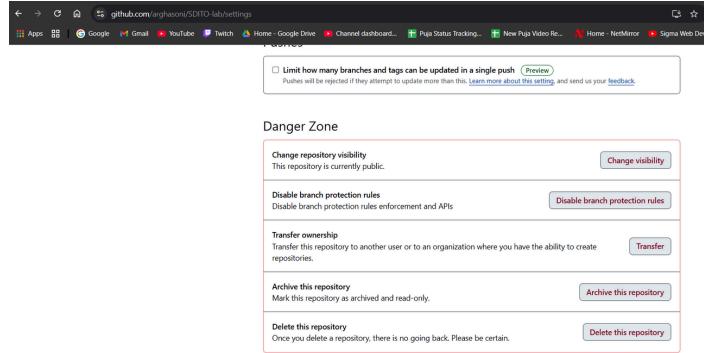


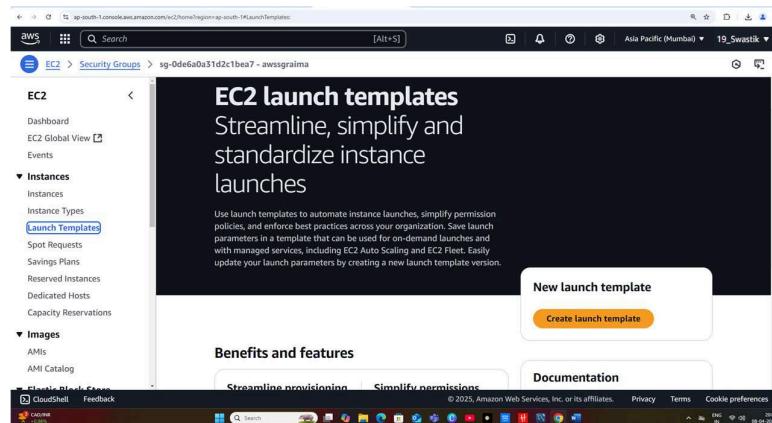
# ASSIGNMENT NO :- 11

TITLE :- Build scaling plans in AWS that balance the load on different EC2 instances.

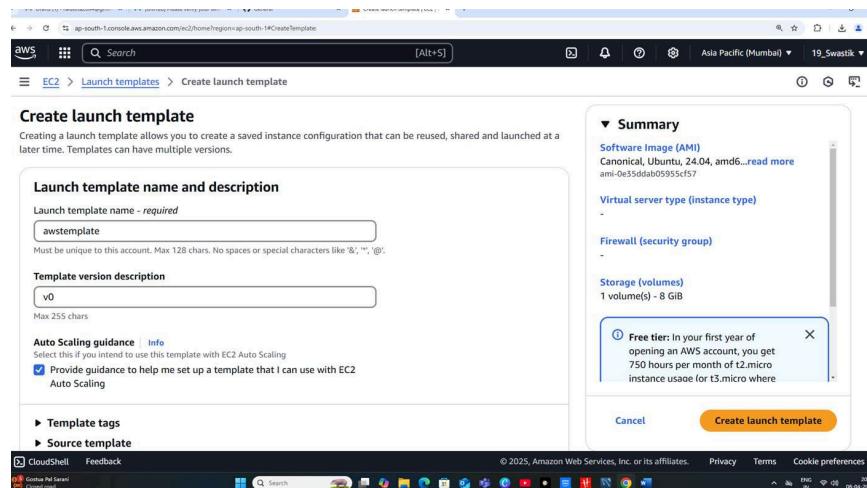
STEP 1 :- Check if your GitHub Repository is public or not .



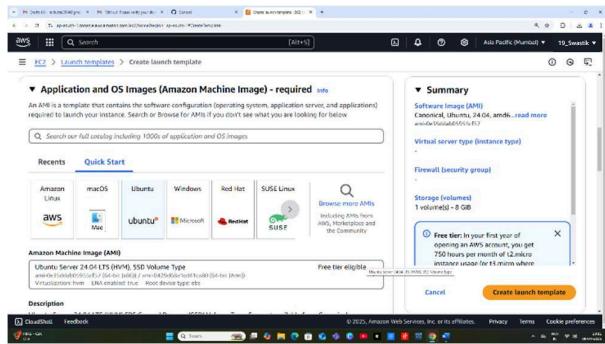
STEP 2 :- Login to your AWS Account and then go to Launch Template and click on Create Launch Template



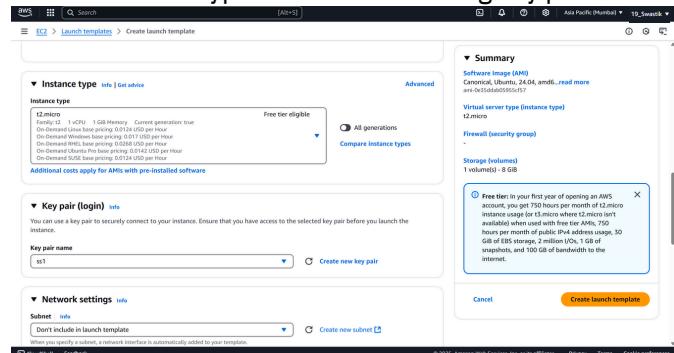
STEP 3:- Give your Template name and version description and select Auto Scaling guidance.



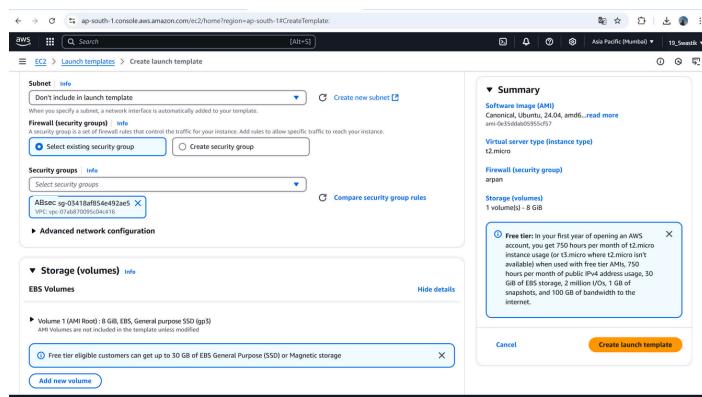
>> Select Ubuntu in Quickstart



>> Select t2.micro as Instance type and use the existing key pair you created

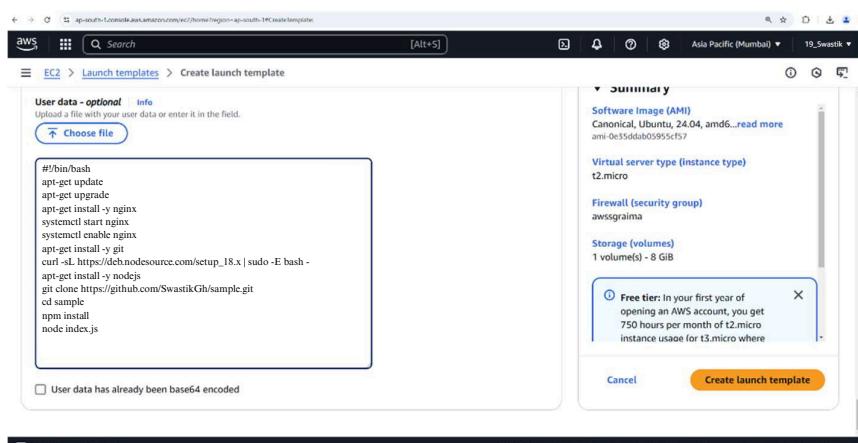


>>

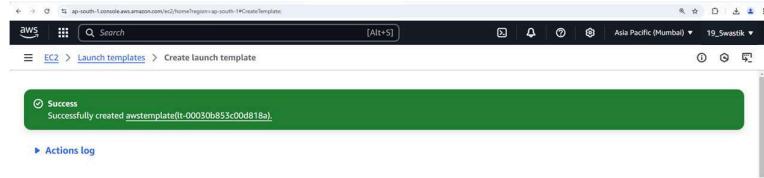


>> then click on Advanced Details

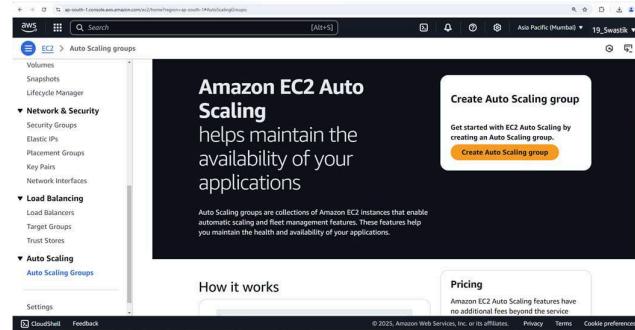
>> write the command line as written below and add your Repository link in Git Clone <Repository Link>



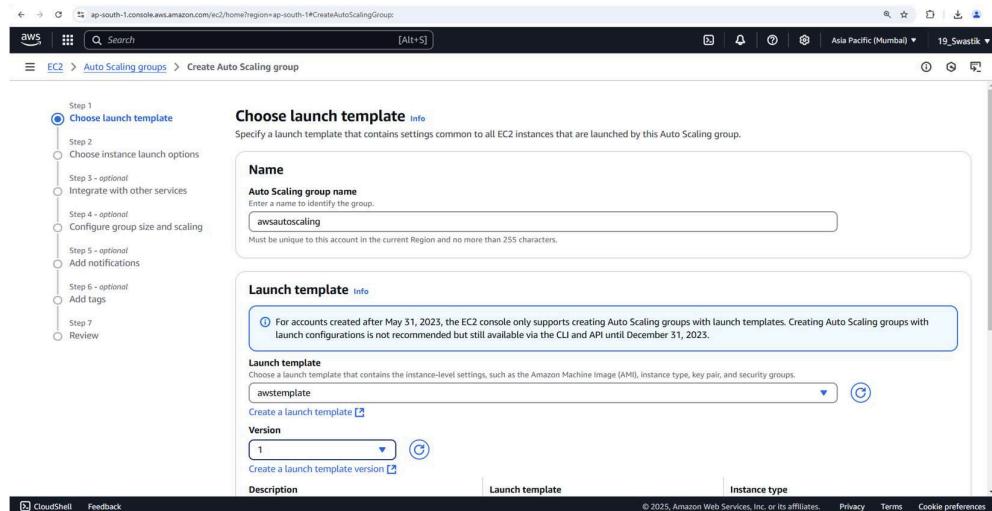
>> Your Template is created successfully.



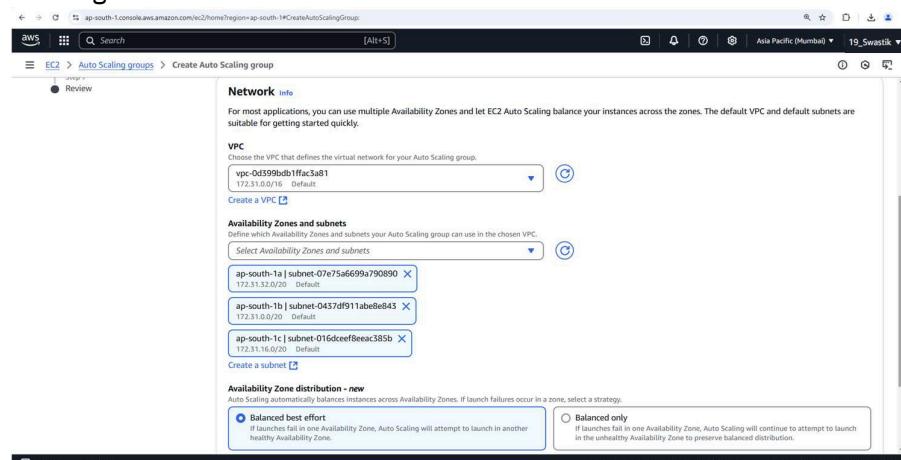
STEP 5:- Click on Auto Scaling Groups and then click on Create Auto Scaling groups.



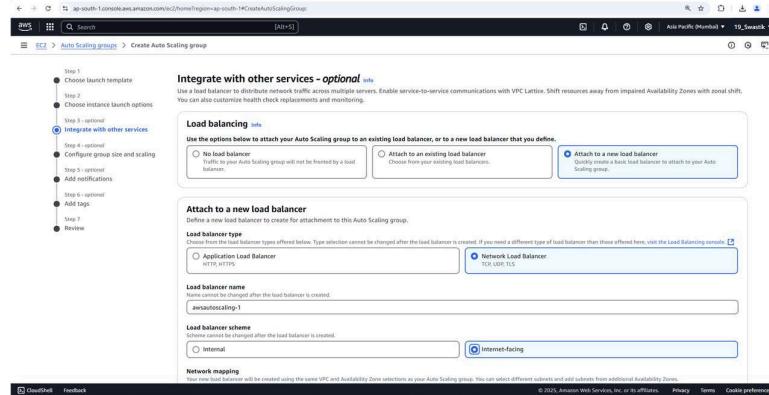
>> Give a name and select your created Template and Give the version as Latest (1) and click on Next



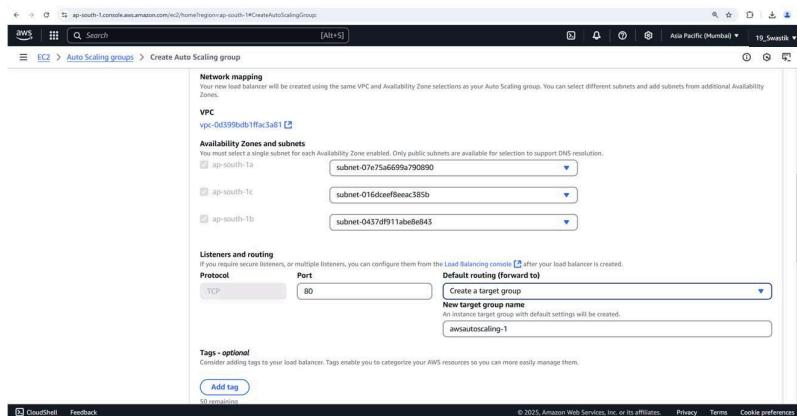
>> In Network region Select the Three Available Zones as servers and click on next



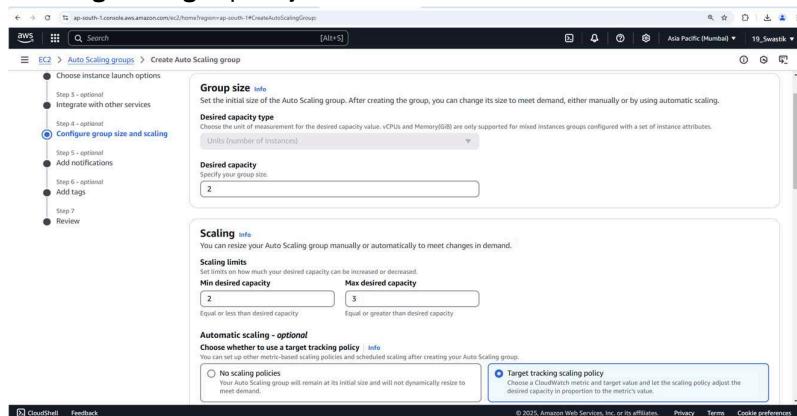
>> Select Attach a new load balancer and then select Network load balancer and then select Internet Facing



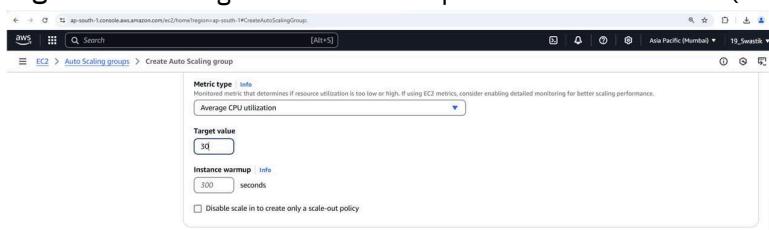
>> click on Create Target Group and select a target group name and then click on next



>> Now select your Desired capacity , Min desired and max desired capacity and then select Target Tracking Scaling capacity.



>> Give the Target Value and give Instance Runup and then click on Next (3times)



>>Review and click on Create Auto Scaling Group

The screenshot shows the AWS Auto Scaling Groups page. It displays a table with one row for the group 'awsautoscaling'. The table columns include Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and Availability Zones. The instance details show 2 instances, both running, with a status of 'InService'. The availability zones listed are ap-south-1c and ap-south-1d.

STEP 5:- As you can see the Instances are automatically created if you will delete one instance then the 3rd one will automatically generated as 3 servers are given

The screenshot shows the AWS EC2 Instances page. It displays a table with three rows of instance data. The instances are all in the 'Running' state with a status of 'InService'. They are located in the 'ap-south-1a' availability zone. The instance IDs listed are i-000562087380855d, i-05734e655f10121e, and i-0144ed7793740211.

STEP 6:- Select any one Instance and Copy its IPV4 address and paste it in Incognito website with port number( IP Address : 4000)

Hellooo!! This is Argha from MCKVIE

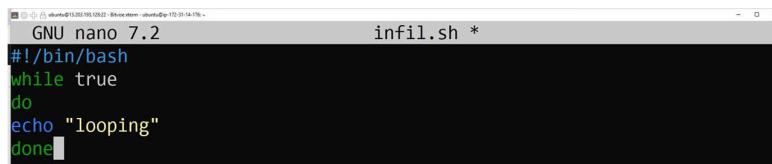
STEP 7:- Paste the address in Bitvise Server and give the required field of username , Initial method and Client key then click on LOG IN and go to the New Terminal

The screenshot shows the Bitvise SSH Client interface. The main window title is 'ubuntu@13.203.193.128:2 - Bitvise SSH Client'. The 'Default profile' tab is selected. In the 'Server' section, the host is set to '13.203.193.128' and the port is '4000'. The 'Authentication' section shows the username 'ubuntu', initial method 'publickey', and client key 'Global 1'. The terminal pane at the bottom shows a log of SSH session activity, including messages about host key synchronization and terminal channel openings.

>> Then write the following command in terminal to open a shell file

```
ubuntu@ip-172-31-14-176:~$ nano infil.sh
```

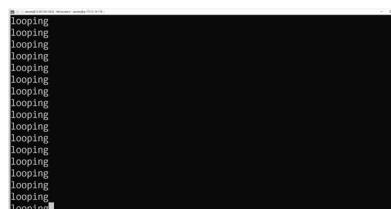
>> Then write the following code in the shell file



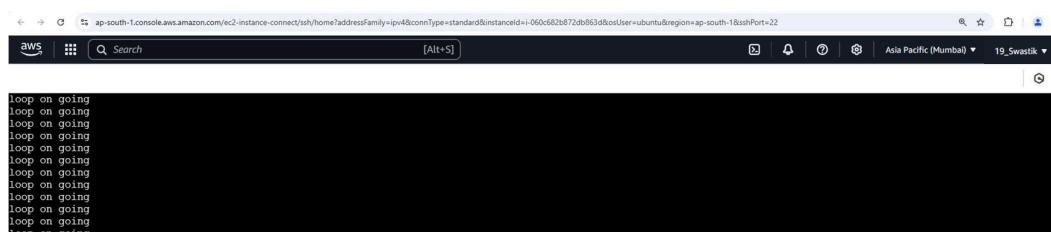
```
GNU nano 7.2
#!/bin/bash
while true
do
echo "looping"
done
```

>> Compile and run the File Using the following command

```
ubuntu@ip-172-31-14-176:~$ chmod +x infil.sh
ubuntu@ip-172-31-14-176:~$ ./infil.sh
```



STEP 6:- Select other Instance and click on Connect and Then write the same command and same code in terminal and then run it.



>> Then click on both the instances and you can see the graph of both



# ASSIGNMENT NO :- 12

TITLE :- Deploy and run the project in AWS without using the port.

## STEP 1 :- Create a new EC2 Instance

The screenshot shows the AWS EC2 Instances page. A list of instances is displayed, with one instance named 'aws12' selected. The details pane for 'aws12' shows the following information:

- Public IPv4 address:** 3.110.56.204 | [open address](#)
- Private IP DNS name (IPv4 only):** ip-172-31-0-38.ap-south-1.compute.internal
- Instance type:** t2.micro

## STEP 2 :- Connect the instance with the server.

The screenshot shows the 'Connect to instance' page for the selected EC2 instance. It displays several connection options:

- EC2 Instance Connect** (selected)
- Session Manager**
- SSH client**
- EC2 serial console**

The 'Public IPv4 address' (3.110.56.204) is selected for connection. Other options include 'Connect using EC2 Instance Connect Endpoint' and 'IPv6 address'. A note at the bottom states: "Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username." A 'Connect' button is visible at the bottom right.

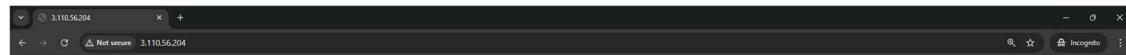
## STEP 4:- Wite the following command in terminal.

```
ubuntu@ip-172-31-0-38:~$ cd ..
ubuntu@ip-172-31-0-38:/home$ cd ..
ubuntu@ip-172-31-0-38:$ cd etc
ubuntu@ip-172-31-0-38:/etc$ cd nginx
ubuntu@ip-172-31-0-38:/etc/nginx$ cd sites-available
ubuntu@ip-172-31-0-38:/etc/nginx/sites-available$ ls
default
ubuntu@ip-172-31-0-38:/etc/nginx/sites-available$ sudo nano default
ubuntu@ip-172-31-0-38:/etc/nginx/sites-available$ sudo chmod 777 default
ubuntu@ip-172-31-0-38:/etc/nginx/sites-available$ sudo systemctl restart nginx
ubuntu@ip-172-31-0-38:/etc/nginx/sites-available$
```

>> Now Open the default file and write the following code.

```
#       location / {
#           # First attempt to serve request as file, then
#           # as directory, then fall back to displaying a 404.
#           try_files $uri $uri/ =404;
#       }
#       location / {
#           proxy_pass http://localhost:4000;
#           proxy_http_version 1.1;
#           proxy_set_header Upgrade $http_upgrade;
#           proxy_set_header Connection 'Upgrade';
#           proxy_set_header Host $host;
#           proxy_cache_bypass $http_upgrade;
#       }
#       # pass PHP scripts to FastCGI server
#       #
#       location ~ \.php$ {
```

STEP 5:- Now copy the IP address of the instance and check the server without PORT number.



Swastik