

# CAPSTONE PROJECT – Migrating to Cloud

## Project Objective

- ‘UrbanSloth’ is a unicorn food delivery app in India which has unexpected increase in the load on their application.
- The objective of the project is to set-up the POC application , which is part of the application hosted on the third party rented infrastructure , in AWS.
- The migration of the application must happen without any loss of data.
- The application hosted on AWS should be efficient, reliable and secure.
- Appropriate AWS services must be selected for the migration based on the requirements.
- After migration of the POC application, end-to-end testing must be done to ensure the application works as expected.

## Project Requirements

- 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 group with application load balancer for the above application configuration.

## Project Implementation

- Setting up the database server (RDS) on AWS.
  1. Creating DB Subnet group.
    - a. Navigate to the ‘RDS’ service in AWS Account. Click on the ‘subnet groups’, and ‘Create DB subnet group’.

The screenshot shows the AWS RDS Subnet groups page. The left sidebar has a 'Subnet groups' section selected. The main area displays a table titled 'Subnet groups (0)' with columns for Name, Description, Status, and VPC. A message at the bottom states 'No db subnet groups' and 'You don't have any db subnet groups.' A 'Create DB Subnet Group' button is visible.

- b. Name of the subnet group – ‘project-rds-subnetgroup’. The VPC in which the subnet group must be created should also be specified. (Choosing default VPC)

The screenshot shows the 'Create DB Subnet Group' wizard. Step 1: Subnet group details. It asks for a name ('project-rds-subnetgroup') and a description ('project-rds-subnetgroup'). A dropdown for 'VPC' is set to 'vpc-03c51c06eedb91512'. Step 2: Add subnets is partially visible below.

- c. Choose all the availability zones and subnets in the region to ensure availability of the database.

The screenshot shows the 'Add subnets' step of the wizard. Under 'Availability Zones', 'us-east-1a', 'us-east-1b', 'us-east-1c', 'us-east-1d', 'us-east-1e', and 'us-east-1f' are selected. Under 'Subnets', six subnets are listed: 'subnet-05d41363fab019764 (172.31.32.0/20)', 'subnet-02ebcea04e90fb85c6 (172.31.80.0/20)', 'subnet-05b809274d6fa8904 (172.31.0.0/20)', 'subnet-0f890c224cff9aa06 (172.31.16.0/20)', 'subnet-0fe00bb3a9381480 (172.31.64.0/20)', and 'subnet-09416f5ac4f71b47c (172.31.48.0/20)'. A summary at the bottom shows 'Subnets selected (6)'.

d. The DB Subnet group is created successfully.

The screenshot shows the AWS RDS Subnet Groups page. At the top, a green banner says "Successfully created project-rds-subnetgroup. View subnet group". Below it, the "Subnet groups (1)" section displays a table with one row. The table columns are Name, Description, Status, and VPC. The single entry is "project-rds-subnetgroup", "project-rds-subnetgroup", "Complete", and "vpc-03c51c06eedb91512". The left sidebar shows navigation options like Dashboard, Databases, and Subnet groups.

## 2. Create security group for RDS.

- Navigate to 'ec2' service in the AWS Dashboard and go to 'Security group'. A Default Security group will already be present. Create a new security group.

The screenshot shows the AWS EC2 Security Groups page. The left sidebar lists services like Capacity Reservations, Images, Elastic Block Store, Network & Security (Security Groups), Load Balancing, Auto Scaling, and more. The "Security Groups" section shows a table with one row for the default security group. The table columns are Name, Security group ID, Security group name, VPC ID, Description, and Owner. The entry is "- sg-0a28193b166150f60 default vpc-03c51c06eedb91512 default VPC security gr... 917249180955". Below the table, a details panel for "sg-0a28193b166150f60 - default" shows tabs for Details, Inbound rules, Outbound rules, and Tags. It also includes a "Run Reachability Analyzer" button.

- Appropriate name and description must be given while creating the security group('Project-rds-sg'). Add an inbound rule to group , where the Type is 'MYSQL' , port is 3306 and the source is the CIDR block of the VPC in which the RDS will be created.(Default VPC)

**Basic details**

Security group name: Project-rds-sg  
Description: Project-rds-sg  
VPC: vpc-03c51c06eedb91512

**Inbound rules**

Type	Protocol	Port range	Source	Description - optional
MySQL/Aurora	TCP	3306	Custom 172.31.0.0/16	

**Outbound rules**

c. Security group is created successfully.

**Details**

Security group name: Project-rds-sg  
Owner: 917249180955  
Security group ID: sg-025549d37e5e4f579  
Description: Project-rds-sg  
VPC ID: vpc-03c51c06eedb91512

**Inbound rules (1/1)**

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-002a7cacd2a04cd21	IPv4	MySQL/Aurora	TCP	3306

### 3. Create an RDS instance.

- Navigate to the 'RDS' service in the AWS Dashboard, and create a new database.

**Databases**

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity
No instances found							

- b. Choose Engine Type as 'MySQL' and select appropriate edition and version.

The screenshot shows the 'Engine options' section of the AWS RDS setup. Under 'Engine type', 'MySQL' is selected. Below it, 'Edition' is set to 'MySQL Community'. A note about 'Known issues/limitations' is displayed, and the 'Version' is set to 'MySQL 8.0.23'.

- c. Choose 'Free Tier' as template. Also, an appropriate name for the RDS instance.

The screenshot shows the 'Templates' section of the AWS RDS setup. 'Free tier' is selected. In the 'Settings' section, the DB instance identifier is 'Projectdatabase', the master username is 'root', and the master password is '\*\*\*\*\*'. The DB instance class is set to 'db.t2.micro'.

- d. The RDS must have a username and password. Password Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), "(double quote) and @ (at sign). The DB Instance class is – 'db.t2.micro'.

The screenshot shows the 'Settings' section of the AWS RDS setup. The master password is '\*\*\*\*\*'. The DB instance class is selected as 'db.t2.micro'.

- e. Allocate 20 GB storage for the database and disable storage autoscaling.

The screenshot shows the 'Storage' configuration section of the AWS RDS console. The 'Allocated storage' field is set to 20 GiB. The 'Enable storage autoscaling' checkbox is unchecked, with a note below stating: '(Enabling this feature will allow the storage to increase once the specified threshold is exceeded.)'

- f. Choose the default VPC for the RDS and choose the subnet group which was created previously. The RDS must not have public access.

The screenshot shows the 'Connectivity' configuration section. The 'Default VPC' dropdown is set to 'Default VPC (vpc-03c5106eedb91512)'. Under 'Public access', the 'No' option is selected, with a note: '(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.)'

- g. Choose the security group which was created previously. Choose Password authentication for database authentication.

The screenshot shows the 'VPC security group' configuration section. The 'Choose existing' option is selected, and the 'Project-rds-sg' security group is chosen. Under 'Database authentication', the 'Password authentication' option is selected, with a note: '(Authenticates using database passwords.)'

## h. The RDS Instance is created successfully. (projectdatabase)

The screenshot shows the AWS RDS Databases page. At the top, a green banner says "Successfully created database projectdatabase". Below it, the "Databases" section lists "projectdatabase" as an available instance. The instance details are: Instance: MySQL Community, Region & AZ: us-east-1f, Size: db.t2.micro, Status: Available, CPU: 4.3. The left sidebar includes options like Dashboard, Databases (selected), Query Editor, Performance Insights, Snapshots, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom Availability Zones, Events, Event subscriptions, Recommendations, and Certificate update.

- Launching and setting up EC2 Instance.

1. Launch an Ubuntu EC2 instance.

- a. Navigate to the 'EC2' service in the AWS Dashboard.  
Click on 'Launch instance'.

The screenshot shows the AWS EC2 Instances page. On the right, there is a prominent orange "Launch instances" button. The main area displays a message: "You do not have any instances in this region". The left sidebar has sections for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations, Images, AMIs, and Elastic Block Store Volumes.

- b. Select the 'ubuntu-20.04' AMI

The screenshot shows the "Step 1: Choose an Amazon Machine Image (AMI)" page. It lists several AMI options under "Quick Start (8)": "Ubuntu Server 20.04 LTS (HVM), SSD Volume Type", "Ubuntu Server 18.04 LTS (HVM), SSD Volume Type", "Deep Learning AMI (Ubuntu 18.04) Version 49.0", and "Deep Learning AMI (Ubuntu 16.04) Version 49.0". Each item has a "Select" button to its right. The "Ubuntu Server 20.04 LTS (HVM), SSD Volume Type" is currently selected. The page also includes links for "My AMIs (0)", "AWS Marketplace (877)", "Community AMIs (37881)", and "Free tier only".

c. The instance type is “t2.micro”.W

The screenshot shows the AWS EC2 console with the 'Choose Instance Type' step selected. A table lists various instance types based on family, type, vCPUs, memory, instance storage, EBS-optimized availability, network performance, and IPv6 support. The 't2.micro' row is highlighted with a blue border and has a green 'Free tier eligible' badge. The table includes columns for Family, Type, vCPUs, Memory (GiB), Instance Storage (GiB), EBS-Optimized Available, Network Performance, and IPv6 Support.

d. While configuring the instance, select the Default VPC and enable the public IP for the instance.

The screenshot shows the 'Configure Instance Details' step. Under the 'Network' section, the 'Subnet' dropdown is set to 'No preference (default subnet in any Availability Zone)'. The 'Auto-assign Public IP' dropdown is set to 'Use subnet setting (Enable)'. Other settings like 'Domain join directory', 'IAM role', 'Shutdown behavior', and 'Monitoring' are also visible.

e. The below bash script must be added in the userdata section:

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

The screenshot shows the 'Configure Instance Details' step with the 'Advanced Details' section expanded. Under 'User data', there is a text input field containing the following bash script:

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

f. Allocate 8GB storage for the instance.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0a52a0f61496c3782	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Cancel Previous Review and Launch Next: Add Tags

g. Appropriate name tag must be given for the instance (“Project-ubuntu-1”)

Key	Value	Instances	Volumes	Network Interfaces
Name	Project-ubuntu-1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

Cancel Previous Review and Launch Next: Configure Security Group

h. Create a new security group for the EC2 instance , and add inbound rules for the instance for “HTTP” type and “SSH” type and source as “Anywhere”

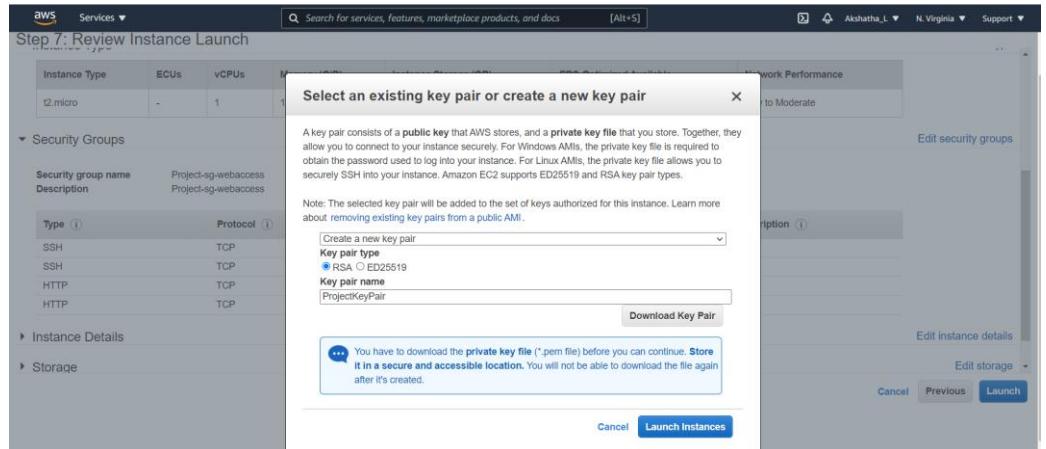
Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere	e.g. SSH for Admin Desktop

Add Rule

⚠ Warning  
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

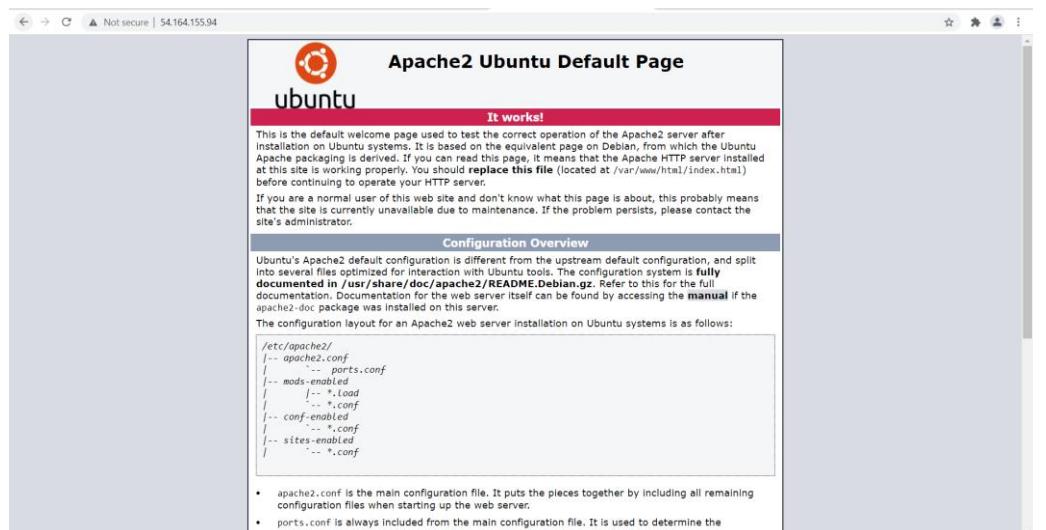
Cancel Previous Review and Launch

i. Launch the instance with existing or new keypair.



j. The instance is launched successfully.

k. In a chrome tab, go to – <http://<Public-IP>:80>, if the apache server is installed , the below page must be visible.

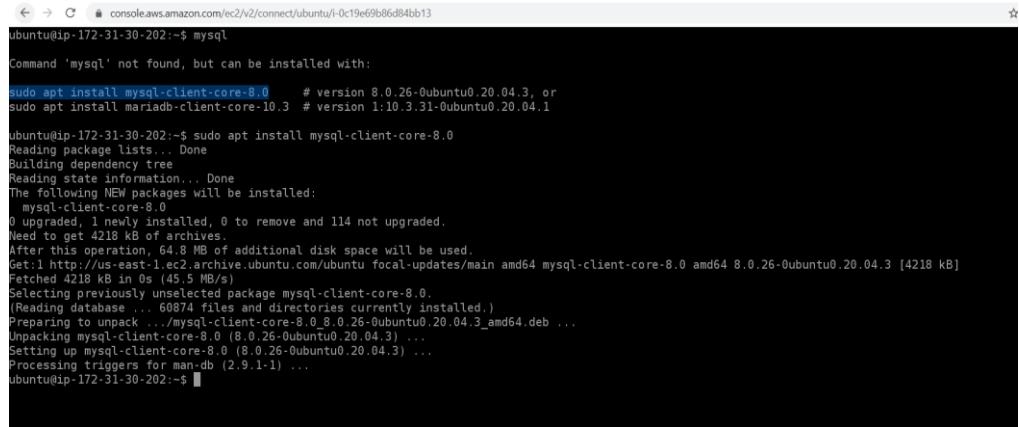


2. Install MySQL client in the EC2 instance.

- SSH into the instance with putty , and execute the following command –

```
sudo apt-get update
```

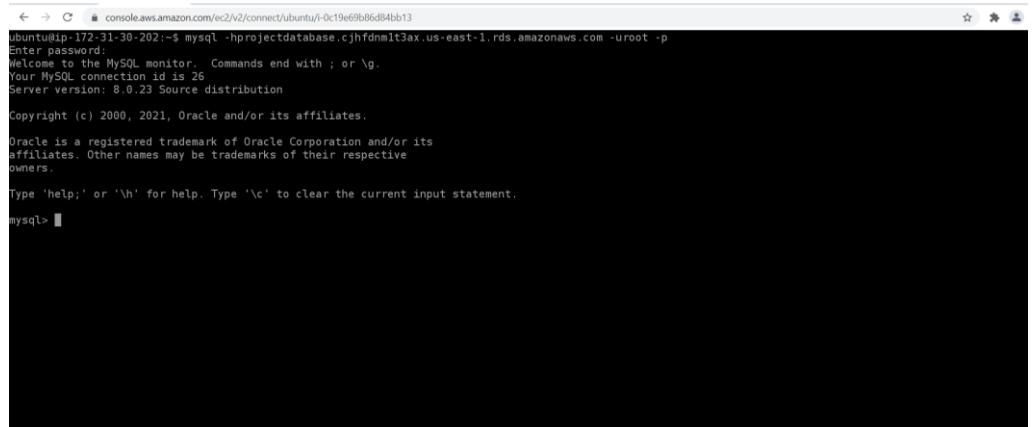
```
sudo apt-get install mysql-client-core-8.0
```



```
ubuntu@ip-172-31-30-202:~$ mysql
Command 'mysql' not found, but can be installed with:
  sudo apt install mysql-client-core-8.0 # version 8.0.26-0ubuntu0.20.04.3, or
  sudo apt install mariadb-client-core-10.3 # version 1:10.3.31-0ubuntu0.20.04.1

ubuntu@ip-172-31-30-202:~$ sudo apt install mysql-client-core-8.0
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  mysql-client-core-8.0
0 upgraded, 1 newly installed, 0 to remove and 114 not upgraded.
Need to get 4218 kB of additional disk space will be used.
After this operation, 64.8 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 mysql-client-core-8.0 amd64 8.0.26-0ubuntu0.20.04.3 [4218 kB]
Fetched 4218 kB in 0s (45.5 MB/s)
Selecting previously unselected package mysql-client-core-8.0.
(Reading database ... 60874 files and directories currently installed.)
Preparing to unpack .../mysql-client-core-8.0_8.0.26-0ubuntu0.20.04.3_amd64.deb ...
Unpacking mysql-client-core-8.0 (8.0.26-0ubuntu0.20.04.3) ...
Setting up mysql-client-core-8.0 (8.0.26-0ubuntu0.20.04.3) ...
Processing triggers for man-db (2.9.1-1) ...
ubuntu@ip-172-31-30-202:~$
```

- b. Connect to the RDS Database with the command –  
*'mysql -h<Endpoint of RDS> -u<username> -p'*



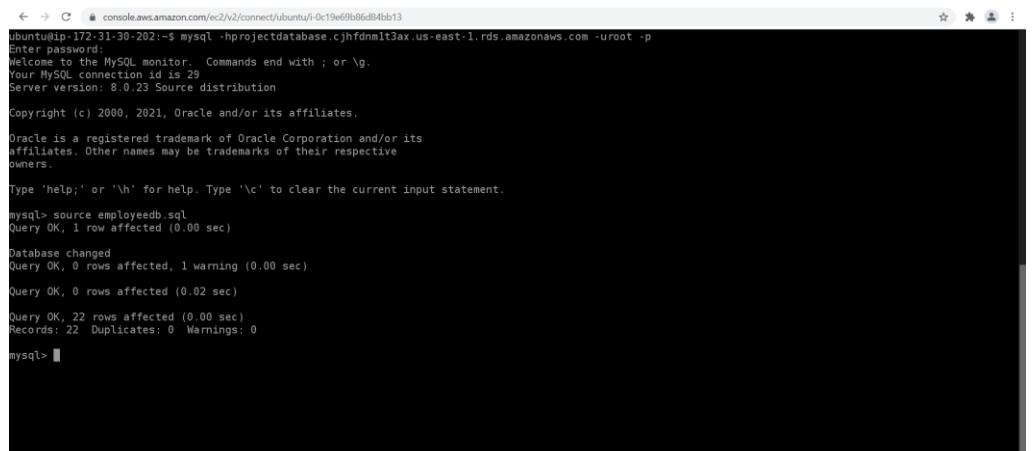
```
ubuntu@ip-172-31-30-202:~$ mysql -hprojectdatabase.cjhfndmlt3ax.us-east-1.rds.amazonaws.com -uroot -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 26
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

- c. After copying the 'employeedb.sql' in the EC2, execute –  
*'source employeedb.sql'*



```
ubuntu@ip-172-31-30-202:~$ mysql -hprojectdatabase.cjhfndmlt3ax.us-east-1.rds.amazonaws.com -uroot -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> source employeedb.sql
Query OK, 1 row affected (0.00 sec)

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

Query OK, 0 rows affected (0.02 sec)

Query OK, 22 rows affected (0.00 sec)
Records: 22  Duplicates: 0  Warnings: 0

mysql>
```

- d. To ensure that the data has been restored properly , check if the database and tables are created .  
*show databases;*  
*use employee\_db;*  
*select \*from employees;*

```

mysql> show databases;
+-----+
| Database      |
+-----+
| employee_db   |
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.01 sec)

mysql> use employee_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

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

mysql> select * from employees;
+-----+
| name        |
+-----+
| Murphy Diane |
| Firelli Jeff  |
| Patterson William |
| Bondur Gerard |
| Bow Anthony    |
| Jennings Leslie |
| Thompson Leslie |
| Firelli Julie   |
| Patterson Steve |
| Tseng Foon Yue  |
| Yamada George  |
| Bondur Louis   |
| 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>

```

i-0c19e69b86d84bb13 (Project-ubuntu-1)  
Public IPs: 54.164.155.94 Private IPs: 172.31.30.202

### 3. Configuring and testing Python-Flask application.

#### a. Install python, pip3 and flask

*python3 –version*

*udo apt-get install python3-pip*

*pip3 install flask*

*pip3 install flask-mysql*

```

ubuntu@ip-172-31-30-202:~$ python3 --version
Python 3.8.5
ubuntu@ip-172-31-30-202:~$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils-binutils-common binutils-x86_64-linux-gnu build-essential cpp cpp-9 dpkg-dev fakeroot g++-9 gcc gcc-10-base gcc-9 gcc-9-base
  libalgorithm-diff-perl libalgorithm-merge-xs-perl libatomic5 libbinutils libc-dev-bin libgcc-1.0 libcrypt-dev
  libgcc1 libgccf0 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntrlock-perl libgcc-9-dev libgcc-s1 libgomp1 libis122 libasan0 libmpc3
  libpython3.8-dev libpython3.8 libpython3.8-minimal libpython3.8-stdlib libquadmath0 libstdc++-9-dev libstdc++6 libtsan0 libubsan1
  linux-libc-dev make manpages-dev python-pip-wml python3-dev python3-wheel python3.8 python3.8-dev python3.8-minimal zlib1g-dev
Suggested packages:
  binutils-doc cpp-doc gcc-9-locales debian-keyring g++-multilib gcc-9-doc gcc-multilib autoconf automake libtool flex bison gdb gcc-doc
  gcc-9-multilib glibc-doc bzip2 libstdc++-9-doc make-doc python3.8-venv python3.8-doc binfmt-support
The following NEW packages will be installed:
  binutils-binutils-common binutils-x86_64-linux-gnu build-essential cpp cpp-9 dpkg-dev fakeroot g++-9 gcc gcc-9-base libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1 libbinutils libc-dev-bin libgcc1 libcrypt-dev libgccf-nobfd0 libgccf0
  libdpkg-perl libexpat1-dev libfakeroot libfile-fcntrlock-perl libgcc-9-dev libgomp1 libis122 libasan0 libmpc3 libpython3-dev libpython3.8-dev
  libquadmath0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev make manpages-dev python-pip-wml python3-dev python3-pip python3.8-dev
  zlib1g-dev
The following packages will be upgraded:
  gcc-10-base libgcc-s1 libpython3.8 libpython3.8-minimal libpython3.8-stdlib libstdc++6 python3.8 python3.8-minimal
8 upgraded, 50 newly installed, 0 to remove and 106 not upgraded.
Need to get 56.6 MB of archives.
After this operation, 214 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get: http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3.8 amd64 3.8.10-0ubuntu1-20.04 [387 kB]

```

i-0c19e69b86d84bb13 (Project-ubuntu-1)

Public IPs: 54.164.155.94 Private IPs: 172.31.30.202

```
ubuntu@ip-172-31-30-202:~$ pip3 install flask-mysql
Collecting Flask-MYSQL
  Downloading Flask_MYSQL-1.5.2-py2.py3-none-any.whl (3.8 kB)
Requirement already satisfied: Flask in /usr/lib/python3/dist-packages (from flask-mysql) (1.1.1)
Collecting PyMySQL
  Downloading PyMySQL-1.0.2-py3-none-any.whl (43 kB)
    100% |██████████| 43 kB 3.5 MB/s
Installing collected packages: PyMySQL, flask-mysql
Successfully installed PyMySQL-1.0.2 flask-mysql-1.5.2
ubuntu@ip-172-31-30-202:~$
```

i-0c19e69b86d84bb13 (Project-ubuntu-1)

Public IPs: 54.164.155.94 Private IPs: 172.31.30.202

- b. Copy the app.py code into the EC2 instance , and change the following in the code-

```
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 'localhost'
# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = 'Root123$'
app.config['MYSQL_DATABASE_DB'] = 'employee_db'
app.config['MYSQL_DATABASE_HOST'] = 'projectdatabase.cjhfdnmlt3ax.us-east-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])
    return str(result)
"app.py" 36L, 1047C
```

i-0c19e69b86d84bb13 (Project-ubuntu-1)

Public IPs: 54.164.155.94 Private IPs: 172.31.30.202

- c. Run the python flask application , with the command –

```
FLASK_APP=app.py flask run --host=0.0.0.0
```

```
ubuntu@ip-172-31-30-202:~$ FLASK_APP=app.py flask run --host=0.0.0.0
* Serving Flask app "app.py"
* 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 http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

i-0c19e69b86d84bb13 (Project-ubuntu-1)

Public IPs: 54.164.155.94 Private IPs: 172.31.30.202

- d. As the python application is running in port 5000, add the inbound rule in the security group of the instance.

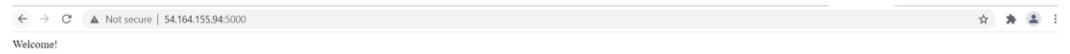
Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-019a51df50c1cc662	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-02ab9b5db67097f5f	SSH	TCP	22	Custom	0.0.0.0/0
sgr-05cbc957ef57908b6	SSH	TCP	22	Custom	:/0
sgr-0bc13026d3942a761	HTTP	TCP	80	Custom	:/0
-	Custom TCP	TCP	5000	Anywhere	0.0.0.0/0

[Add rule](#)

Cancel [Preview changes](#) **Save rules**

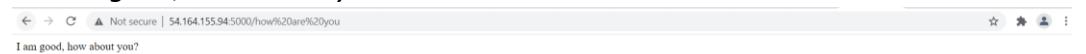
- e. Open a browser and enter the URL

*http://<IP address of your EC2 Instance>:5000 -> Welcome*



*http://<IP address of your EC2 Instance>:5000/how%20are%20you*

*-> I am good, how about you?*

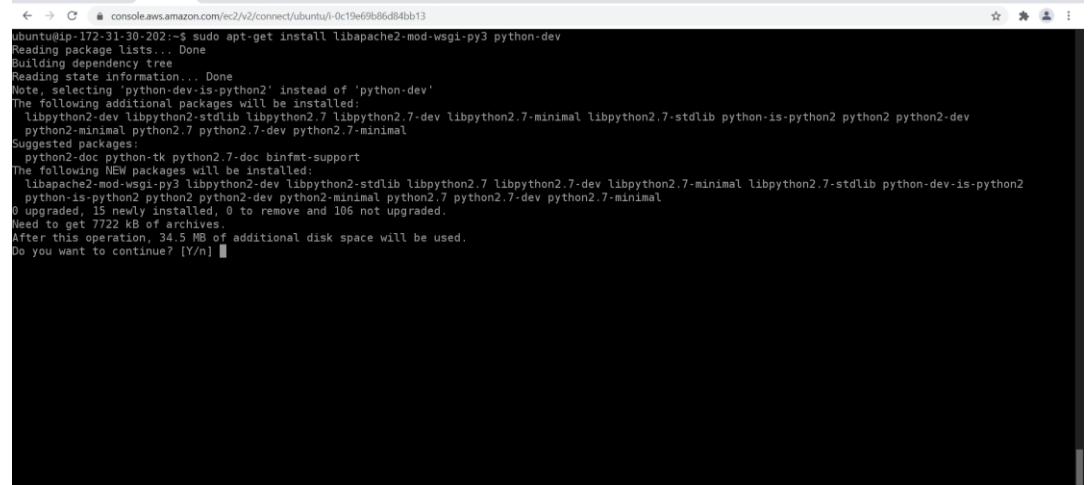


*http://<IP address of your EC2 Instance>:5000/read%20from%20database-><List of employee names>*



A screenshot of a web browser window. The URL bar shows the address: "54.164.155.94:5000/read%20from%20database". The page content displays a list of employee names: Murphy,Diane,Firelli,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,Marsch,Peter,King,Tom,Nishi,Mami,Kato,Yoshimi,Gerard,Martin.

- Change Port of Flask application from 5000 to 80.
  1. Install and enable mod\_wsgi(Web server Gateway Interface module)  
**\$ sudo apt-get install libapache2-mod-wsgi-py3 python-dev**  
**\$ sudo a2enmod wsgi**



```
ubuntu@ip-172-31-202-4:~$ sudo apt-get install libapache2-mod-wsgi-py3 python-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python-dev-is-python2' instead of 'python-dev'
The following additional packages will be installed:
  libpython2-dev libpython2-stdlib libpython2.7 libpython2.7-dev libpython2.7-minimal libpython2.7-stdlib python-is-python2 python2 python2-dev
  python2-minimal python2.7 python2.7-dev python2.7-minimal
Suggested packages:
  python2-doc python-tk python2.7-doc binfmt-support
The following NEW packages will be installed:
  libapache2-mod-wsgi-py3 libpython2-dev libpython2-stdlib libpython2.7 libpython2.7-dev libpython2.7-minimal libpython2.7-stdlib python-dev-is-python2
  python-is-python2 python2 python2-dev python2-minimal python2.7 python2.7-dev python2.7-minimal
0 upgraded, 15 newly installed, 0 to remove and 106 not upgraded.
Need to get 7722 kB of archives.
After this operation, 34.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] ■
```

i-0c19e69b86d84bb13 (Project-ubuntu-1)

Public IPs: 54.164.155.94 Private IPs: 172.31.30.202

```
ubuntu@ip-172-31-30-202:~$ sudo a2enmod wsgi
Module wsgi already enabled
ubuntu@ip-172-31-30-202:~$
```

i-0c19e69b86d84bb13 (Project-ubuntu-1)  
Public IPs: 54.164.155.94      Private IPs: 172.31.30.202

List the contents of the default web pages directory.

```
$ ls -l /var/www
```

```
ubuntu@ip-172-31-30-202:~$ ls -l /var/www
total 4
drwxr-xr-x 2 root root 4096 Oct  3 10:42 html
ubuntu@ip-172-31-30-202:~$
```

i-0c19e69b86d84bb13 (Project-ubuntu-1)  
Public IPs: 54.164.155.94      Private IPs: 172.31.30.202

2. Create the required sub-directories in the above and copy the existing flask application into the subdirectory.

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

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

```
Unpacking libpython2.7-amd64 (2.7.18-1~20.04.1) ...
Selecting previously unselected package libpython2.7-dev:amd64.
Preparing to unpack .../3-libpython2.7-dev_2.7.18-1~20.04.1_amd64.deb ...
total 4
drwxr-xr-x 2 root root 4096 Oct  3 10:42 html
ubuntu@ip-172-31-30-202:~$ sudo mkdir -p /var/www/FlaskApp/FlaskApp
ubuntu@ip-172-31-30-202:~$ sudo cp app.py /var/www/FlaskApp/FlaskApp/__init__.py
ubuntu@ip-172-31-30-202:~$ cat /var/www/FlaskApp/FlaskApp/__init__.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'
if 'MYSQL_CONFIG' in os.environ:
    app.config.from_object(os.environ['MYSQL_CONFIG'])
else:
    app.config['MYSQL_DATABASE_USER'] = 'root'
    app.config['MYSQL_DATABASE_PASSWORD'] = 'Root123$'
    app.config['MYSQL_DATABASE_DB'] = 'employee_db'
    app.config['MYSQL_DATABASE_HOST'] = 'projectdatabase.cjhfndm1t3ax.us-east-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():

i-0c19e69b86d84bb13 (Project-ubuntu-1)
```

Public IPs: 54.164.155.94      Private IPs: 172.31.30.202

3. Configure and Enable a New Virtual Host at port number 80.

- a) Create the configuration file for the FlaskApp.

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

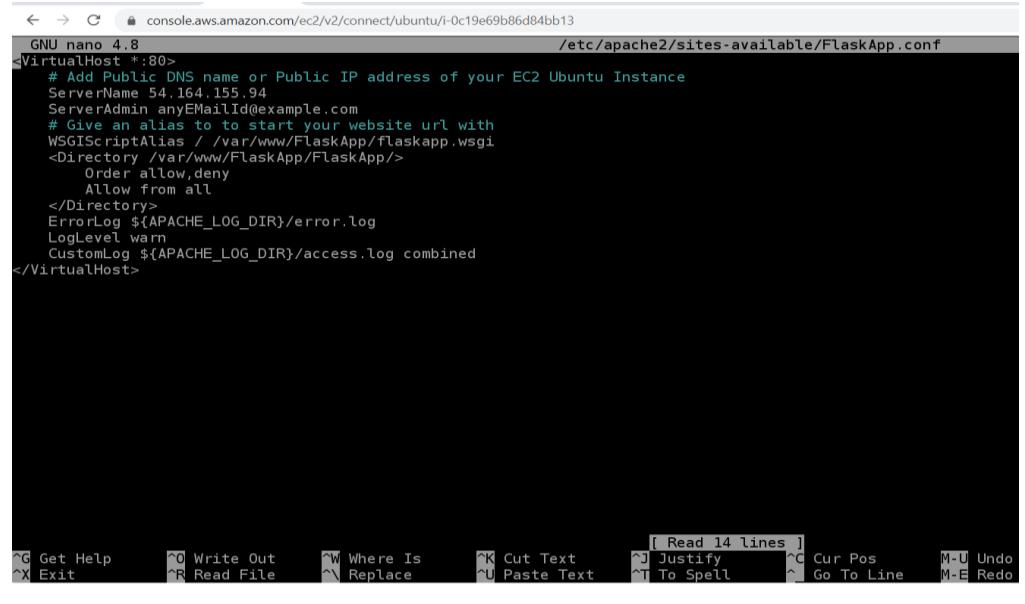
And copy paste the following lines:

Copy paste the follwing lines.

-->

```
<VirtualHost *:80>
    # Add Public DNS name or Public IP address of your EC2 Ubuntu
    Instance
    ServerName ec2-54-164-155-94.compute-1.amazonaws.com
    ServerAdmin anyEmailId@example.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>
```

<--

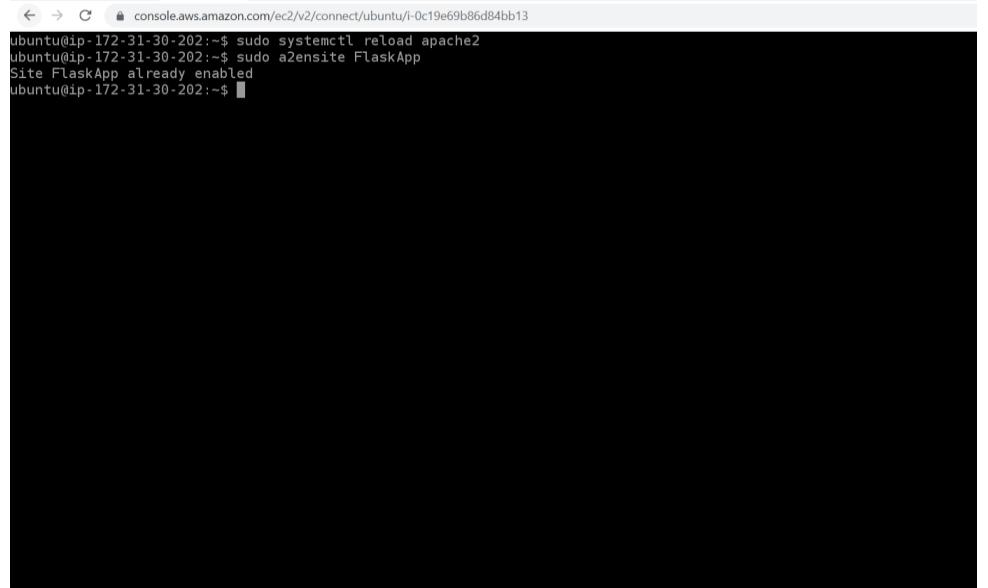


```
GNU nano 4.8                               /etc/apache2/sites-available/FlaskApp.conf
<VirtualHost *:80>
    # Add Public DNS name or Public IP address of your EC2 Ubuntu Instance
    ServerName 54.164.155.94
    ServerAdmin anyEmailId@example.com
    # Give an alias to to start your website url 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>
```

- b) Enable the virtual host with the command below.

```
$ sudo a2ensite FlaskApp
```

```
$sudo systemctl reload apache2
```



```
ubuntu@ip-172-31-30-202:~$ sudo systemctl reload apache2
ubuntu@ip-172-31-30-202:~$ sudo a2ensite FlaskApp
Site FlaskApp already enabled
ubuntu@ip-172-31-30-202:~$
```

i-0c19e69b86d84bb13 (Project-ubuntu-1)

#### 4. 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 = 'Add your secret key or a Dummy
string'
```

<--

```
#!/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 = 'Add your secret key or a Dummy string'
```

... INSERT ...

8,65

i-0c19e69b86d84bb13 (Project-ubuntu-1)
Public IPs: 54.164.155.94 Private IPs: 172.31.30.202

5. Finally restart apache2 server(*\$ sudo service apache2 restart*)



```
← → ⌂ console.aws.amazon.com/ec2/v2/connect/ubuntu/i-0c19e69b86d84bb13
ubuntu@ip-172-31-30-202:~$ sudo service apache2 restart
ubuntu@ip-172-31-30-202:~$
```

6. Now open a browser on your laptop and in the address bar give the Public IP address or Public DNS name of your EC2 instance.

*http://<IP address of your EC2 Instance>/ => “Welcome”*



*http://<IP address of your EC2 Instance>/how%20are%20you => I am good, how about you?*



`http://<IP address of your EC2 instance/read%20from%20database => <list of employee names>`



This indicates that you have installed Flask and configured a sample test Flask application to use port number 80 properly.

- Set up auto-scaling group with application load balancer for the above application configuration.
  1. Create Security group for load balancer.
    - a. Navigate to the “EC2 service” in AWS Dashboard and go to “Security groups” option to create a security group.

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

**Basic details**

Security group name [Info](#)  
Loadbalancer-SG  
Name cannot be edited after creation.

Description [Info](#)  
Loadbalancer-SG

VPC [Info](#)  
vpc-03c51c06eedb91512

**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 <a href="#">Info</a>	Delete
HTTP	TCP	80	Anywh... <a href="#">Search</a> 0.0.0.0/0 <a href="#">X</a>		<a href="#">Delete</a>
HTTP	TCP	80	Anywh... <a href="#">Search</a> 0.0.0.0/0 <a href="#">X</a>		<a href="#">Delete</a>

2. Create target group for load balancer.

- Move to “Target groups” and click on “Create target group”.  
Select target type as Instance.

The screenshot shows the 'Specify group details' step of the target group creation wizard. It includes sections for 'Basic configuration' (where 'Instances' is selected), 'Choose a target type' (with 'Instances' highlighted), and other options like 'IP addresses', 'Lambda function', and 'Application Load Balancer'. The 'Basic configuration' section notes that settings cannot be changed after creation.

- Give an appropriate name to the target group.

The screenshot shows the 'Register targets' step of the target group creation wizard. It includes fields for 'Target group name' (set to 'Project-targetgroup'), 'Protocol' (set to 'HTTP' with port '80'), 'VPC' (listing 'vpc-05c51c06ed91512' and 'IPv4: 172.31.0.0/16'), and 'Protocol version' (with 'HTTP1' selected). The 'HTTP1' option is described as supporting requests using HTTP/1.1 or HTTP/2.

- Do not register any instance with the target group now.

The screenshot shows the 'Available instances (1)' table with one entry: 'i-0c19e69b8b6d84bb13' (Name: 'Project-ubuntu-1', State: 'running', Security groups: 'Project-sg-webaccess', Zone: 'us-east-1b', Subnet ID: 'subnet-0f890c224cff9aa06'). Below the table, the 'Ports for the selected instances' field contains '80' and '1-65535 (separate multiple ports with commas)'. A button 'Include as pending below' is visible.

- Target group is created successfully.

The screenshot shows the AWS EC2 Target groups page. On the left, there's a sidebar with navigation links for EC2 Dashboard, Global View, Events, Tags, Limits, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations), Images (AMIs), and Elastic Block Store (Volumes). The main content area is titled 'Target groups (1) Info'. It shows a table with one row for 'Project-targetgroup'. The columns are Name, ARN, Port, Protocol, Target type, and Load balancer. The 'Name' column shows 'Project-targetgroup', 'ARN' shows 'arn:aws:elasticloadbalancing...', 'Port' shows '80', 'Protocol' shows 'HTTP', 'Target type' shows 'Instance', and 'Load balancer' is empty. Below the table, a message says 'Select a target group above.' with three small icons.

### 3. Create application load balancer.

- Go to “Load balancer” page in AWS Dashboard and click on “Create Load balancer”.

The screenshot shows the AWS EC2 Load Balancers page. The left sidebar is identical to the previous screenshot. The main content area has a header 'Create Load Balancer Actions'. Below it is a search bar and a table with columns for Name, DNS name, State, VPC ID, Availability Zones, Type, and Created. A message at the top right says 'You do not have any load balancers in this region.' Below the table, a section titled 'Select a load balancer' contains a message 'You do not have any load balancers in this region.' and three small icons.

- Select Load balancer type as “Application Load balancer”.

The screenshot shows the 'Select load balancer type' page. At the top, it says 'Select load balancer type' and provides a note: 'Elastic Load Balancing supports four types of load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. Choose the load balancer type that meets your needs. Learn more about which load balancer is right for you.' Below are four cards:

- Application Load Balancer**: Shows a blue circle with 'HTTP HTTPS' and a 'Create' button. Below it: '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 architectures, including microservices and containers.' A 'Learn more >' link is at the bottom.
- Network Load Balancer**: Shows a blue circle with 'TCP TLS UDP' and a 'Create' button. Below it: 'Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.' A 'Learn more >' link is at the bottom.
- Gateway Load Balancer**: Shows a blue circle with 'IP' and a 'Create' button. Below it: 'Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.' A 'Learn more >' link is at the bottom.
- Classic Load Balancer**: This card is partially visible at the bottom.

At the bottom right, there's a 'Cancel' link.

- c. Let the load balancer be internet facing , and give a name to the load balancer.

**Basic configuration**

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

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  
Recommended for internal load balancers.  
 Dualstack  
Includes IPv4 and IPv6 addresses.

**Network mapping** [Info](#)  
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

- d. Place the Load balancer in default VPC and select all available subnets in the VPC.

**Network mapping** [Info](#)  
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC [Info](#)  
Select the virtual private cloud (VPC) for your targets. Only VPCs with an internet gateway are enabled for selection. The selected VPC cannot be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#)

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. Availability Zones that are not supported by 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. Availability Zones that are not supported by the load balancer or the VPC are disabled. At least two subnets must be specified.

<input checked="" type="checkbox"/> us-east-1a	Subnet <input type="text" value="subnet-02dbcea04e80f85c"/>
<input checked="" type="checkbox"/> us-east-1b	Subnet <input type="text" value="subnet-0f890c224cff9aa06"/>

- e. Select the security group and target group that were previously created and attach it to the load balancer.

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

Security groups  
  
[Create new security group](#)

Loadbalancer-SG sg-0fa0f37017ba386c8

**Listeners and routing** [Info](#)  
A listener is a process that checks for connection requests, using the protocol and port you configure. Traffic received by the listener is then routed per your specification. You can specify multiple rules and multiple certificates per listener after the load balancer is created.

▼ Listener HTTP:80

Protocol	Port	Default action
HTTP	: 80 1-65535	<a href="#">Info</a> Forward to <b>Project-targetgroup</b> Target type: Instance <a href="#">Create target group</a>

f. The load balancer is created successfully.

The screenshot shows the AWS CloudFormation console with the search bar set to 'Project-LoadBalancer'. A table lists one resource: 'Project-LoadBalancer' (Type: application). Below the table, the 'Basic Configuration' section shows the following details:

Name	Project-LoadBalancer
ARN	arn:aws:elasticloadbalancing:us-east-1:917249180955:loadbalancer/app/Project-LoadBalancer/26cd2f91e1dff794
DNS name	Project-LoadBalancer-2082773641.us-east-1.elb.amazonaws.com (A Record)
State	Provisioning
Type	application
Scheme	internet-facing
IP address type	Ipv4

4. Create an AMI from the existing EC2 instance.

- a. SSH into the EC2 instance , and edit the “FlaskApp.conf” .

Replace the Ec2 DNS with Load balancer DNS.

```
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 Project-LoadBalancer-2082773641.us-east-1.elb.amazonaws.com
    ServerAdmin anyMailId@example.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>
```

- b. Enable the virtual host with the command below-

*\$ sudo a2ensite FlaskApp*

*\$ sudo service apache2 restart.*

```
i-0c19e69b86d84bb13 (Project-ubuntu-1)
Public IPs: 54.164.166.94   Private IPs: 172.31.30.202
Ubuntu@ip-172-31-30-202:~$ sudo a2ensite FlaskApp
Site FlaskApp already enabled
ubuntu@ip-172-31-30-202:~$
```

- c. Click on the instance in AWS dashboard and , Create an AMI from it.

The screenshot shows the AWS EC2 Instances page. A single instance, 'Project-ubuntu-1' (Instance ID: i-0c19e69b86d84bb13), is listed as 'Running'. A context menu is open over this instance, with the 'Create image' option highlighted. Other options in the menu include 'Create template from instance' and 'Launch more like this'. The top navigation bar shows 'Instances (1/1)' and 'Info'.

- d. Give a name to the image .

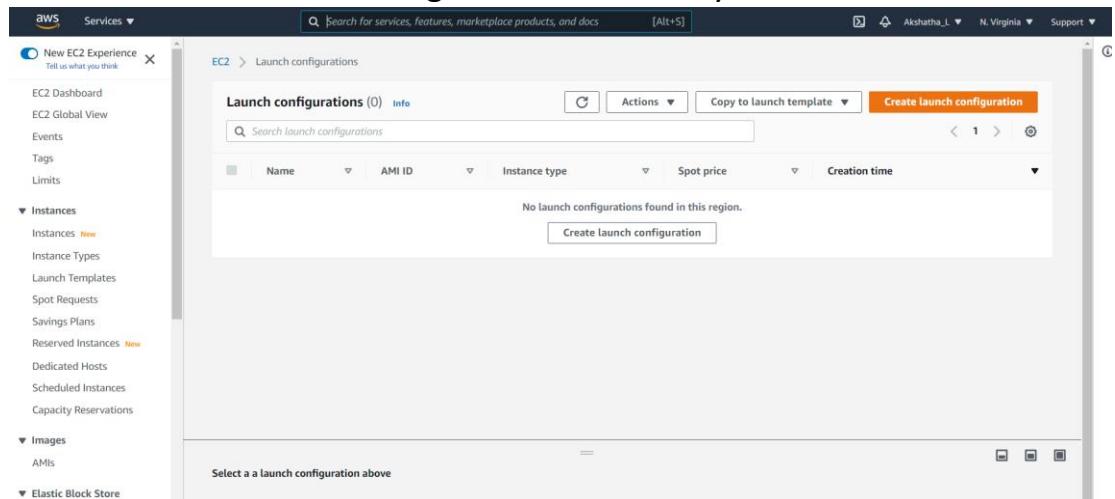
The screenshot shows the 'Create image' wizard. Step 1: 'Create image'. It shows the instance ID 'i-0c19e69b86d84bb13 (Project-ubuntu-1)' selected. The 'Image name' field is filled with 'Project-e2c-image'. The 'Image description - optional' field contains 'Project-e2c-image'. Under 'Instance volumes', there is one EBS volume selected with a size of 8 GiB, IOPS of 100, and throughput of 100. The 'Delete on termination' checkbox is checked.

- e. The image is created successfully.

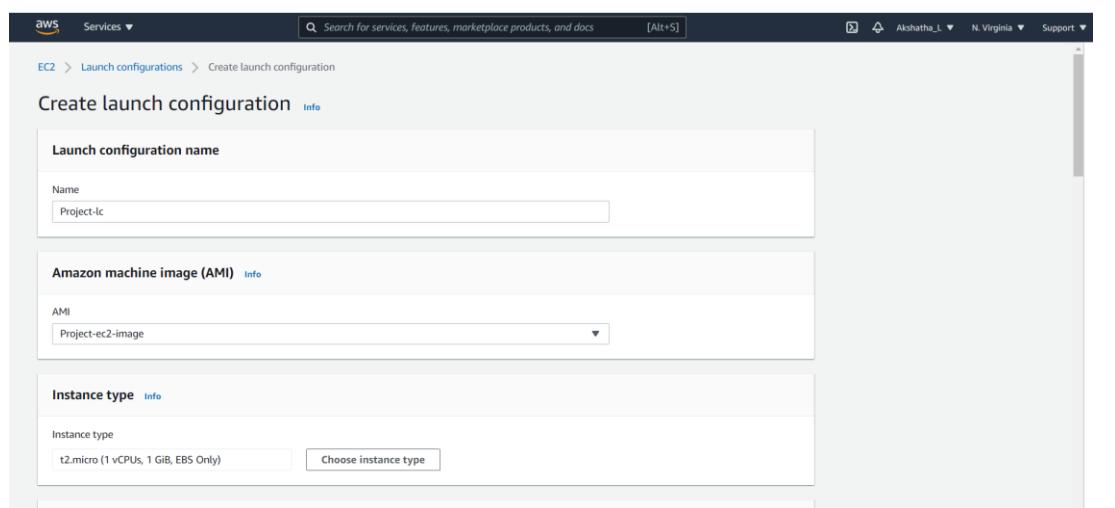
The screenshot shows the AWS EC2 Instances page. A success message at the top says 'Successfully created ami-Debe4917b7cb76228 from instance i-0c19e69b86d84bb13.' Below it, the instance list shows 'Project-ubuntu-1' (Instance ID: i-0c19e69b86d84bb13) with its new AMI information. The top navigation bar shows 'Instances (1)' and 'Info'.

## 5. Create Launch configuration from the AMI.

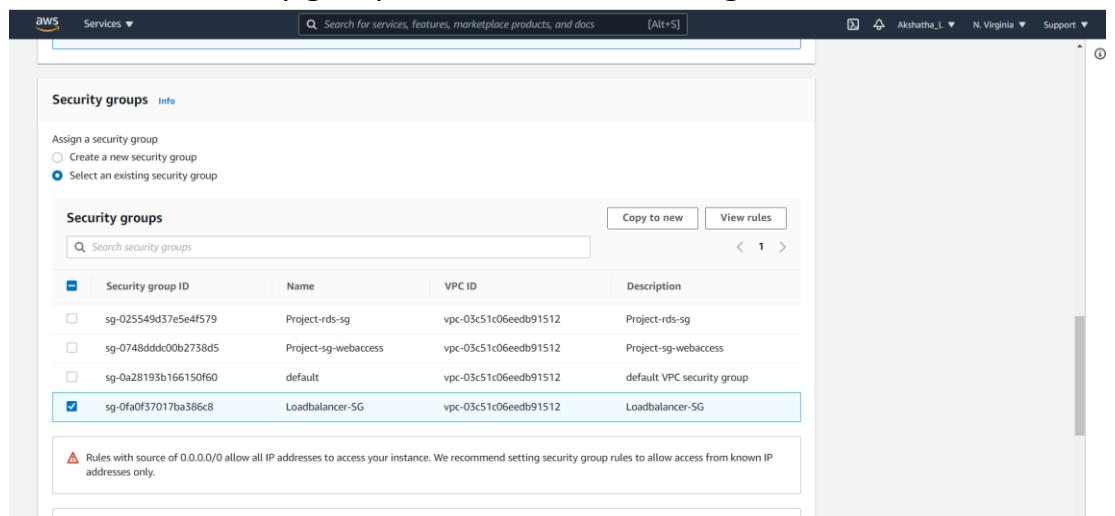
- On the navigation pane, under AUTO SCALING, choose Launch Configurations. Choose Create launch configuration, and enter a name for the launch configuration such as my-lc-01.



- Select the AMI which was created from the Ec2 instance.



- Attach the security group with the launch configuration.



- d. The Launch configuration is created successfully.

The screenshot shows the AWS EC2 Launch Configurations page. At the top, a green banner says "Successfully created launch configuration: Project-lc". Below it, the "Launch configurations" section displays a table with one row. The columns are Name, AMI ID, Instance type, Spot price, and Creation time. The data row is: Name - Project-lc, AMI ID - ami-0ebe4917b7..., Instance type - t2.micro, and Creation time - Sun Oct 03 2021 19:00:20 GMT+0530 (India Standard Time). The left sidebar shows navigation options like Instances, Images, and Elastic Block Store.

## 6. Create Autoscaling group.

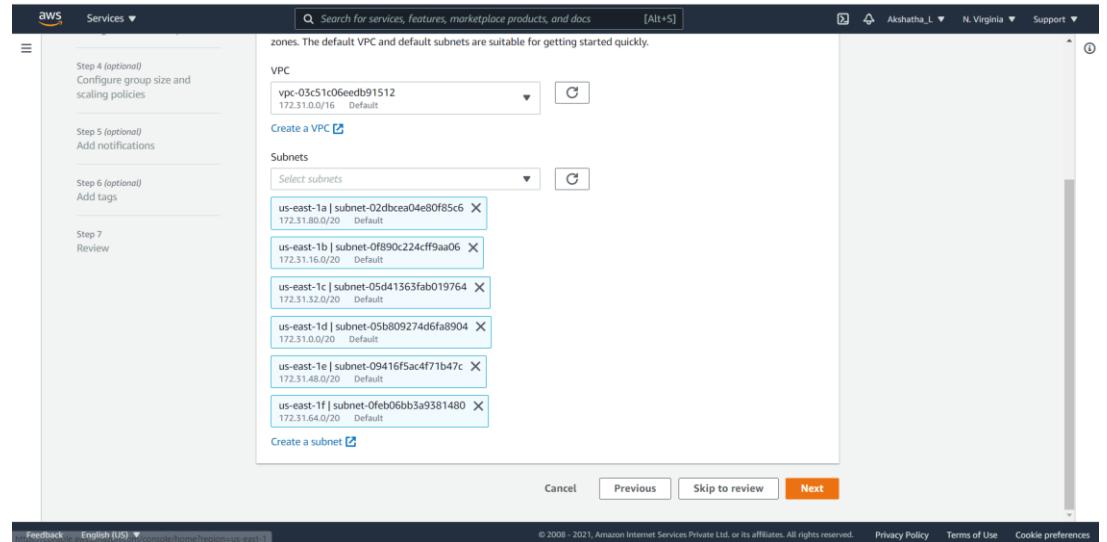
- a. On EC2 services page - Choose Create an Auto Scaling group on left navigation panel.

The screenshot shows the "Create Auto Scaling group" wizard. Step 1: How it works. It features a diagram titled "Auto Scaling group" showing four squares: two solid and two dashed, with arrows indicating scaling. Labels "Minimum size" and "Scale out as needed" are shown below. To the right, there's a "Pricing" section stating that no additional fees are required for Auto Scaling features. A "Getting started" link is also present.

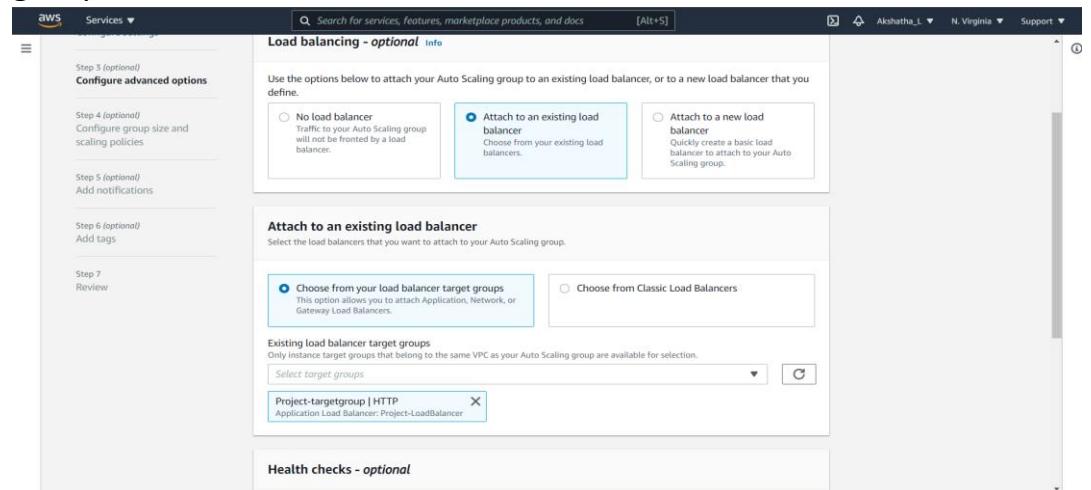
- b. Give a name to the auto scaling group and select the launch configuration that was created in the previous group.

The screenshot shows the "Create Auto Scaling group" wizard. Step 2: Configure settings. The "Name" field is filled with "Project-asg". The "Launch configuration" section shows "Project-lc" selected. Other fields include "AMI ID" (ami-0ebe4917b7cb76228), "Instance type" (t2.micro), and "Security groups" (sg-0fa0f57017ba386c8). Buttons at the bottom are "Cancel" and "Next".

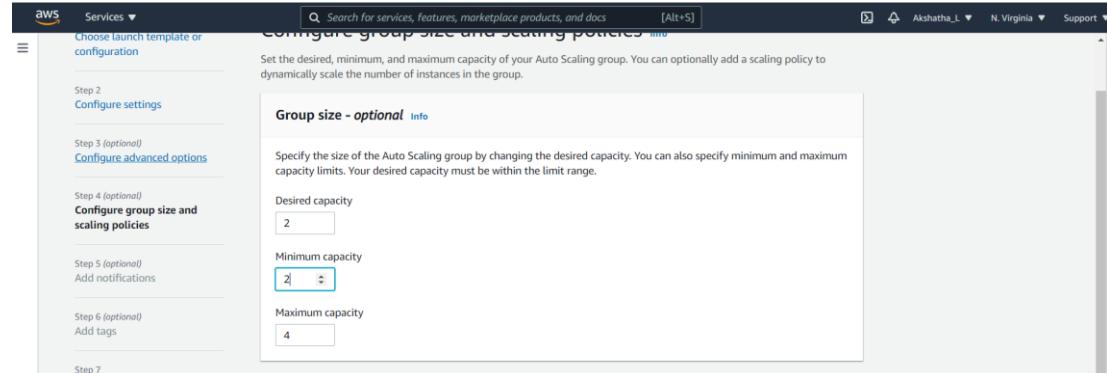
- c. Place the auto scaling group in the default VPC and select all the subnets available in the VPC.



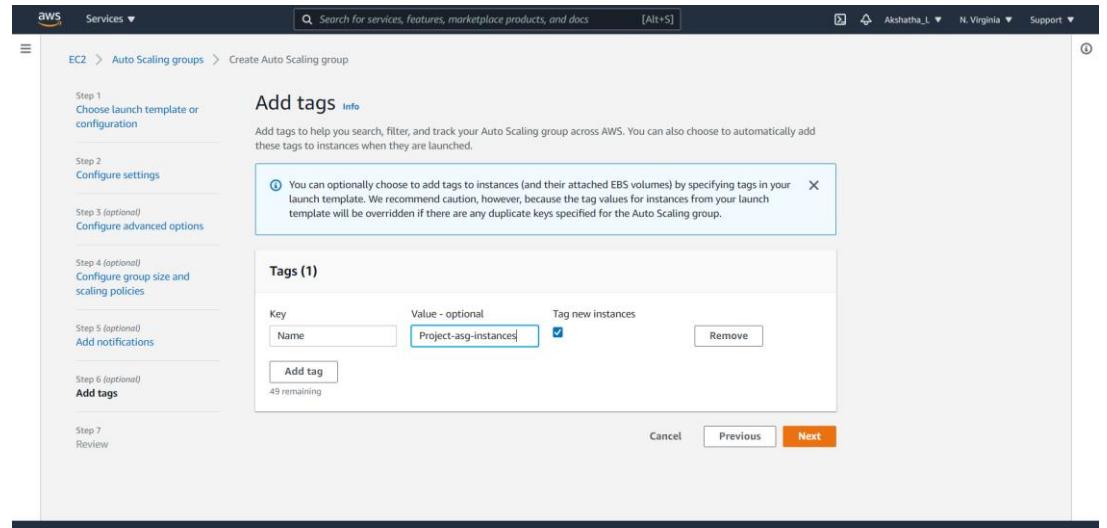
- d. Attach the auto scaling group with the load balancer and target group that was created.



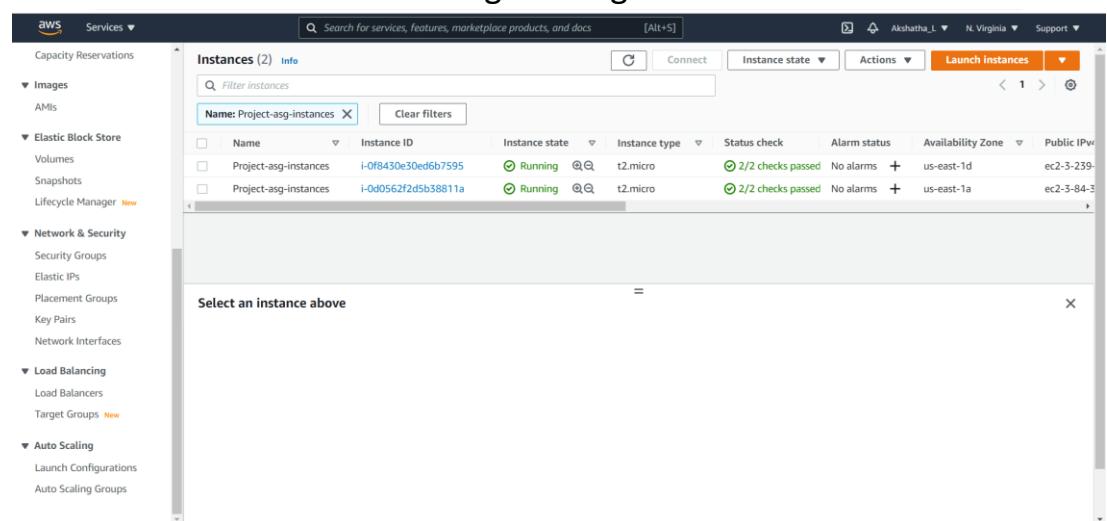
- e. Choose the group size to be – Min -2 , max -2, desired -2



f. Add tags to the auto scaling group and create the group.



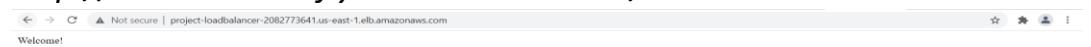
g. After the auto scaling group is created successfully, we can see two instances with the same tag running.



7. Testing.

a. Open a browser and enter the URL –

*http://<DNS name of your loadbalancer>/ => Welcome.*



*http://<DNS name of your loadbalancer>/how%20are%20you  
=> I am good, how about you?*



---

*http://<DNS name of your  
loadbalancer>/read%20from%20database => <List of employee  
names>*

