Proj_EBS_SG	vpc-698e4302	launch-wizard-3 created 2021-08-17T16:26:47.776+05:30

And select the Security Group of the Elastic Load Balancer

**Key pair (login) [Info](#)**

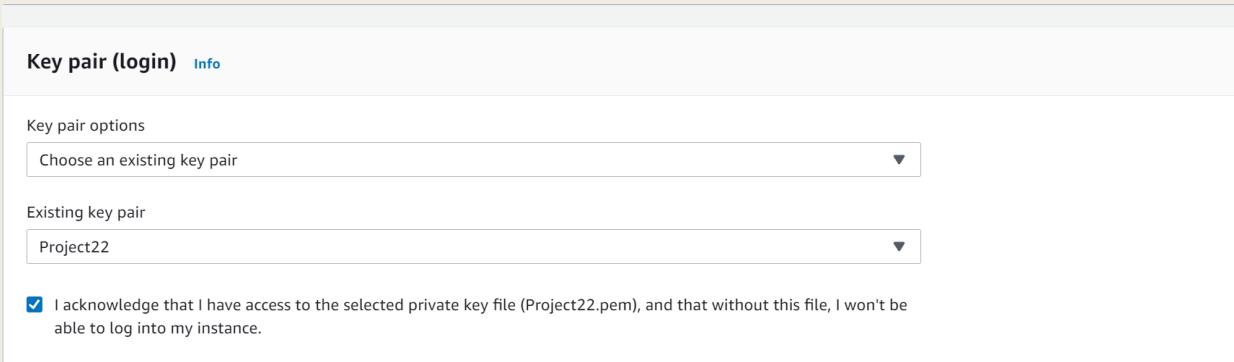
Key pair options

Choose an existing key pair ▾

Existing key pair

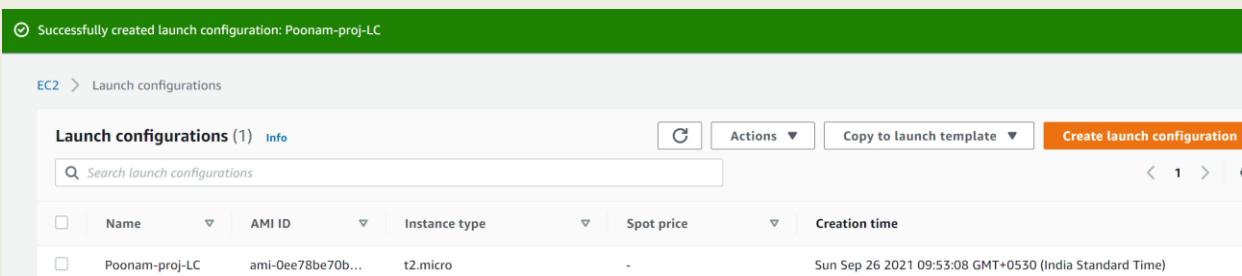
Project22 ▾

I acknowledge that I have access to the selected private key file (Project22.pem), and that without this file, I won't be able to log into my instance.



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

Successfully created launch configuration: Poonam-proj-LC						
EC2 > Launch configurations						
Launch configurations (1) <a href="#">Info</a>						
<input type="checkbox"/>	Name	AMI ID	Instance type	Spot price	Creation time	
<input type="checkbox"/>	Poonam-proj-LC	ami-0ee78be70b...	t2.micro	-	Sun Sep 26 2021 09:53:08 GMT+0530 (India Standard Time)	



**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’

**Choose launch template or configuration** Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.

Name
Auto Scaling group name Enter a name to identify the group. <b>Poonam-Proj-ASgroup</b>
Must be unique to this account in the current Region and no more than 255 characters.

**Launch configuration** Info [Switch to launch template](#)

Choose a launch configuration that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Poonam-proj-LC	<a href="#">Create a launch configuration</a>
Launch configuration	AMI ID
Poonam-proj-LC	ami-0ee78be70b995cee6
Security groups	Instance type
sg-02df99a6121f09018	t2.micro
Date created	
Sun Sep 26 2021 09:53:08 GMT+0530 (India Standard Time)	Key pair name
-	

**Step 1** [Choose launch template or configuration](#)

**Step 2** [Configure settings](#)

**Step 3 (optional)** [Configure advanced options](#)

**Step 4 (optional)** [Configure group size and scaling policies](#)

**Step 5 (optional)** [Add notifications](#)

**Step 6 (optional)** [Add tags](#)

**Step 7** [Review](#)

**Configure settings** Info

Configure the settings below. Depending on whether you chose a launch template or configuration, some steps might be different or omitted. These settings help you make optimal use of EC2 resources.

**Network** Info

For most applications, you can use multiple Availability Zones and let EC2 automatically spread instances across them. The default VPC and default subnets are suitable for getting started.

**VPC**

vpc-0bc403c54a7f58d37 (Poonamvpc1) 10.0.0.0/16 [Edit](#)

**Create a VPC** [Edit](#)

**Subnets**

Select subnets [Edit](#)

ap-south-1a   subnet-0fff4a10f08254692 (Poonam_subnet1) 10.0.1.0/24	X
ap-south-1b   subnet-04cc3960bcd6a942a (Poonam_subnet2) 10.0.2.0/24	X
ap-south-1c   subnet-0c3e1c43030737770 (Poonam_subnet3) 10.0.3.0/24	X

## Attach existing Load Balancer

Choose a load balancer to distribute incoming traffic for your application across instances to make it more reliable and easily scalable. You can also set options that give you more control over health check replacements and monitoring.

### Load balancing - optional [Info](#)

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer  
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer  
Choose from your existing load balancers.

Attach to a new load balancer  
Quickly create a basic load balancer to attach to your Auto Scaling group.

### Attach to an existing load balancer

Select the load balancers that you want to attach to your Auto Scaling group.

Choose from your load balancer target groups  
This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

#### Existing load balancer target groups

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups

Poonam-Proj-TG | HTTP  
Application Load Balancer: ProjectpoonamLB X

## Reduce health check duration to 120seconds

### Health checks - optional

#### Health check type [Info](#)

EC2 Auto Scaling automatically replaces instances that fail health checks. If you enabled load balancing, you can enable ELB health checks in addition to the EC2 health checks that are always enabled.

EC2  ELB

#### Health check grace period

The amount of time until EC2 Auto Scaling performs the first health check on new instances after they are put into service.

120 seconds

### Group size - optional [Info](#)

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

#### Desired capacity

2

#### Minimum capacity

2

#### Maximum capacity

4

### Scaling policies - optional

Choose whether to use a scaling policy to dynamically resize your Auto Scaling group to meet changes in demand. [Info](#)

Target tracking scaling policy

Choose a desired outcome and leave it to the scaling policy to add and remove capacity as needed to achieve that outcome.

None

## 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	<a href="#">subnet-0fff4a10f08254692</a>	10.0.1.0/24
ap-south-1b	<a href="#">subnet-04cc3960bdc6a942a</a>	10.0.2.0/24
ap-south-1c	<a href="#">subnet-0c3e1c43030737770</a>	10.0.3.0/24

### Step 3: Configure advanced options

Edit

#### Load balancing

Load balancer 1  
Name [ProjectpoonamLB](#) Type Application/HTTP Target group [Poonam-Proj-TG](#)

#### Health checks

Health check type EC2 Health check grace period 120 seconds

#### Additional settings

Monitoring  
Disabled

Step 4: Configure group size and scaling policies

**Group size**

Desired capacity	Minimum capacity	Maximum capacity
2	2	4

**Scaling policy**

No scaling policy
-------------------

**Instance scale-in protection**

Instance scale-in protection	<input checked="" type="checkbox"/> Enable instance protection from scale in
------------------------------	--

Step 5: Add notifications

**Notifications**

No notifications

Cancel
Create Auto Scaling group

⌚ Poonam-Proj-ASgroup created successfully X

EC2 > Auto Scaling groups

Auto Scaling groups (1)								
<input type="button" value="C"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Create an Auto Scaling group"/>								
<input type="text"/> Search your Auto Scaling groups								
<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
<input type="checkbox"/>	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

Instances (3) <a href="#">Info</a>								
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input type="checkbox"/>	Proj22	i-0d5187cacbaf30b24	<span>Running</span>	t2.micro	<span>Initializing</span>	No alarms	ap-south-1a	ec2-3-110-32-130.ap-s...
<input type="checkbox"/>	Poonam_Proj_EC2_ubuntu	i-0b3022973bdb0b0b8	<span>Running</span>	t2.micro	<span>2/2 checks passed</span>	No alarms	ap-south-1b	ec2-13-126-70-24.ap-s...
<input type="checkbox"/>	Proj22	i-0344ab1972bfd1f15	<span>Running</span>	t2.micro	<span>Initializing</span>	No alarms	ap-south-1b	ec2-3-109-157-76.ap-s...

Terminated the first instance created

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

The image contains three vertically stacked screenshots of a web browser window. Each screenshot shows a different response from a terminated EC2 instance. The browser's address bar shows the URL as 'Not secure | projectpoonamlb-1722643524.ap-south-1.elb.amazonaws.com'. The first screenshot displays the message 'Welcome!'. The second screenshot displays the message 'I am good, how about you?'. The third screenshot displays a long list of names: Murphy Diane, Firrelli Jeff, Patterson William, Bondur Gerard, Bow Anthony, Jennings Leslie, Thompson Leslie, Firrelli Julie, Patterson Steve, Tseng Foon.

This image shows a single screenshot of a web browser window. The browser's address bar shows the URL as 'Not secure | projectpoonamlb-1722643524.ap-south-1.elb.amazonaws.com/read%20from%20database'. The page content lists a large number of names, including Murphy Diane, Firrelli Jeff, Patterson William, Bondur Gerard, Bow Anthony, Jennings Leslie, Thompson Leslie, Firrelli Julie, Patterson Steve, Tseng Foon, and many others.

Murphy Diane, Firrelli Jeff, Patterson William, Bondur Gerard, Bow Anthony, Jennings Leslie, Thompson Leslie, Firrelli Julie, Patterson Steve, Tseng Foon

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:

Auto Scaling groups (1/1)								
<input type="button" value="C"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Create an Auto Scaling group"/>								
<input checked="" type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
<input checked="" type="checkbox"/>	Poonam-Proj-ASgroup	Poonam-proj-LC	2	-	2	2	4	ap-south-1c, ap-south-1...
<hr/>								
Status	Description	Cause	Start time	End time				
WaitingForELBConnectionDraining	Terminating EC2 instance: i-0344ab1972bfd1f15 - Waiting For ELB Connection Draining.	At 2021-09-26T05:08:50Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2021 September 26, 10:38:50 AM +05:30					
Successful	Launching a new EC2 instance: i-0344ab1972bfd1f15	At 2021-09-26T04:46:51Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2021 September 26, 10:16:53 AM +05:30	2021 September 26, 10:17:55 AM +05:30				
	Launching a new EC2	At 2021-09-26T04:46:26Z a user request created an AutoScalingGroup changing the desired capacity	2021 September	2021 September				

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

Instances (4) <a href="#">Info</a>										
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Pu	Actions
<input type="checkbox"/>	Proj22	i-0d5187cacbaf30b24	<span>Running</span>  	t2.micro	<span>2/2 checks passed</span>  	No alarms 	ap-south-1a	ec2-3-110-32-130.ap-s...	3.1	<a href="#">Launch instances</a>
<input type="checkbox"/>	Poonam_Proj_...	i-0b3022973bdb0b0b8	<span>Terminated</span>  	t2.micro	-	No alarms 	ap-south-1b	-	-	<a href="#">Actions</a>
<input type="checkbox"/>	Proj22	i-0344ab1972bfd1f15	<span>Terminated</span>  	t2.micro	-	No alarms 	ap-south-1b	-	-	<a href="#">Actions</a>
<input type="checkbox"/>	Proj22	i-05aec4573b683d587	<span>Running</span>  	t2.micro	<span>2/2 checks passed</span>  	No alarms 	ap-south-1b	ec2-13-234-76-31.ap-s...	13	<a href="#">Launch instances</a>

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