PROJECT REPORT

Group number: 9

Prudhvi Nath Naidu Cherukuri: 981212T292

Sagar Premchand Yatam: 000306T058

Sree Kavya Ganja: 000217T122

Yuktha Mullagura: 000104T060

Akhilandeswari Komara: 990109T166

PROJECT DESCRIPTION:

The main aim of this project is to deploy a simple calculator scalable web application using AWS Cloud infrastructure and implement the auto scaling policy group for the instance. The auto scaling shall be performed on the instance if there is high CPU utilization than average threshold utilization with increase in data usage and more of users.

The features that are considered are scalability with respect to computation and security of data at rest.

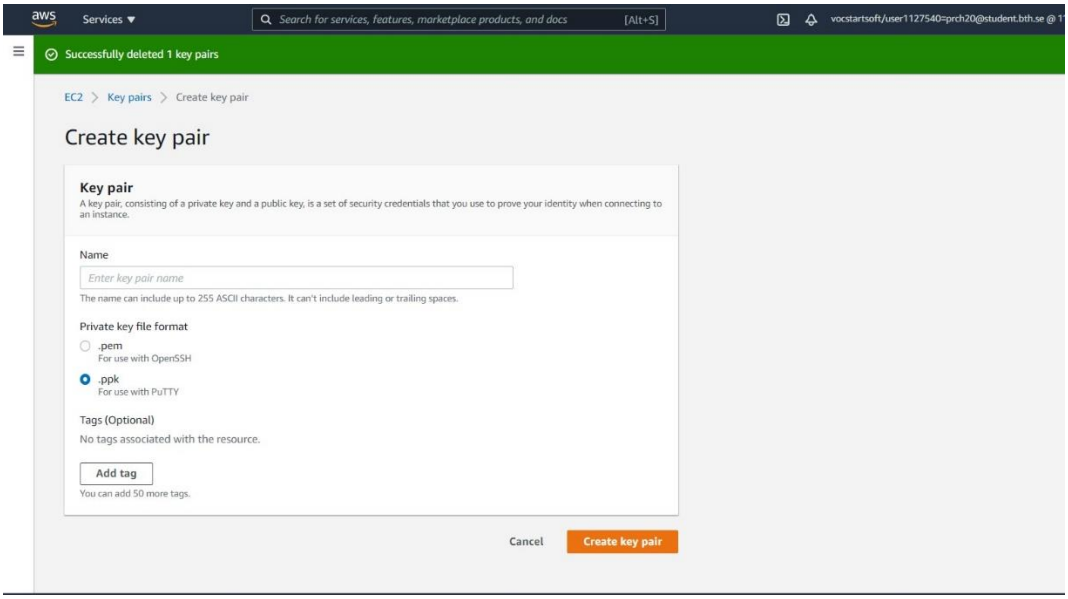
- The large amount of input data given while using the application can be handled and fulfill the feature scalability with respect to CPU utilization.
- If user does not perform any operations i.e., at the rest time the data present previously is secured while using the application. Hence, the feature security of data at rest is achieved.

AWS cloud technology is chosen to accomplish the given task. The main intention is to use web application framework i.e., Flask with python programming in AWS cloud infrastructure and test/validate our application with large amount of data from many users which in turn increases CPU utilization.

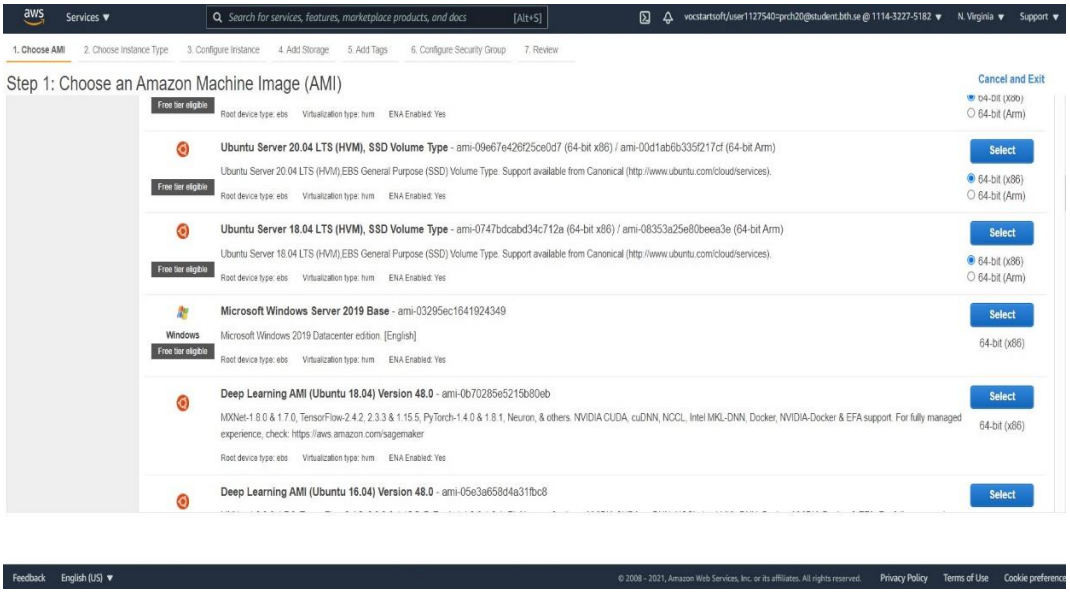
IMPLEMENTATION:

STEP-1: LAUNCHING THE EC2 INSTANCE

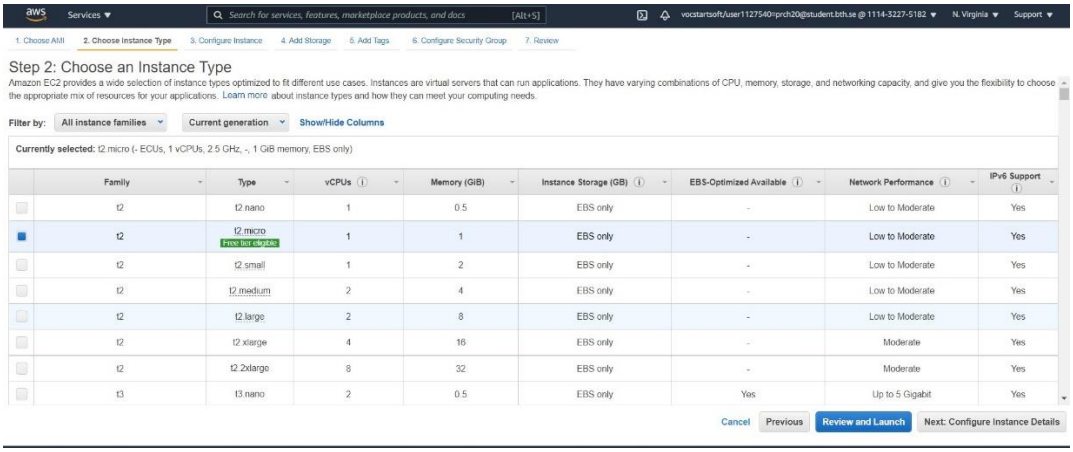
1. Firstly, key pair should be created to launch the EC2 instance.



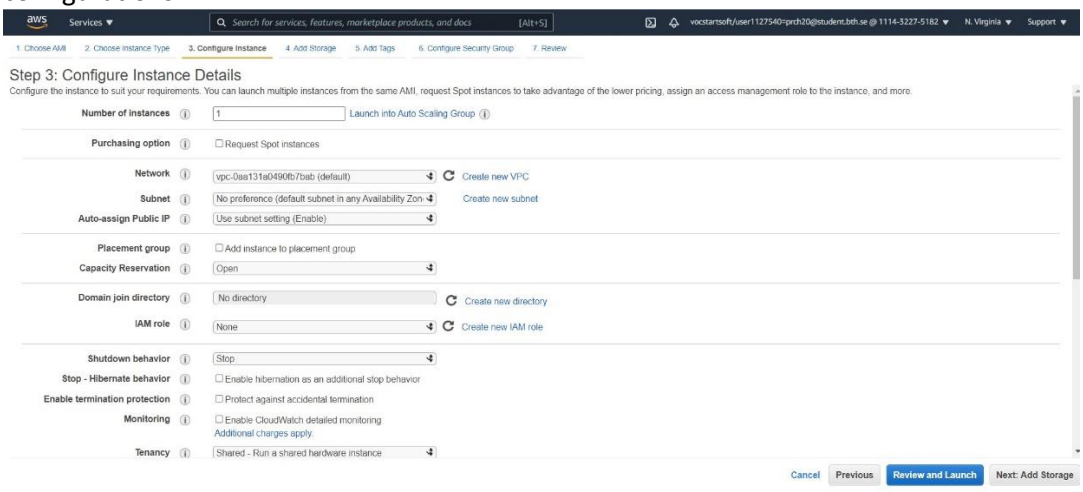
2. On the AWS console choose EC2 instance option and click on launch instance button.
3. Select the ubuntu server image from the list of images (AMI).



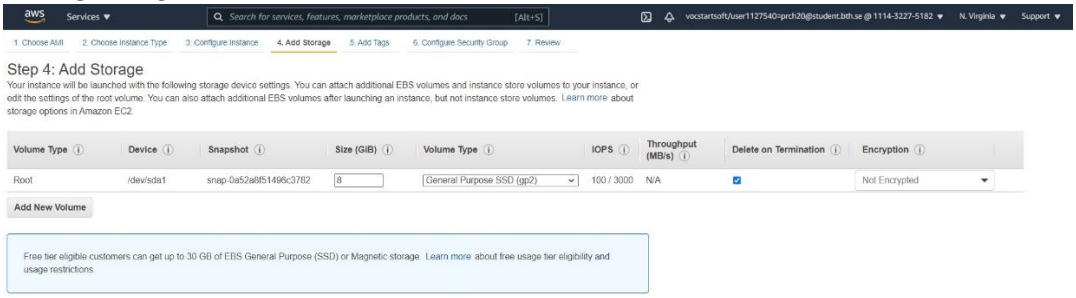
4. Choosing the type of instance i.e., t2 micro.



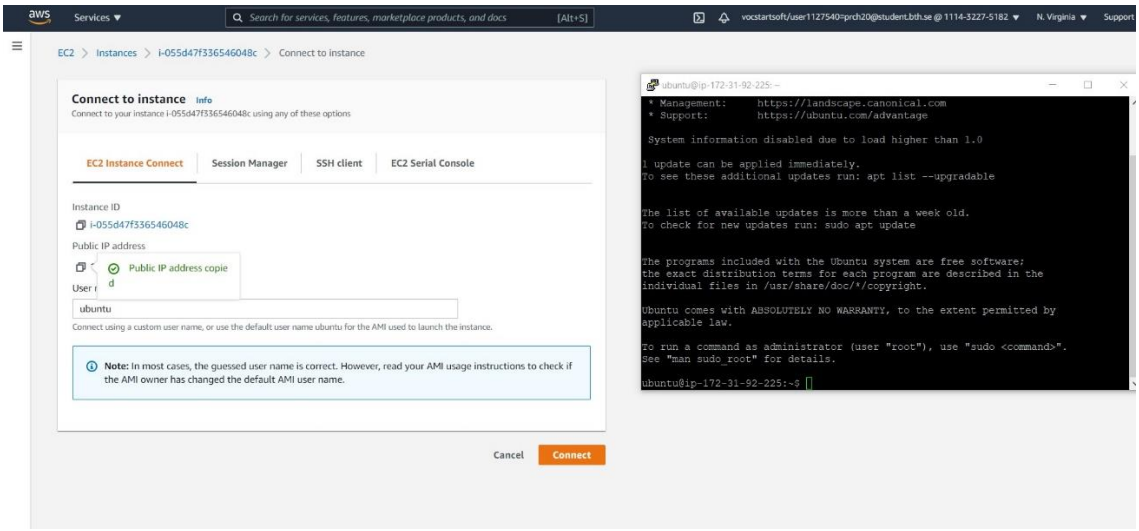
5. Configuring the instance where we select number of instances, network and subnet configurations.



6. Adding storage



7. Configuring the security group.
8. Click on Launch instance button and to access the created instance we need to use putty on the system and launch the instance using IP address and previously created key file. the instance will be launched as shown in the below figure.



STEP-2: DEPLOYING THE CALCULATOR APP

To deploy the flask application in ubuntu it is important to implement the following commands in putty:

i. Updating existing packages – as the ubuntu server is a new operating system where it contains outdated packages, so it is important to update the existing packages in the operating system.

cmd: sudo apt-get update

```
ubuntu@ip-172-31-92-225: ~
Using username "ubuntu".
Authenticating with public key "Flask-AWS"
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1045-aws x86_64)

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

 System information disabled due to load higher than 1.0

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-92-225:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [11
4 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [
101 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 Packag
es [8628 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/universe Translation-
en [5124 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f
Metadata [265 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse amd64 Pack
ages [144 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse Translatio
n-en [104 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n
-f Metadata [9136 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 P
ackages [1127 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main Transla
```

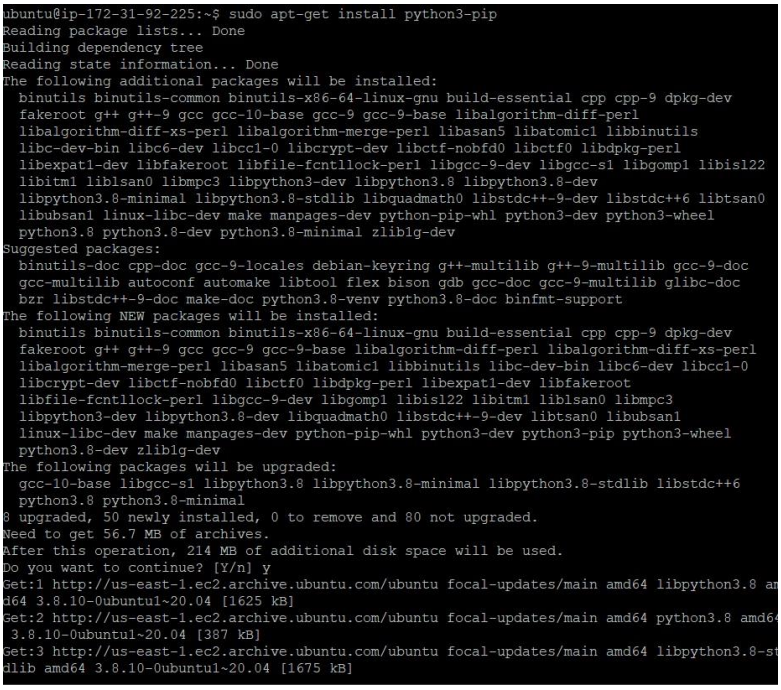
ii. Installing python – to run the flask applications it is important to have python in our ubuntu server.

cmd: sudo apt-get install python3

```
Fetched 19.6 MB in 3s (5775 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-92-225:~$ sudo apt-get install python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.8.2-0ubuntu2).
python3 set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 88 not upgraded.
ubuntu@ip-172-31-92-225:~$
```

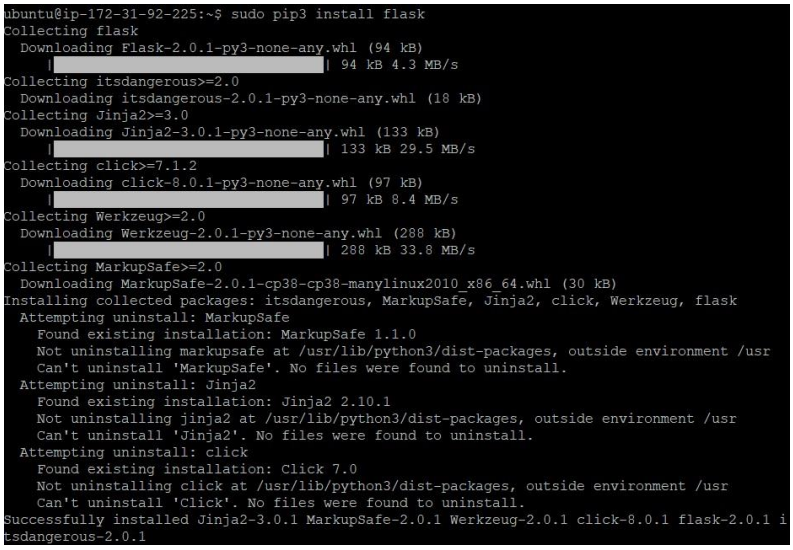
iii. Installing pip i.e python package manager – after installing python pip is installed into the ubuntu server which allows us to install and manage additional libraries and dependencies that are not distributed as part of python library.

cmd: sudo apt-get install python3-pip



iv. Downloading flask into our server – as we are deploying a flask application our server needs to be installed with supported package called flask.

cmd: sudo pip3 install flask

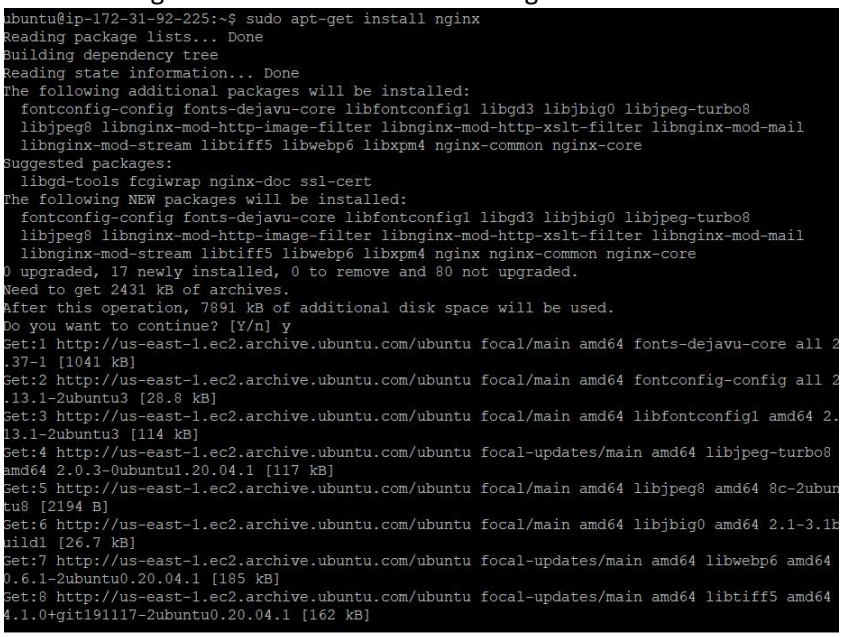


v. Ngnix is used as reverse proxy and load balancer. We will be using it as web server for deploying our web application along with that we are using guicorn for the purpose of python web application.

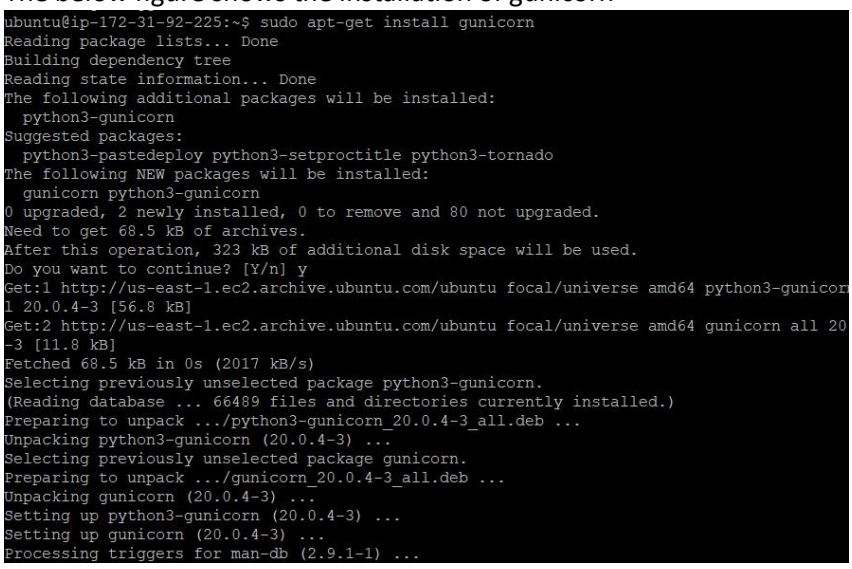
cmd: sudo apt-get install ngnix

cmd: sudo apt-get install gunicorn3

The below figure shows the installation of nginx

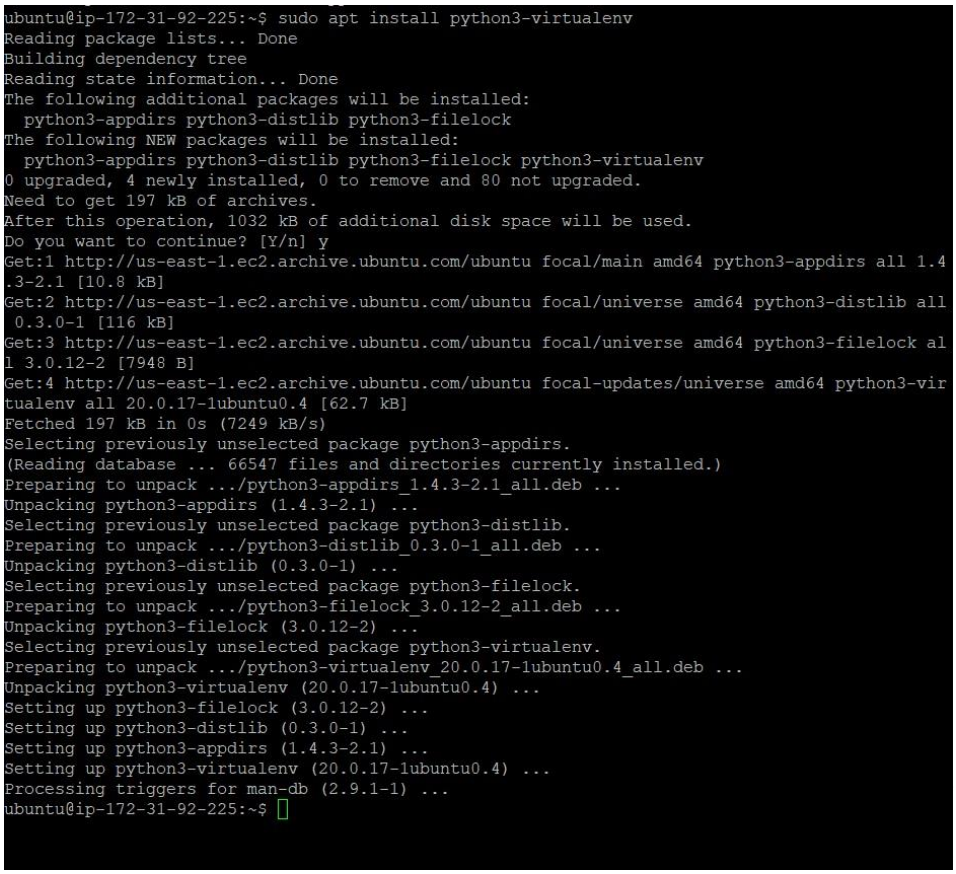


The below figure shows the installation of gunicorn



vi. A virtual environment is created for installing and running the flask application in an isolated python environment.

cmd: sudo apt install python3-virtuallenv



vii. To install the flask application, we are going to create a new directory

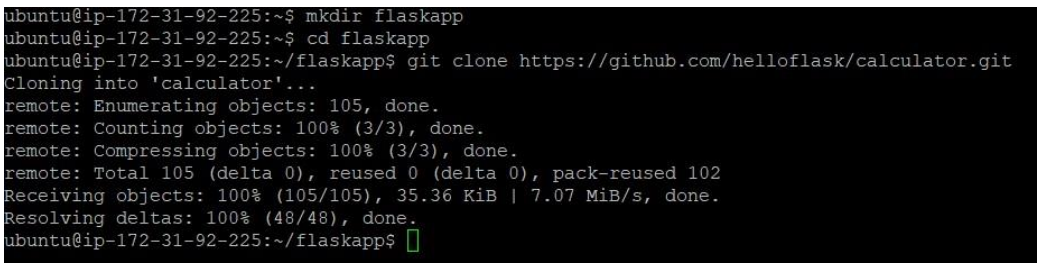
cmd: mkdir flaskapp

viii. Go to created directory.

cmd:cd flaskapp

ix. Our flask application is present in the git repository to clone the repository into our web server the following command is used.

cmd: git clone <git link>



x. To run the application we need to initiate virtual environment th efollwing commands are used to initiate the virtual environment.

cmd: virtuallenv venv

cmd: source venv/bin/activate

```
ubuntu@ip-172-31-92-225:~/flaskapp$ virtualenv venv
created virtual environment CPython3.8.10.final.0-64 in 185ms
  creator CPython3Posix(dest=/home/ubuntu/flaskapp/venv, clear=False, global=False)
  seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest, pkg_resourc
es=latest, via=copy, app_data_dir=/home/ubuntu/.local/share/virtualenv/seed-app-data/v1.0.1.d
ebian.1)
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,
XonshActivator
ubuntu@ip-172-31-92-225:~/flaskapp$ ls
calculator  venv
ubuntu@ip-172-31-92-225:~/flaskapp$
```

xi. The following command is used to download all the required files into the webserver to run the flask application in the virtual environment.

cmd: pip install -r calculator/requirements.txt

```
ubuntu@ip-172-31-92-225:~/flaskapp$ source venv/bin/activate
(venv) ubuntu@ip-172-31-92-225:~/flaskapp$ pip install -r calculator/requirements.txt
Collecting Flask==0.12.2
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83 kB)
     |████████████████████████████████| 83 kB 1.9 MB/s
Collecting Jinja2==2.8
  Downloading Jinja2-2.8-py2.py3-none-any.whl (263 kB)
     |████████████████████████████████| 263 kB 33.2 MB/s
Collecting Werkzeug==0.13
  Downloading Werkzeug-0.13-py2.py3-none-any.whl (311 kB)
     |████████████████████████████████| 311 kB 44.6 MB/s
Collecting gunicorn==19.5.0
  Downloading gunicorn-19.5.0-py2.py3-none-any.whl (113 kB)
     |████████████████████████████████| 113 kB 50.7 MB/s
Collecting gevent
  Downloading gevent-21.1.2-cp38-cp38-manylinux2010_x86_64.whl (6.3 MB)
     |████████████████████████████████| 6.3 MB 37.9 MB/s
Collecting itsdangerous>=0.21
  Downloading itsdangerous-2.0.1-py3-none-any.whl (18 kB)
Collecting click>=2.0
  Downloading click-8.0.1-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 9.3 MB/s
Collecting MarkupSafe
  Downloading MarkupSafe-2.0.1-cp38-cp38-manylinux2010_x86_64.whl (30 kB)
Collecting zope.event
  Downloading zope.event-4.5.0-py2.py3-none-any.whl (6.8 kB)
Collecting greenlet<2.0,>=0.4.17; platform_python_implementation == "CPython"
  Downloading greenlet-1.1.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (164 kB
)
     |████████████████████████████████| 164 kB 40.8 MB/s
Collecting zope.interface
  Downloading zope.interface-5.4.0-cp38-cp38-manylinux2010_x86_64.whl (259 kB)
     |████████████████████████████████| 259 kB 39.2 MB/s
Requirement already satisfied: setuptools in ./venv/lib/python3.8/site-packages (from gevent-
>-r calculator/requirements.txt (line 6)) (44.0.0)
Installing collected packages: MarkupSafe, Jinja2, Werkzeug, itsdangerous, click, Flask, guni
corn, zope.event, greenlet, zope.interface, gevent
Successfully installed Flask-0.12.2 Jinja2-2.8 MarkupSafe-2.0.1 Werkzeug-0.13 click-8.0.1 gev
ent-21.1.2 greenlet-1.1.0 gunicorn-19.5.0 itsdangerous-2.0.1 zope.event-4.5.0 zope.interface-
5.4.0
(venv) ubuntu@ip-172-31-92-225:~/flaskapp$
```

xii. The following commands are used to change the directory where the flask application is located and run the flask application.

cmd: cd calculator

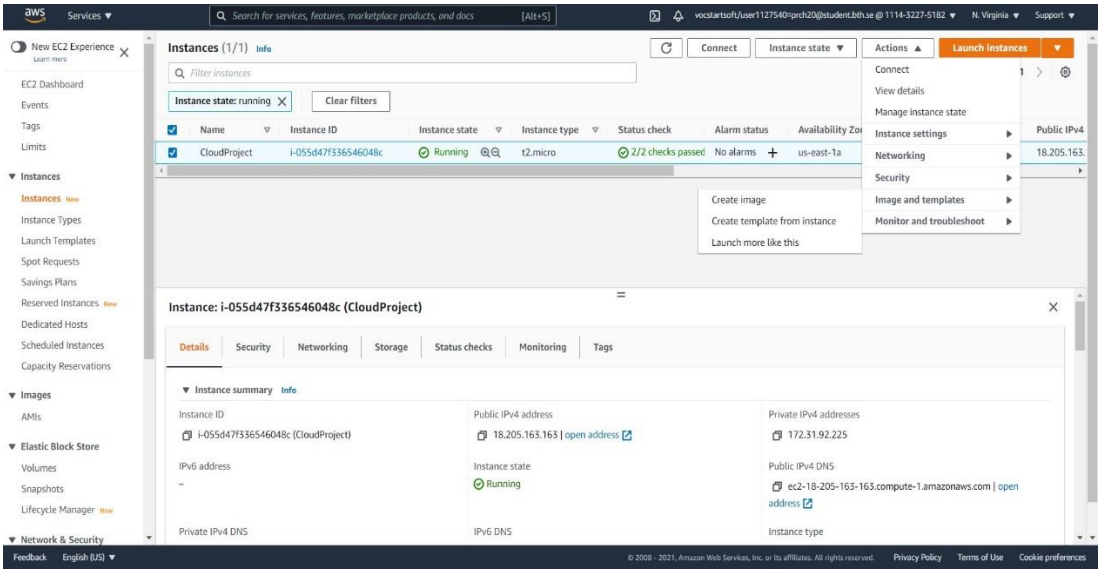
cmd: python3 app.py

```
(venv) ubuntu@ip-172-31-92-225:~/flaskapp/calculator$ python3 app.py
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
193.11.185.72 - - [27/Jul/2021 10:16:35] "GET / HTTP/1.1" 200 -
193.11.185.72 - - [27/Jul/2021 10:16:35] "GET /static/css/style.css HTTP/1.1" 200 -
193.11.185.72 - - [27/Jul/2021 10:16:35] "GET /static/js/main.js HTTP/1.1" 200 -
193.11.185.72 - - [27/Jul/2021 10:16:35] "GET /static/banner.png HTTP/1.1" 200 -
193.11.185.72 - - [27/Jul/2021 10:16:37] "GET /static/favicon.ico HTTP/1.1" 200 -
193.11.185.72 - - [27/Jul/2021 10:16:45] "GET /_calculate?number1=7&operator=%2B&number2=3 HT
TP/1.1" 200 -
```

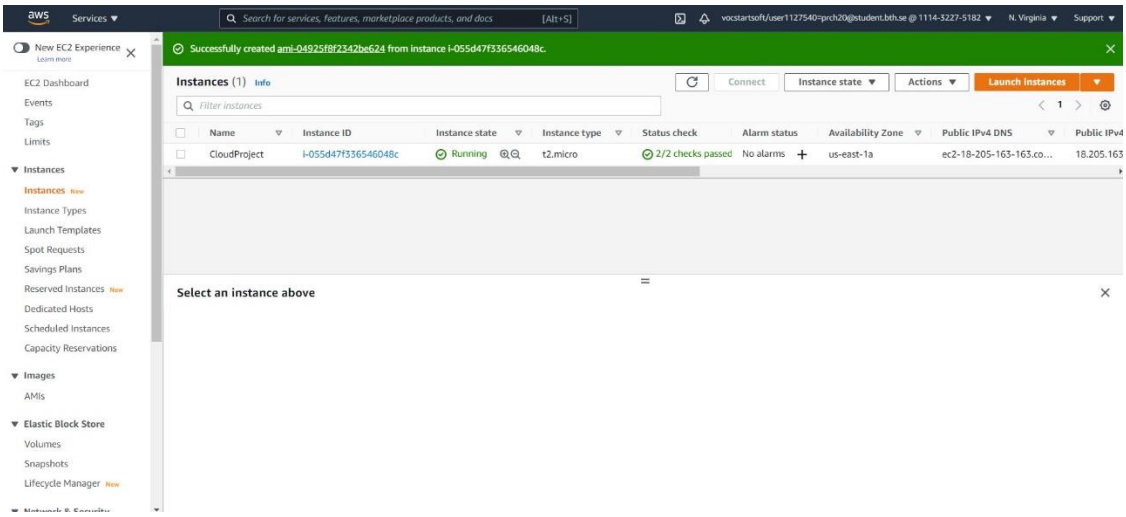
xiii. Launching the flask web application in the web browser using the public IP address and port number: 8080 of our ubuntu webserver.

STEP-3: IMAGE FOR THE CREATED INSTANCE

In this step we are going to create image for our instance where this image is used to launch the additional EC2 instances when there is stress generated on our flask web application so that a greater number of people can use this application without any interruption. The below image shows how to create the image for our instance in AWS.



After successful creation of image for our instance we get the results as shown in the figure below.

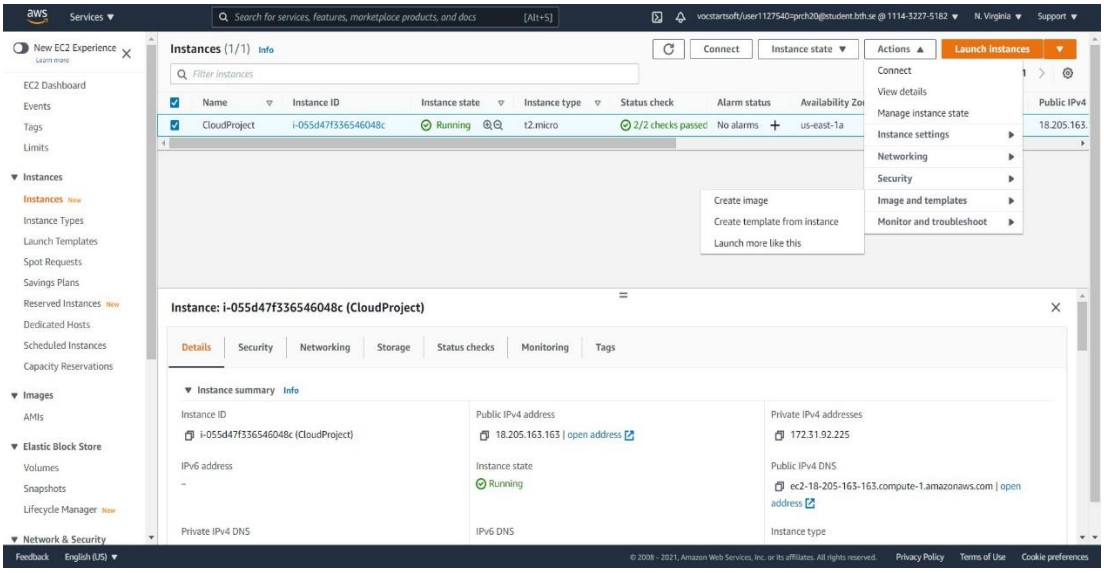


STEP 4: TEMPLATE CREATION FOR INSTANCE.

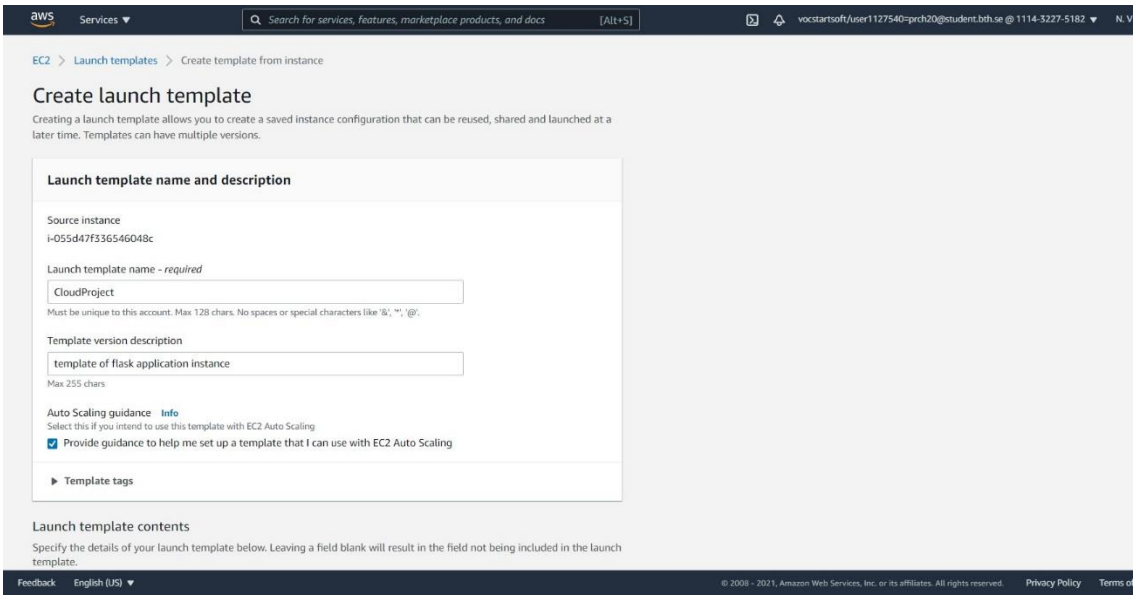
In this step we are going to create template for our instance where this template contains all the necessary instance configuration information that includes id of our instance, instance type, a key pair, security groups and other parameters that we used to launch EC2 instance.

Launch templates are very streamline and simplify the launch process for autoscaling and on demand instances. It reduces the number of steps required to create an instance by capturing all launch parameters within one resource and this makes the process simple and to reproduce.

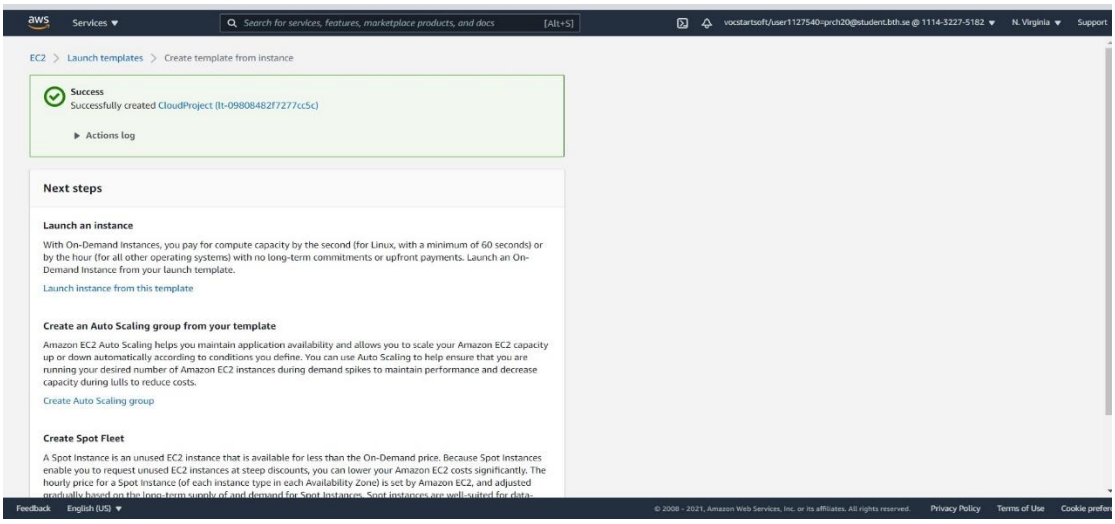
The below figure shows how to initiate the creation of template for our instance.



The below figure shows the steps involved in creating instance templates.



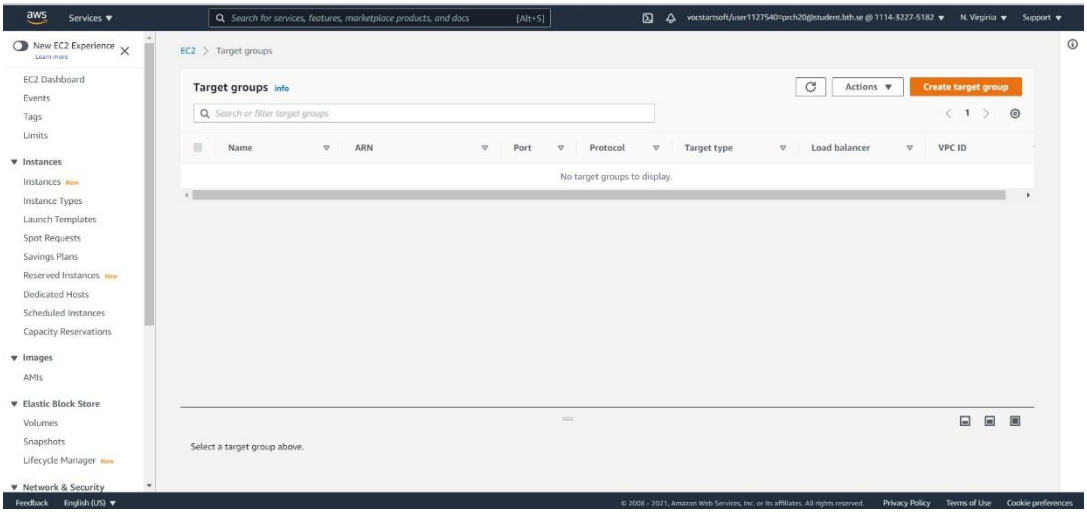
After successful creation of templates to our image we get the results as shown in the figure below.



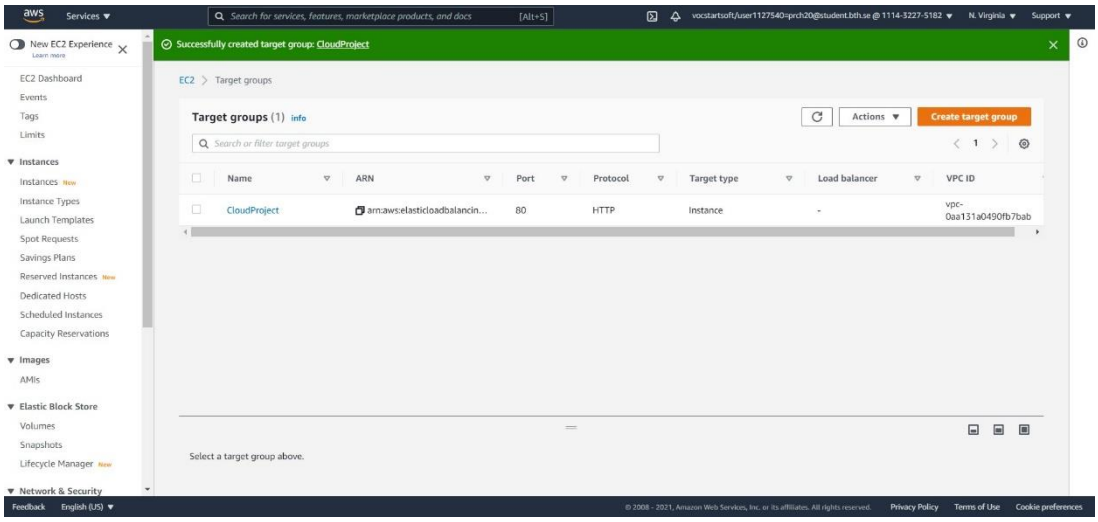
STEP 5: CREATING TARGET GROUPS FOR AUTOSCALING

In this step we are going to create a target group that is used to inform the load balancer or initiate the load balancer where to direct the traffic to: EC2 instances, fixed IP addresses. In this target groups we configure the instance that we are target and its related image.

The below figure explains about the creation of target groups for our instance.



The below figure shows the successful creation of target groups.



STEP-6: AUTO SCALING OF OUR INSTANCE

In this project we are going to use auto scaling of our instance when there is excess data generated while using the flask application by greater number of users.

The main purpose of autoscaling is that it allows us to automatically add or remove EC2 instances according to the conditions that we define while setting up autoscaling policy.

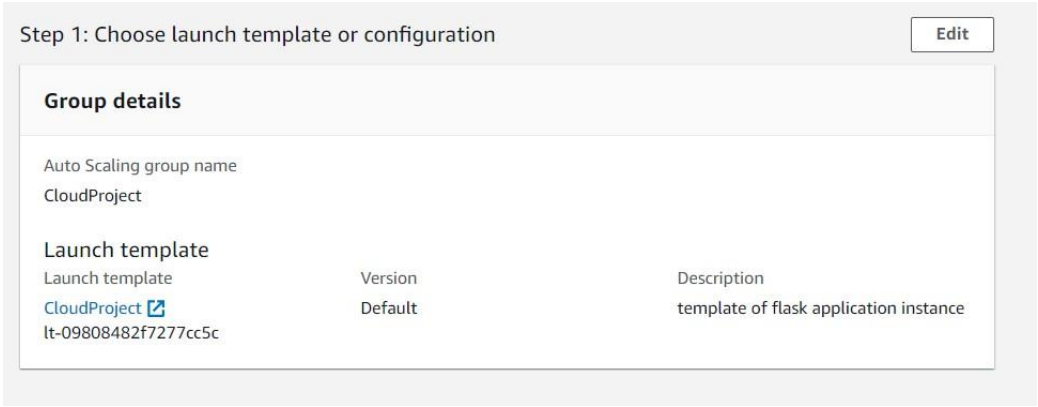
The benefits of autoscaling are

- It can detect an unhealthy instance, terminate it, and replace it with the new one.
- It ensures that our application always has the right amount of compute and proactively provisions capacity with predictive scaling.

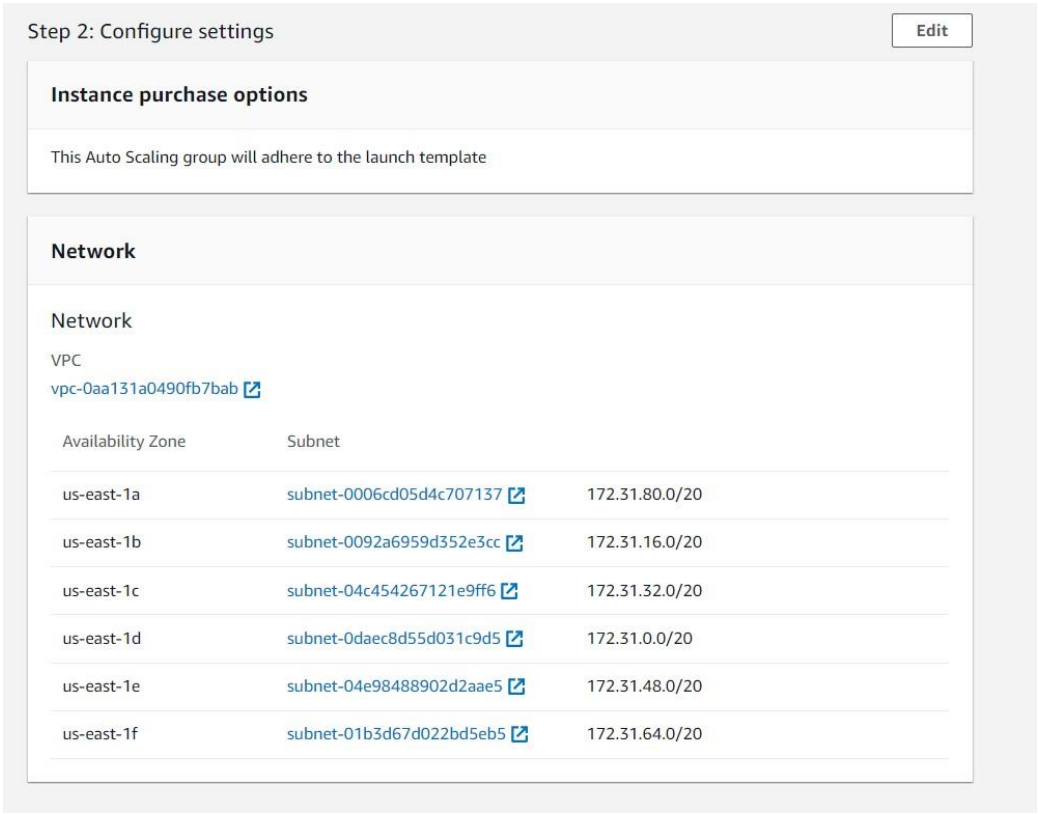
- It only adds instances when needed and it can scale across different purchase options to optimize the price and performance.

In this project the following steps are performed in creating and configuring autoscaling to our instance.

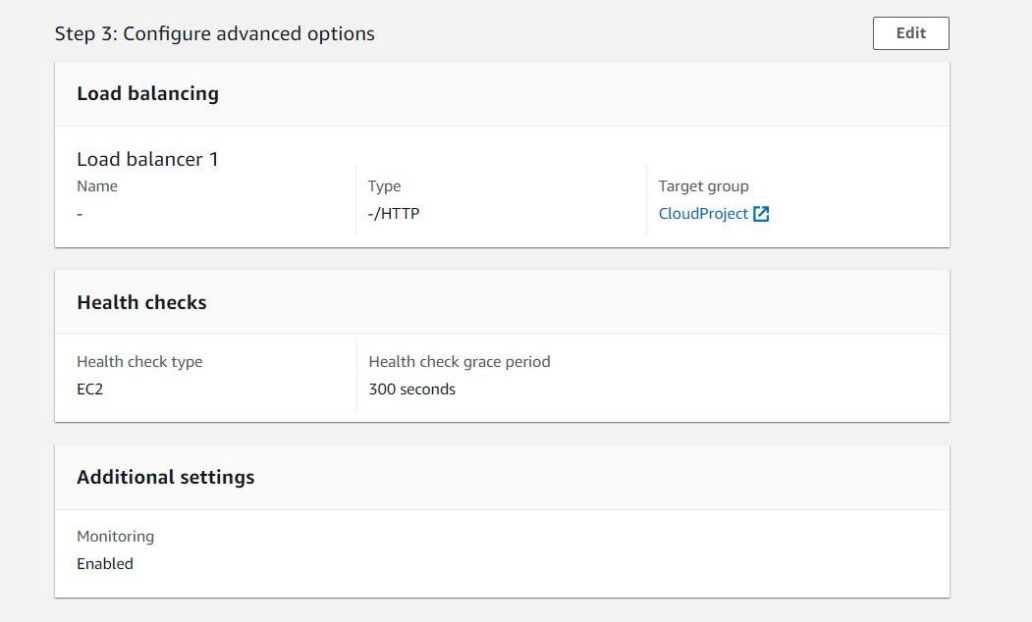
1. Choose launch template or configuration.



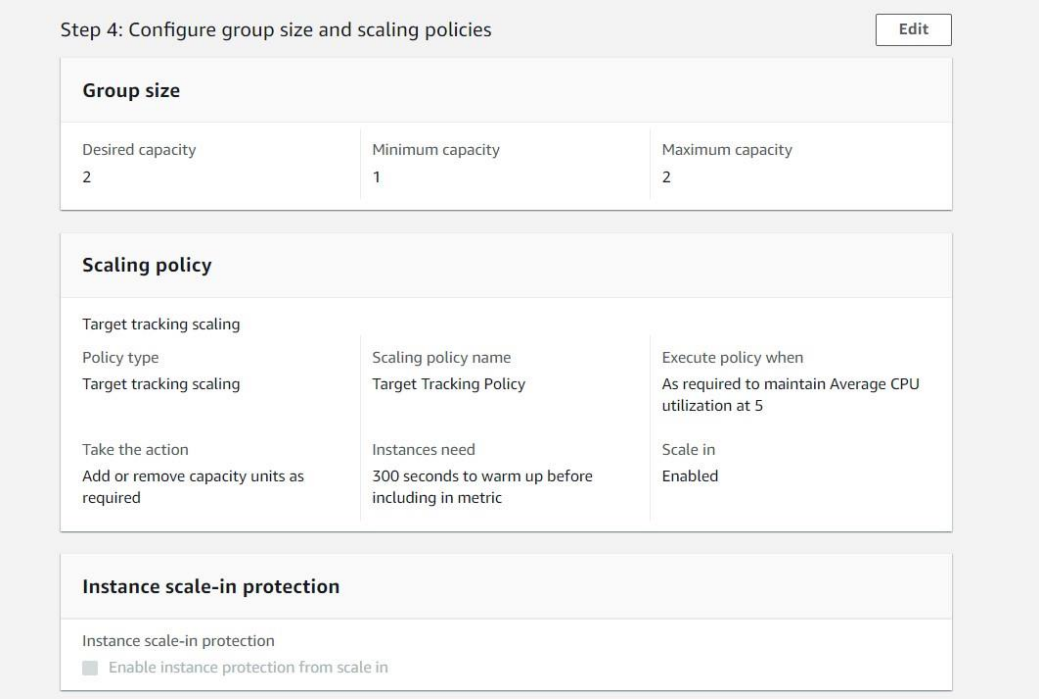
2. Configure settings.



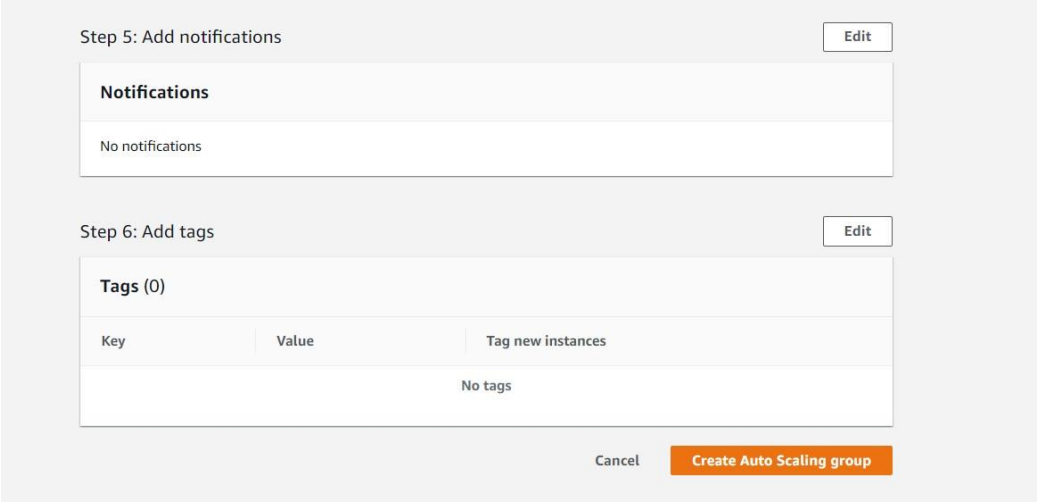
3. Configure advanced options.



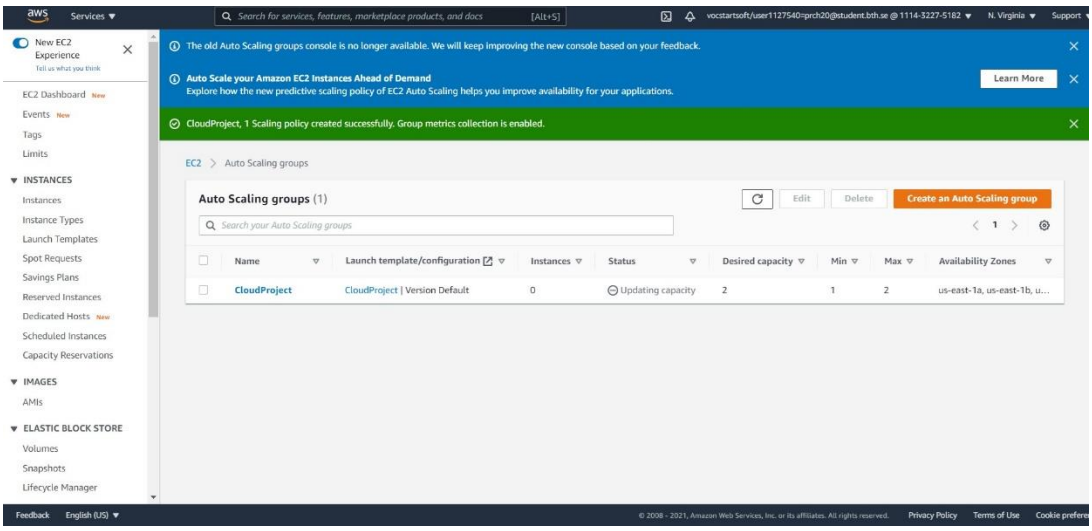
4. Configure group size and scaling policies



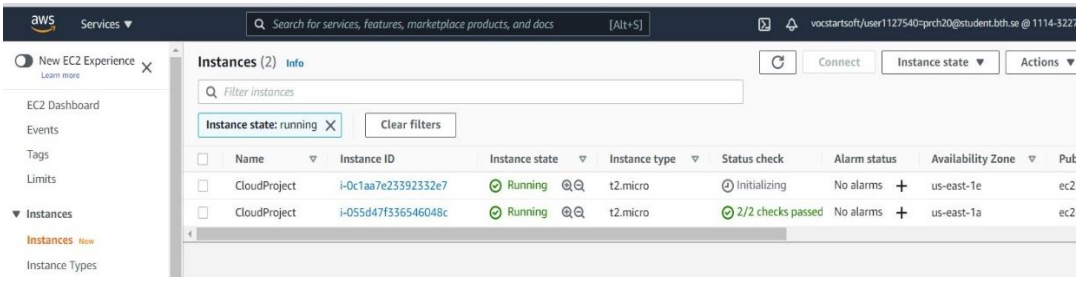
5. Adding notifications and tags



6. After performing above steps auto scaling group is successfully created as shown in the figure below.



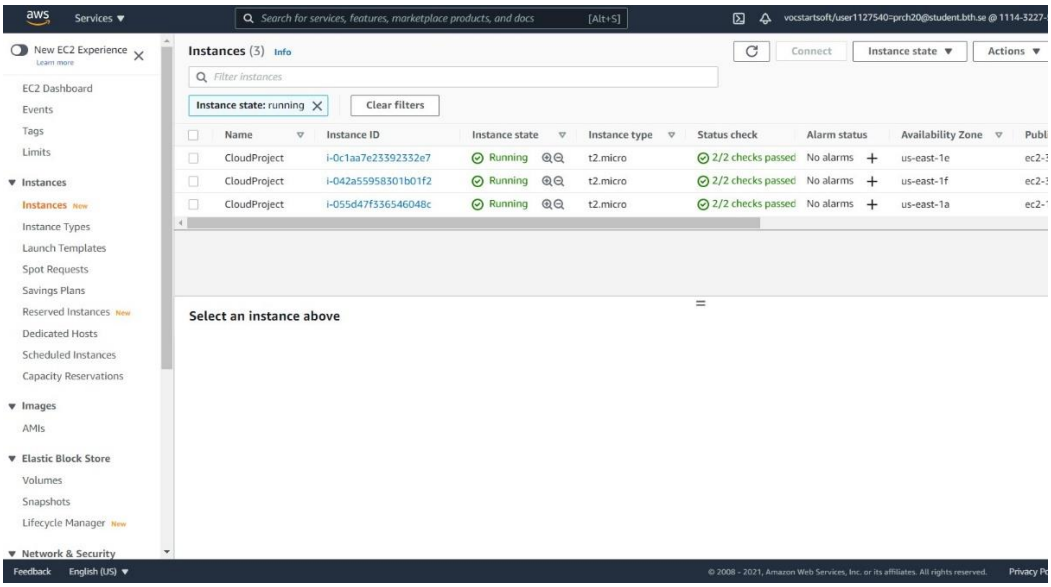
After setting up of auto scaling policy group for the instance “CloudProject” we have given the desired minimum capacity as 1 with similar configuration then an additional instance is activated as shown in the below figure.



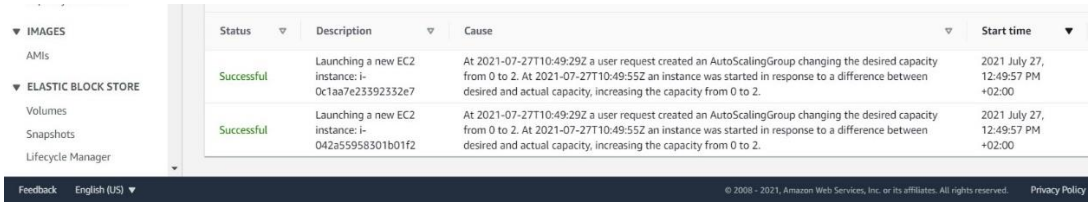
As the number of requests and data increases as n number of users access the application there is an increase in the load on CPU. Therefore, another instance is launched when the threshold value is greater than 20.

The max capacity of our auto scaling group is 2, therefore another resource is added to the instance CloudProject after satisfying the auto scaling policies.

The below figure shows the added resources to our instance.



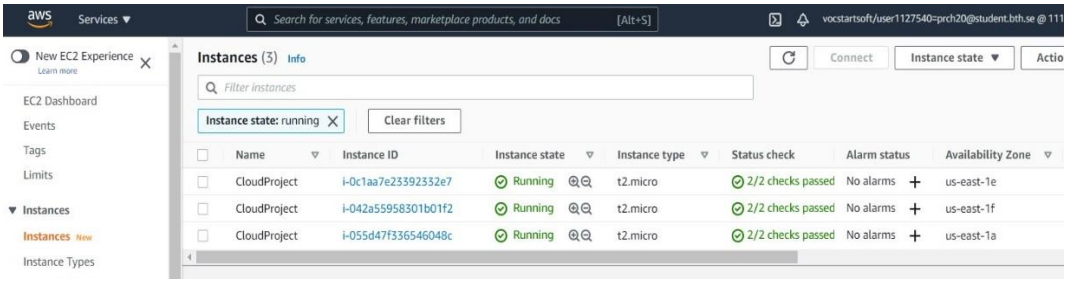
The below image shows the monitoring of activity of auto scaling policy group.



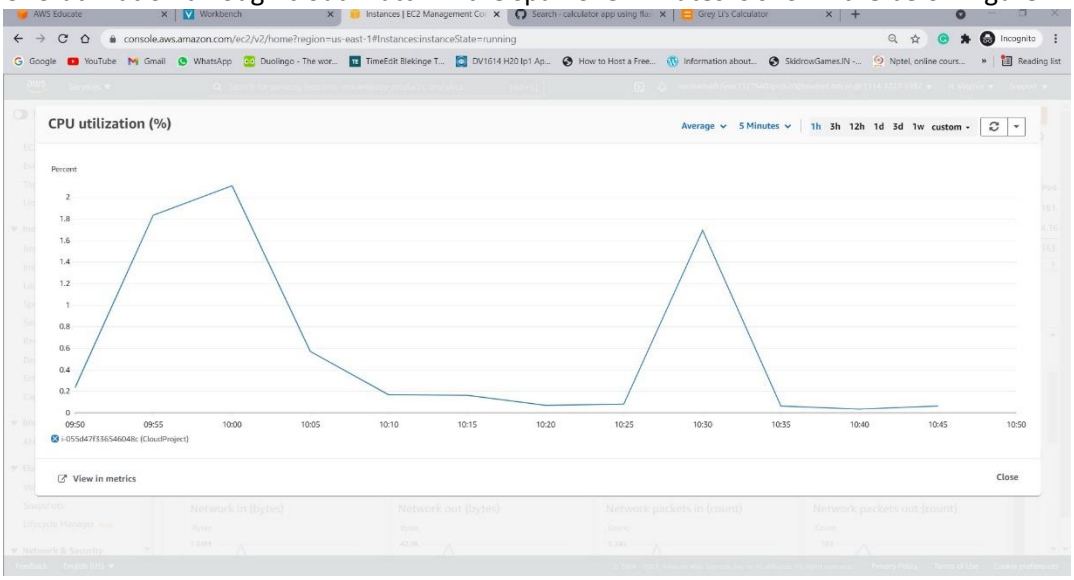
Validation:

- Deploying the flask web application in the instance and accessing the application using public DNS IP address 18.205.163.163 and port number 8080.

- After reaching max CPU threshold value by 20, a new resource is added in the instance list.



- CPU utilization through cloud watch in the span of 5 minutes is shown the below figure.



Results:

- Scalability with respect to data is achieved in this project by adding the required number of resources to the instance according to the conditions present in the auto scaling policy group.

- Here the scalability is obtained as there is an exceeded in the number of users of the application the load on the CPU increased that the average threshold.

- The security of data at rest is achieved by utilizing the AWS KMS feature as the access to encrypt or decrypt the data within the service is independently controlled by AWS KMS policies under the customer's control, customers can isolate control over access to the data, from access to the keys.

- We tried or utilize this feature to secure the data of our application using client-side encryption in AWS KMS, but the results obtained were not appropriate to the proposed solution. Though we have learned a lot during this project.