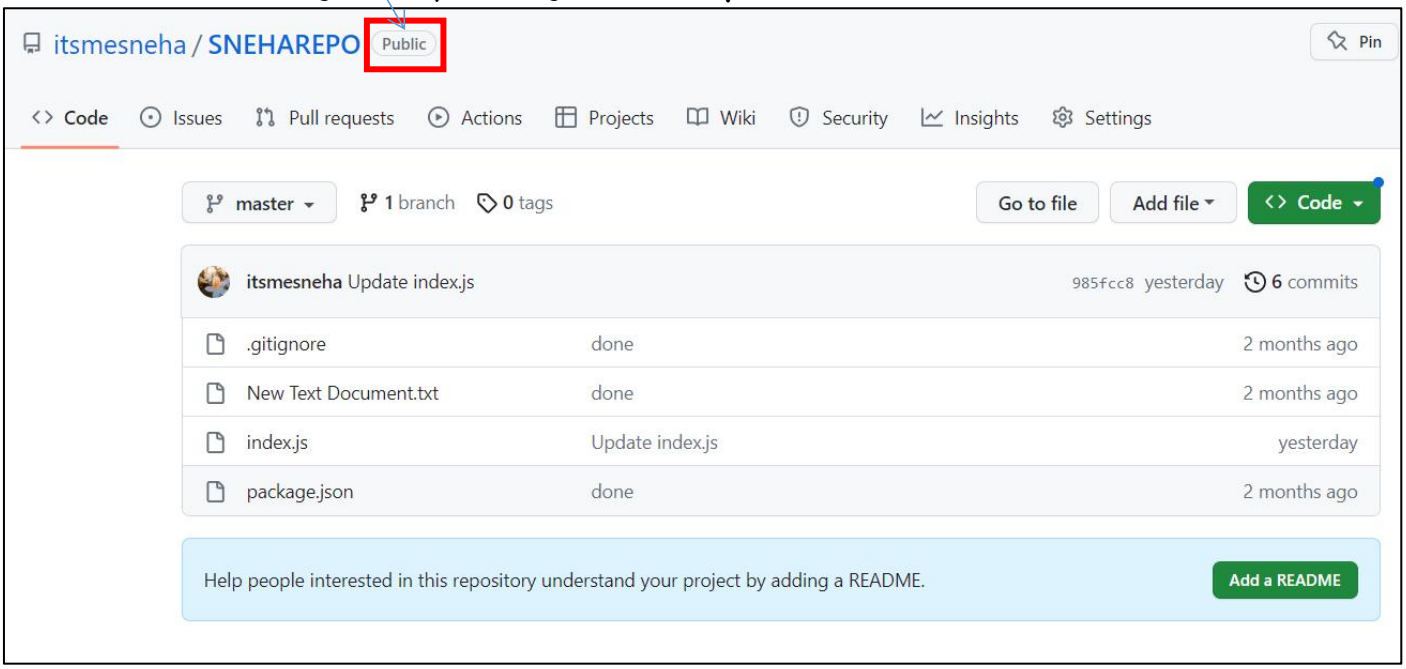
Assignment : 10

Title : Deploy a project from github to EC2 by creating new security group and user data.

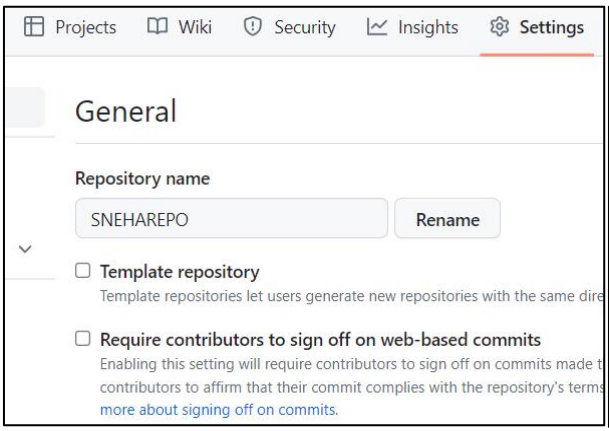
About :

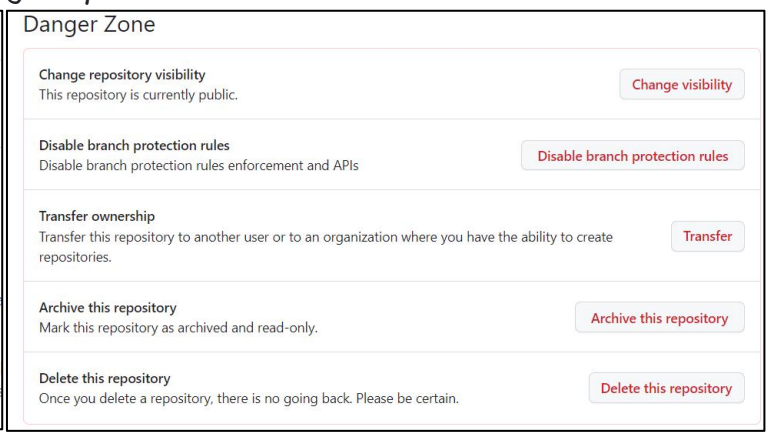
A security group acts as a virtual firewall to control inbound and outbound traffic.

- First open your github account , go to your repository that was already created in previous assignments - https://github.com/itsmesneha/SNEHAREPO
- Make sure that your repository is made '**public**'



- If the repository is not public, go to **Settings** -> scroll down to **Change repository visibility** and make change visibility to public.



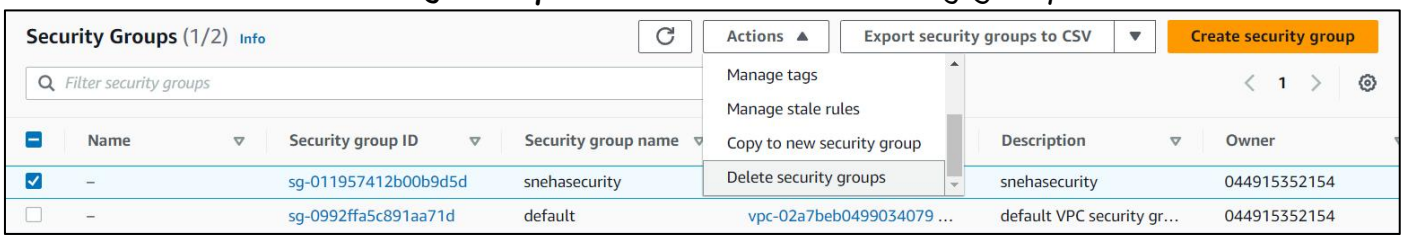


Steps to create a security group :

1. Go to EC2 Dashboard -> **Security Groups**
2. Delete all security groups (default cannot be deleted) -

Select the security group , go to **actions** - choose **Delete Security groups**

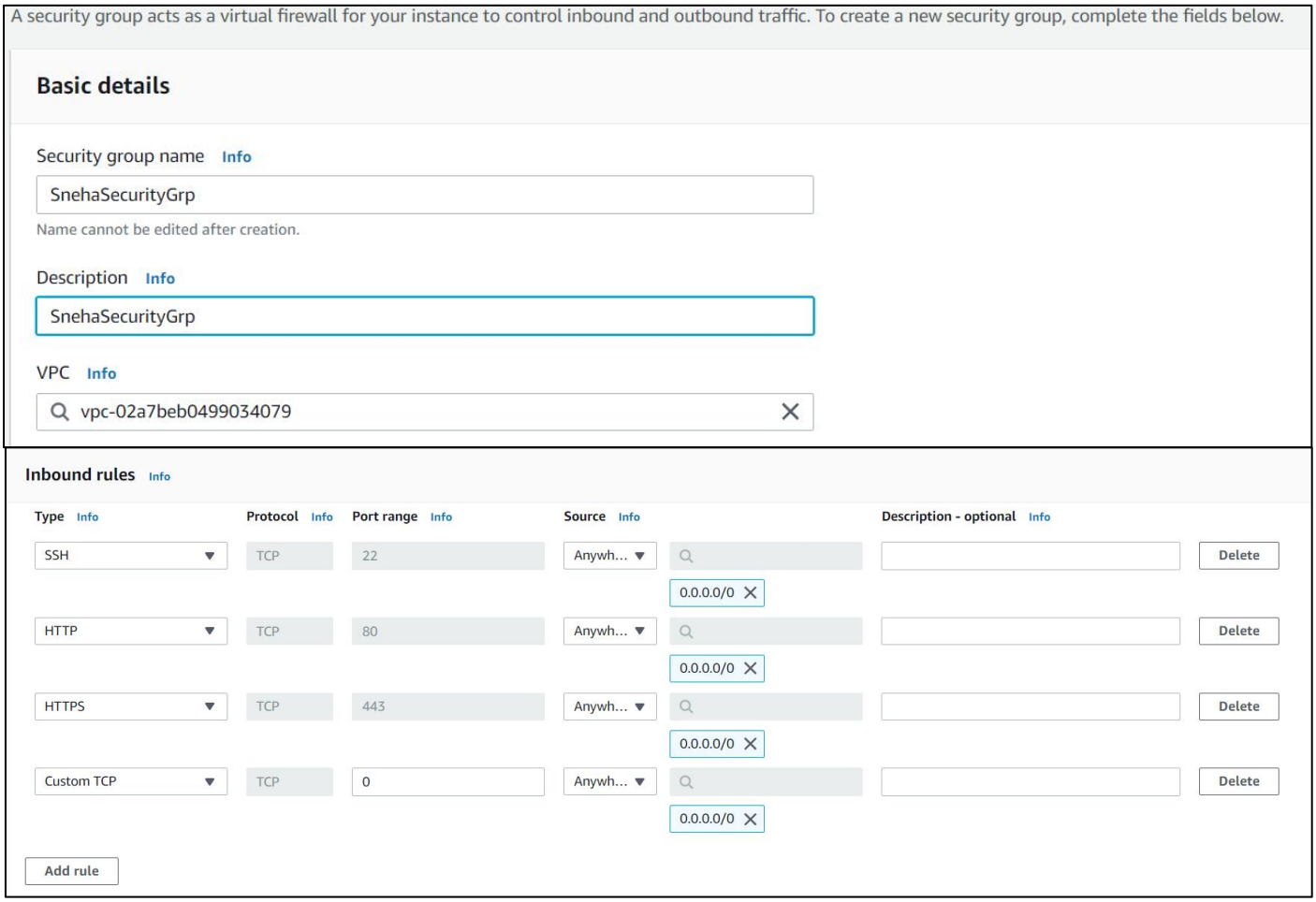
3. Click on **Create Security Group** to create a new security group.

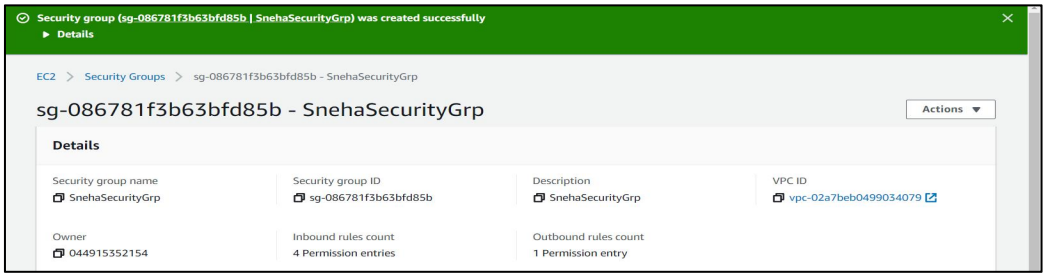


4. Now you need to give the **name (SnehaSecurityGroup) , description** details and change the **inbound rules** as follows :

Click on **Add rule** and then complete the following changes :

Type	Protocol	Port range	Source
SSH	TCP	22	Anywhere / 0.0.0.0
HTTP	TCP	80	Anywhere / 0.0.0.0
HTTPS	TCP	443	Anywhere / 0.0.0.0
Common TCP	TCP	0	Anywhere / 0.0.0.0

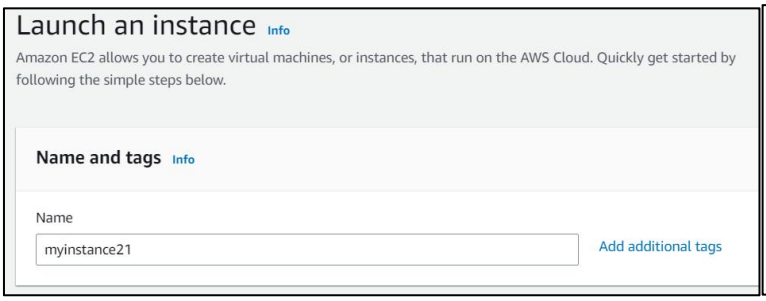


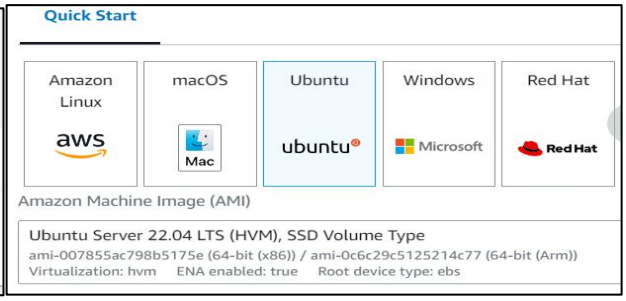


5.Click on create Security group and your security group is created.

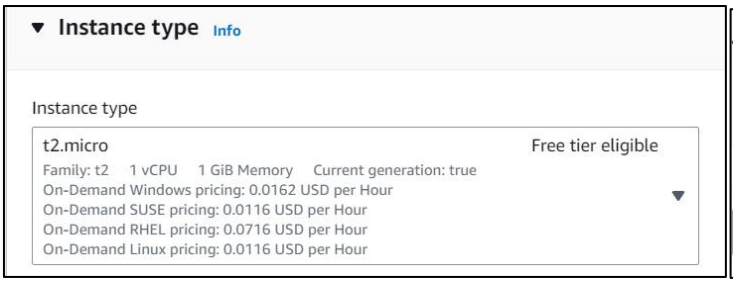
Steps to create an instance using security group :

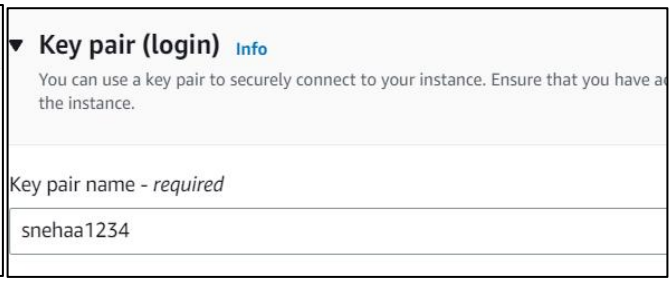
1. Click on **EC2 dashboard** and choose **Launch instance**
2. In the **launch an Instance** page give the name e.g. **myinstance21**
3. Under **Quick start** , choose **Ubuntu** (eligible for free tier).



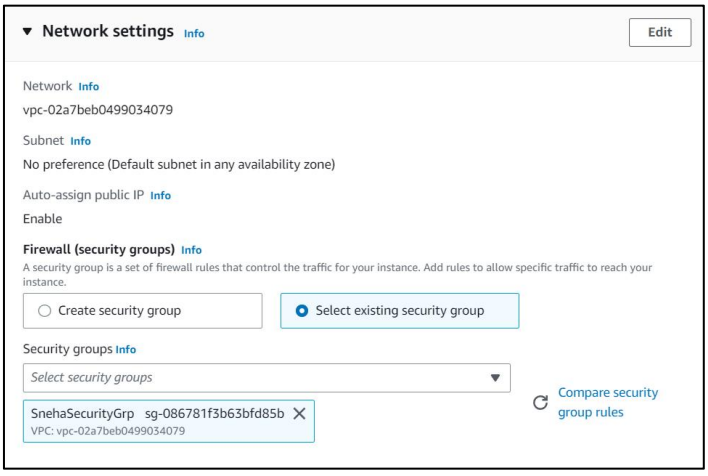


4. In **Instance type** select **t2 micro** and under **Key pair(login)** and select the key pair you have already created **(snehaa1234)** or **create a new key pair** by clicking on Create new key pair and create a new one.



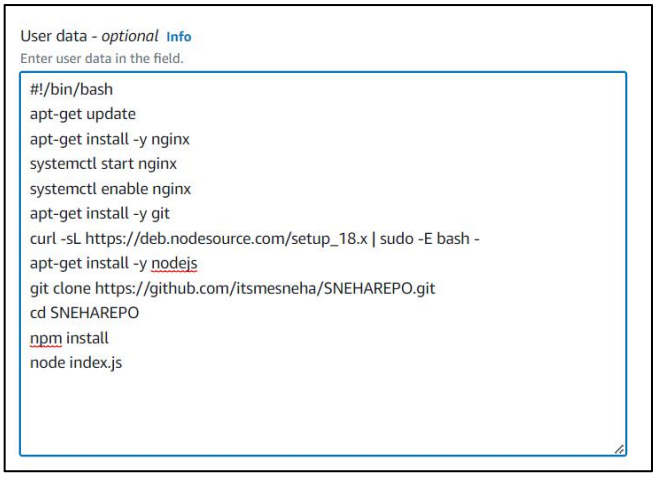


5. Now , under **Network settings** choose the Security group you previously created.



6. Under the user data , we need to copy paste the code given and change the git clone path to the one with our repository
And include the repository name beside cd command

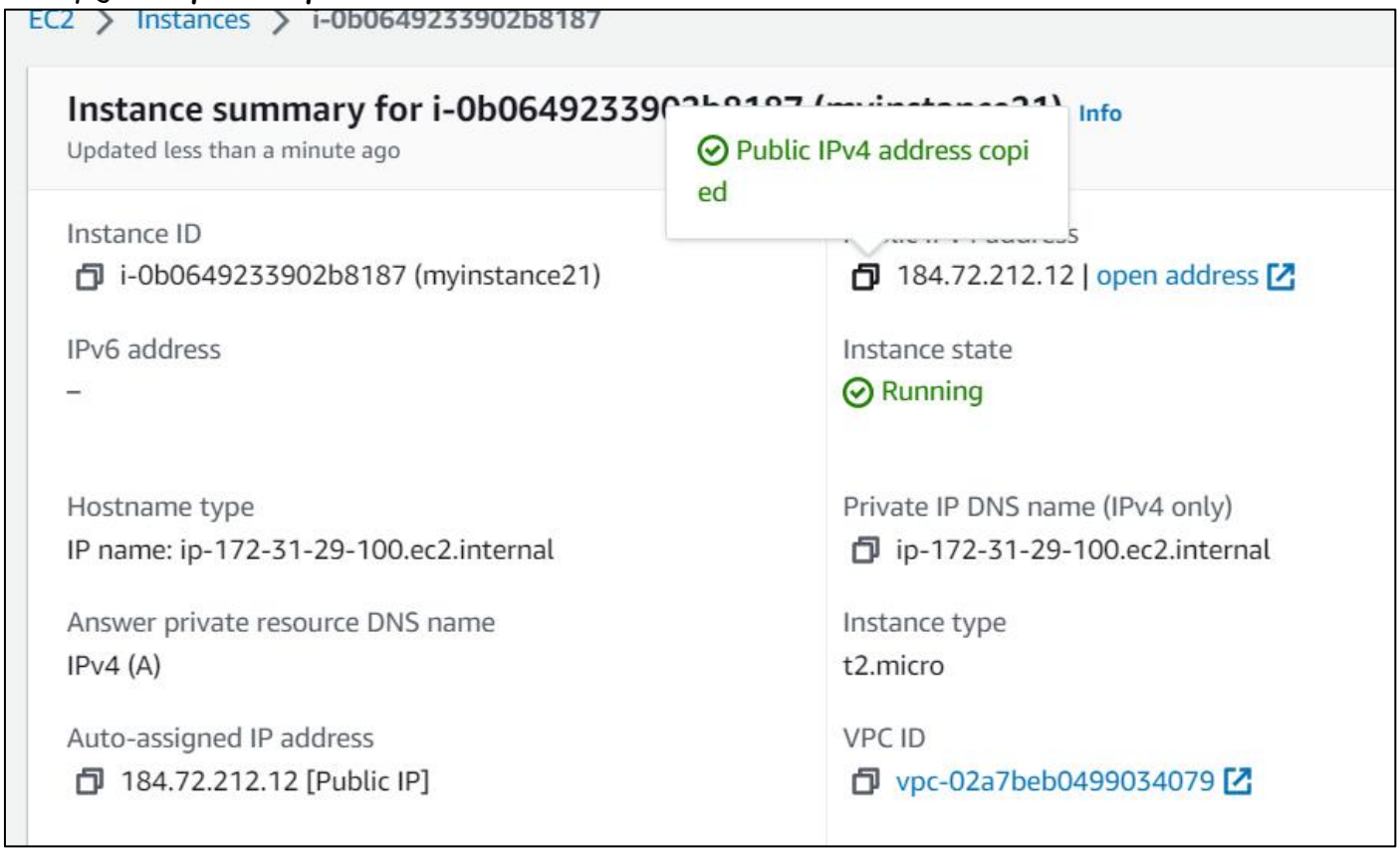
```
#!/bin/bash
apt-get update
apt-get install -y nginx
systemctl start nginx
systemctl enable nginx
apt-get install -y git
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
apt-get install -y nodejs
git clone https://github.com/.../...
cd ...
npm install
node index.js
```



Now the instance is created as seen below :

7.Click on the **instance id** which will lead to the **instance summary** of the instance

8.Copy the **public ipv4 address**



Paste the copied address in the browser along with the port number and see it runs.

Assignment : 11

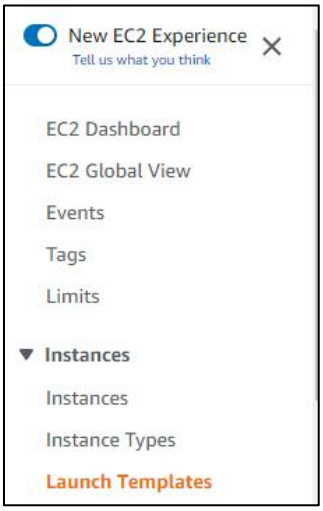
Title : **Build Scaling Plans in AWS that balance load on different EC2 instances.**

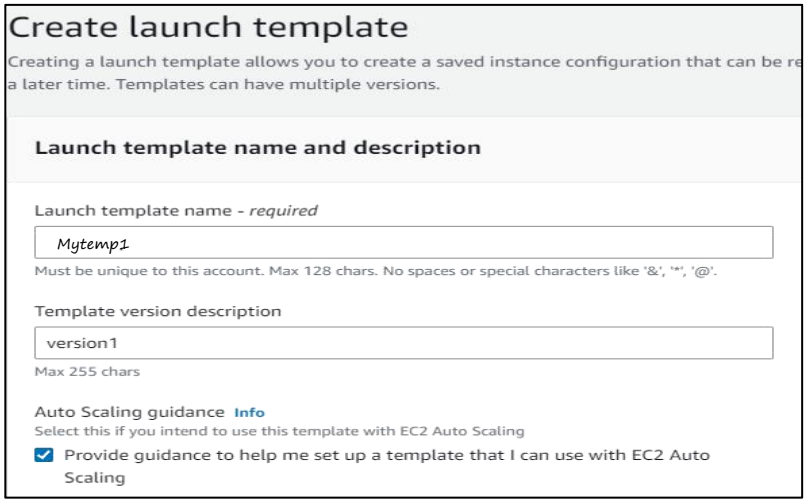
About :

Elastic Load Balancing automatically distributes your incoming application traffic across all the EC2 instances that you are running. Elastic Load Balancing helps to manage incoming requests by optimally routing traffic so that no one instance is overwhelmed.

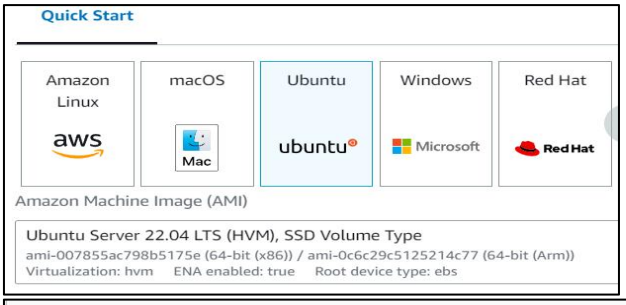
Steps to create Template :

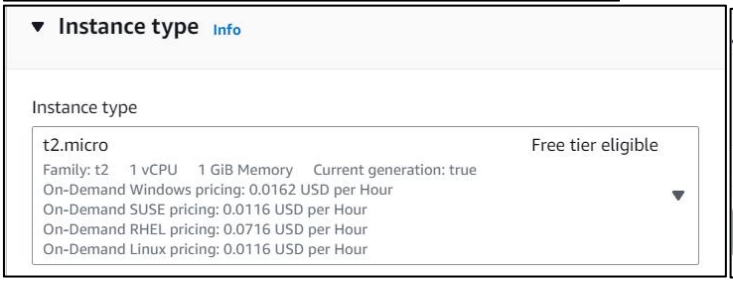
1. Go to EC2 dashboard and in the left side select "**Launch Templates**".
2. Click **New launch template** . Give template name, template version, check auto scaling guidance box.

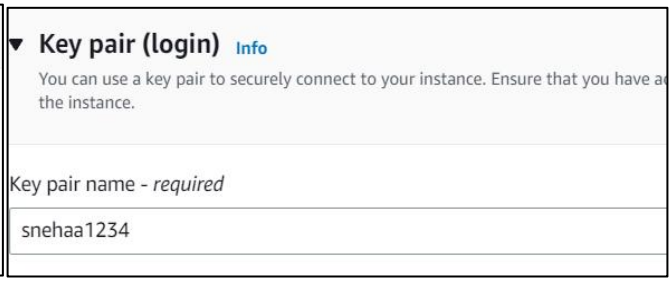




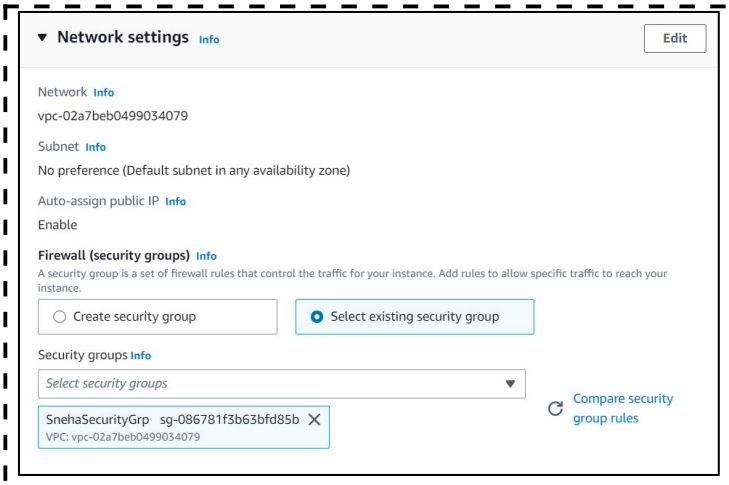
3. In hardware select **ubuntu**, instance type **t2.micro** ,give key pair name(in case if you have existing key pair give that otherwise create new one).





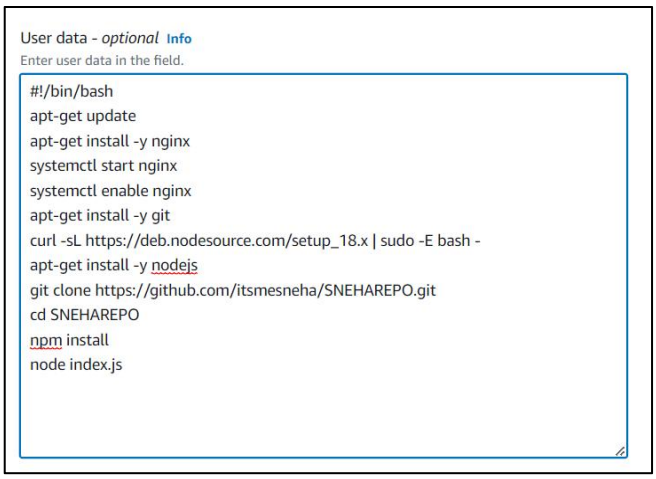


4. Now , under **Network settings** choose the Security group you previously created.



5.Under the user data , we need to copy paste the code given and change the git clone path to the one with our repository
And include the repository name beside cd command

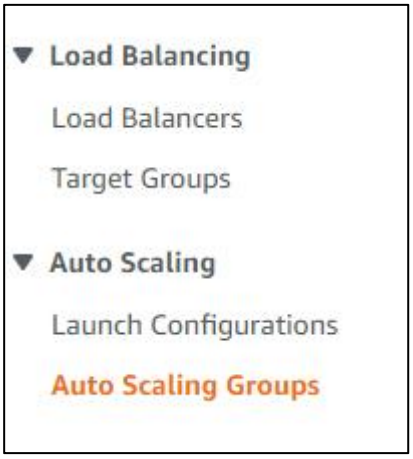
```
#!/bin/bash
apt-get update
apt-get install -y nginx
systemctl start nginx
systemctl enable nginx
apt-get install -y git
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
apt-get install -y nodejs
git clone https://github.com/.../...
cd ...
npm install
node index.js
```

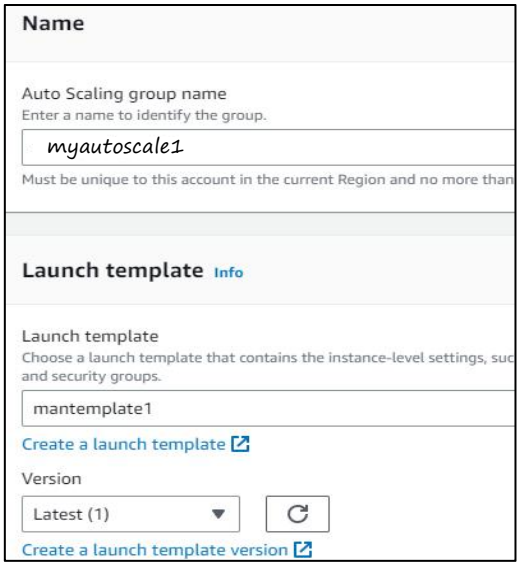


Note- before copying the github repo make sure it is public. If not then perform following steps- Go to repo settings and by scrolling down in danger zone click "**change repository visibility**". And change it to public.

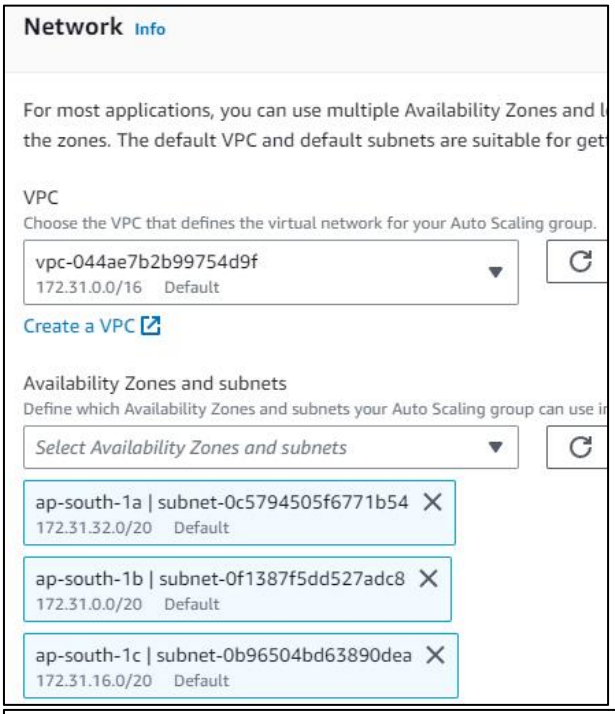
Steps to create Autoscaling groups:

1.In EC2 dashboard click Auto Scaling Groups. Click on **Create Auto Scaling group.**
2. Give auto scaling group name (**ex-myautoscale1**). In **launch template** click on the existing template(ex-**mytemplate1**) ,give version **Latest(1)** and click next

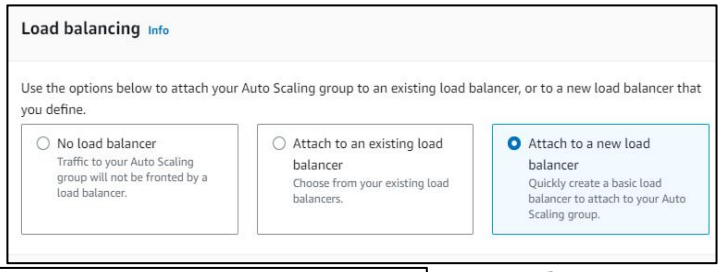


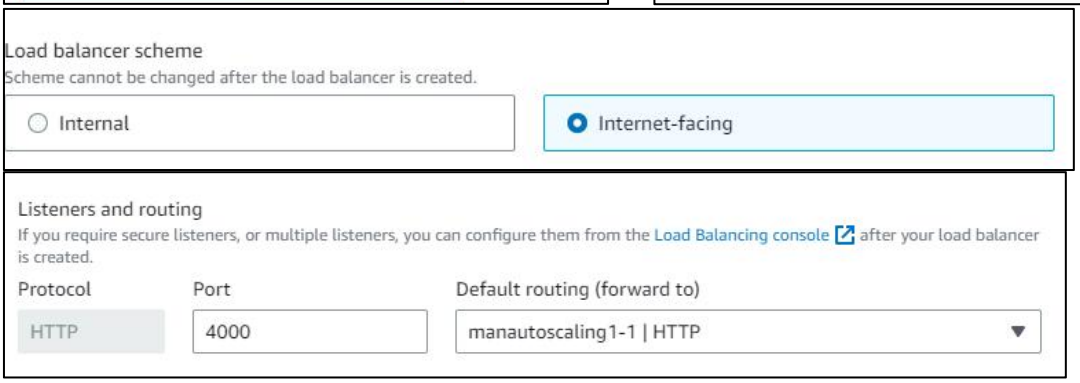


3.In **Network ,Availability Zones and subnets** click all the zones and click next.



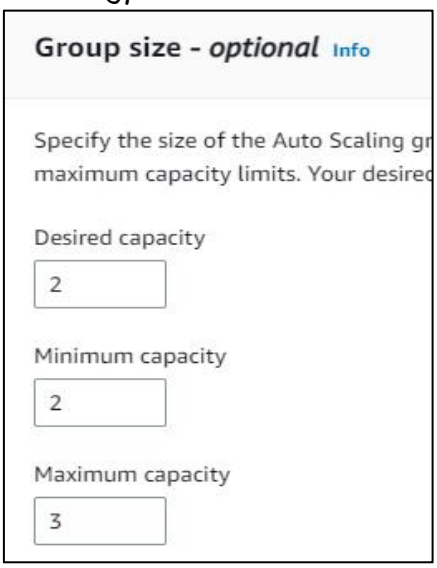
8.In load balancing click "**Attach to a new load balancer**", in load balancer scheme select "**internet-facing**" , in **listeners and routing** give port number 4000 and default routing select autoscaling group(ex-manautoscaling1-1|HTTP). And click next.

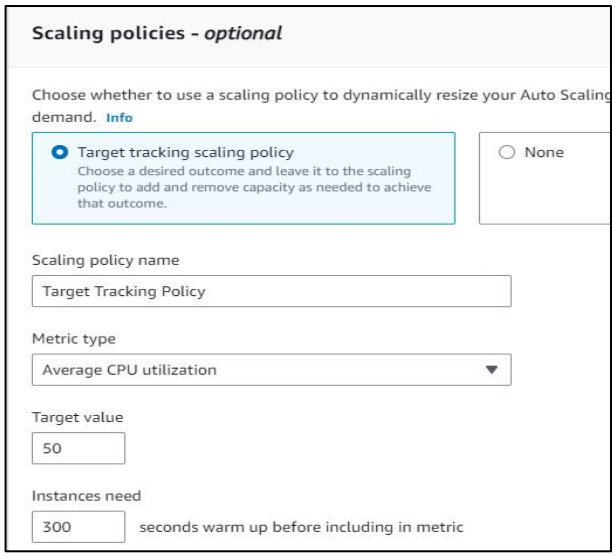




9.in Group size give desired capacity 2,minimum capacity 2,maximum capacity 3.

10.In Scaling policies click " Target tracking scaling policy" and instances need section type 300.





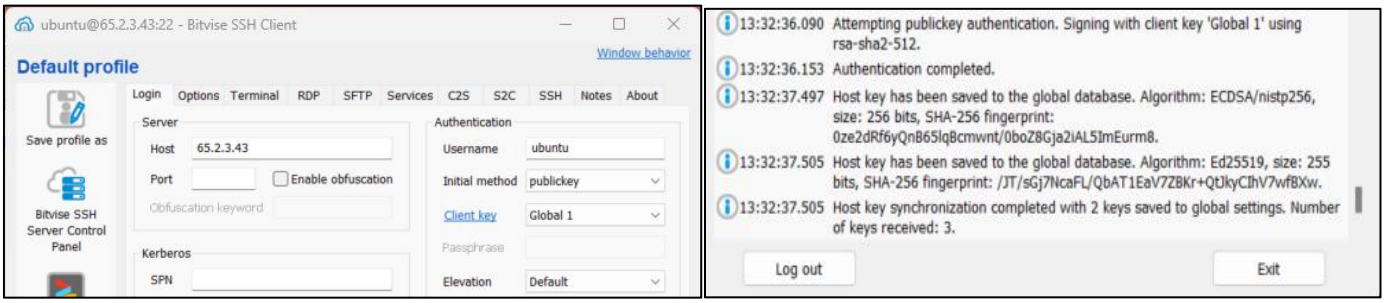
11. **Click next** and click **Create Auto Scaling group**. And it will be created.

Now, we have to crash these two running servers. for that we will crash one server with bitvise ssh client and in another one we will crash through directly opening terminal.

For one server:

1.Copy **public IPv4 address**(ex-65.2.3.43) and paste it on **Bitvise SSH Client**.

2.Give username **ubuntu**, initial method **publickey,** in client key manager import that same existed key pair .pem file(**ex-snehaa1234.pem**) and click **Global1** in Client key.and click log in. We are already logged in now .

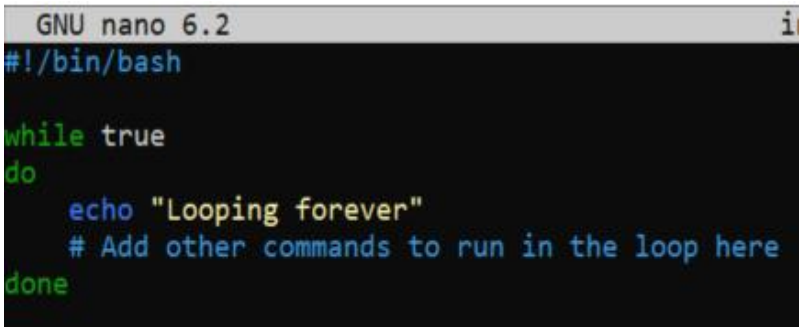


3.In Terminal type **nano infi1.sh** and in the file write the following lines of code and save it. To execute the file give command **chmod +x infi1.sh**.

4. To run give command **./infi1.sh** and infinite loop will start.

The code :

```
#!/bin/bash
While true
do
   echo "Looping forever"
   # Add another commands to run in the loop here
done
```



For another server:

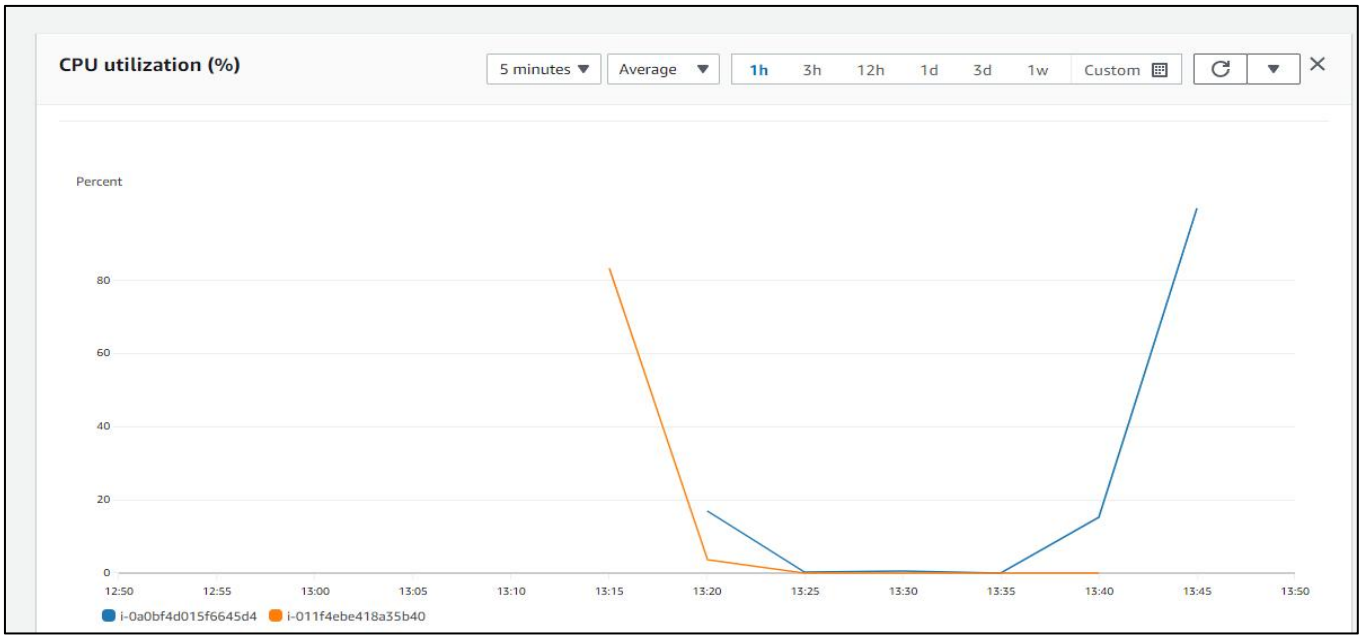
1. Click on **connect** option and one terminal will open.

2. In the terminal type the command as same as previous. And run .

```
To run a command as administrator (user "root"),
See "man sudo_root" for details.

ubuntu@ip-172-31-35-242:~$ nano infi2.sh
ubuntu@ip-172-31-35-242:~$ chmod +x infi2.sh
ubuntu@ip-172-31-35-242:~$ ./infi2.sh
```

Now, the servers will be overloaded and we can see that by click on CPU utilization.



After some time,we can see that new instance is created automatically for load balancing.