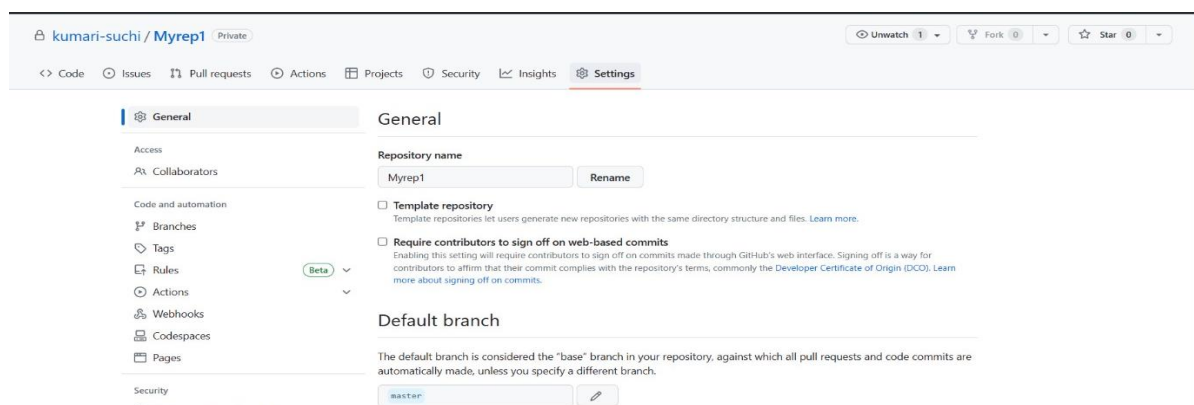
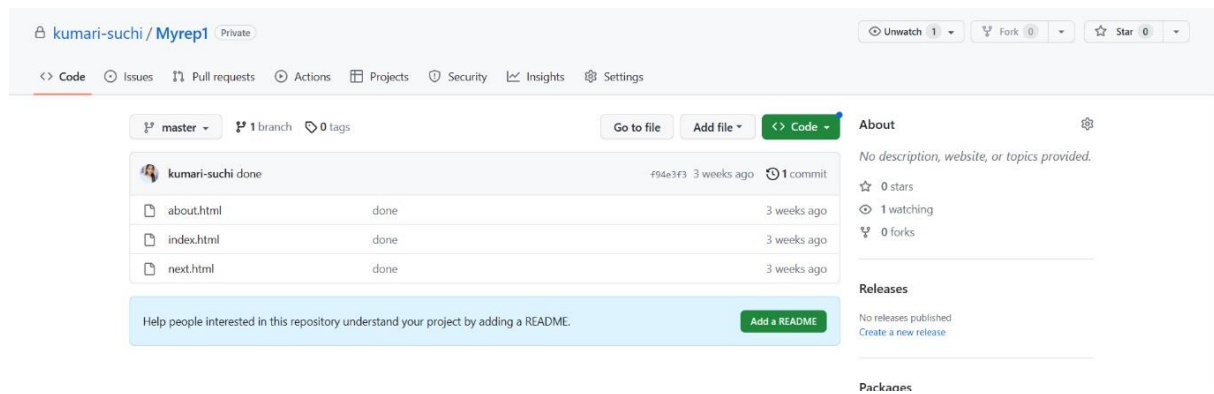


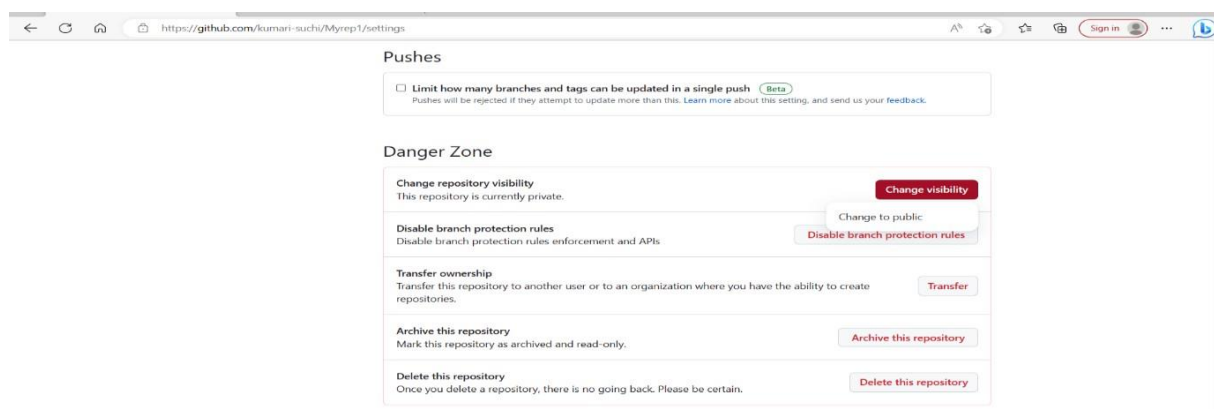
AWS-11

Build scaling plans in AWS that balance load on different EC2 instances.

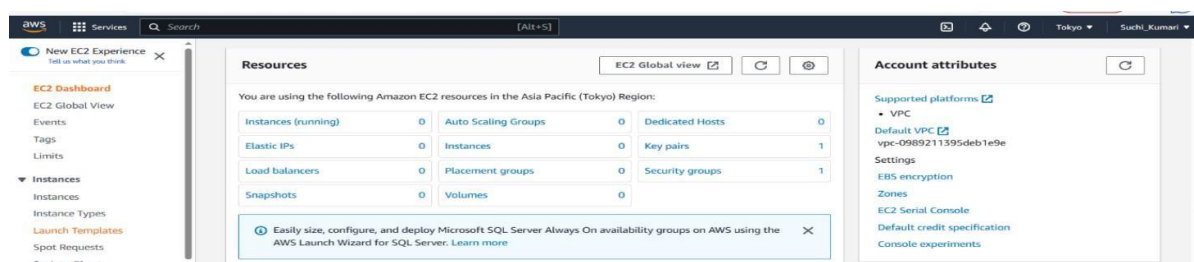
1. Sign in to your GitHub account.
2. Then go to your repository . Make sure the Repository which will be cloned is made public or not.
3. For public repository first go to settings.



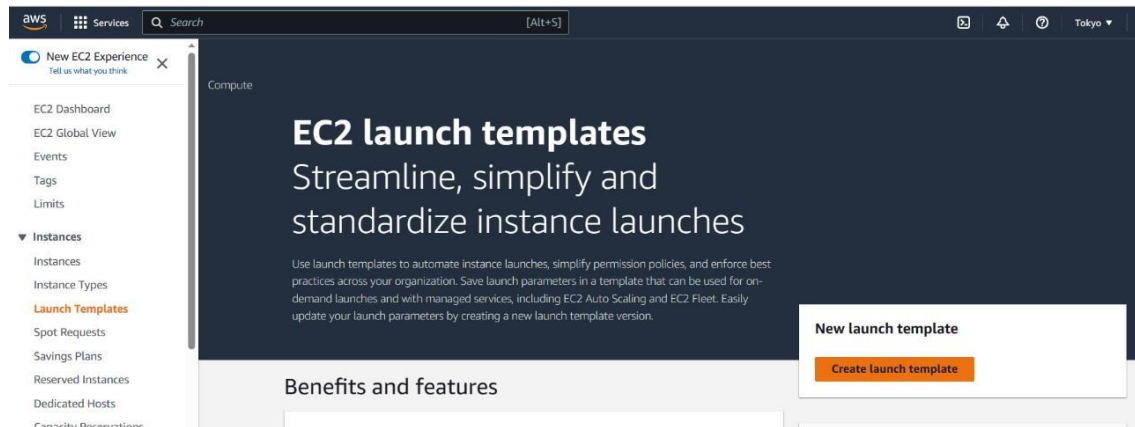
4. Then scroll down and go to Danger Zone then click on change visibility and change to public.



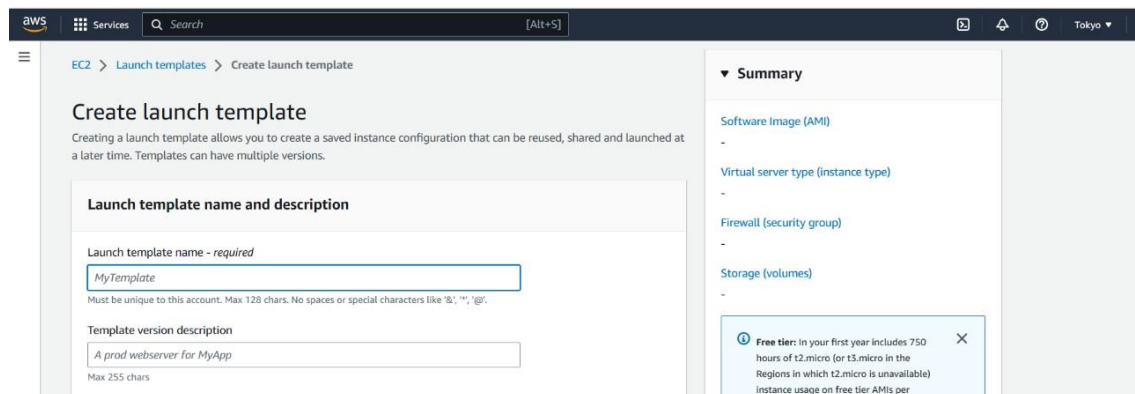
5. Then sign in to aws account then click EC2. Then go to EC2 Dashboard and then click on Launch Templates.



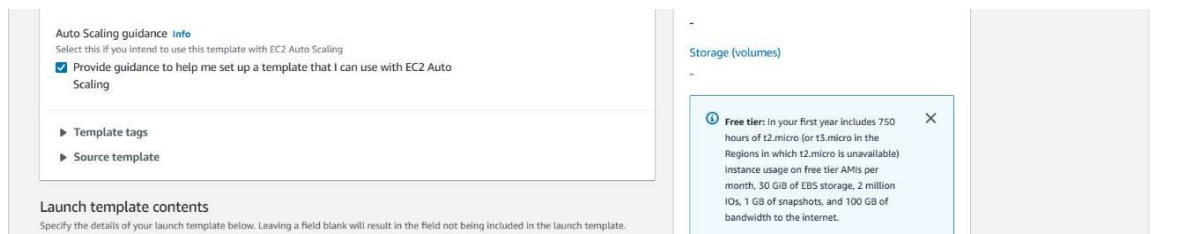
6. Then click on Create Launch template.



7. Then enter Launch template name and Template version description.



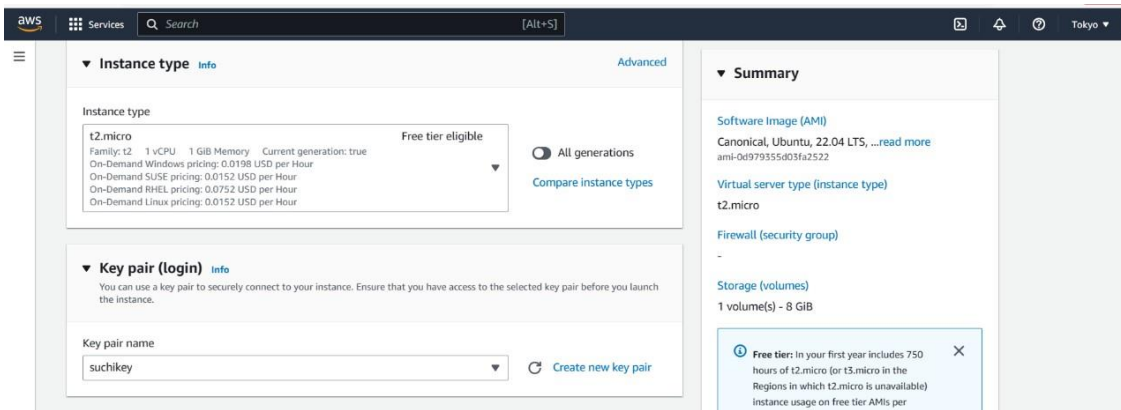
8. Then Check the “Provide Guidance” box.



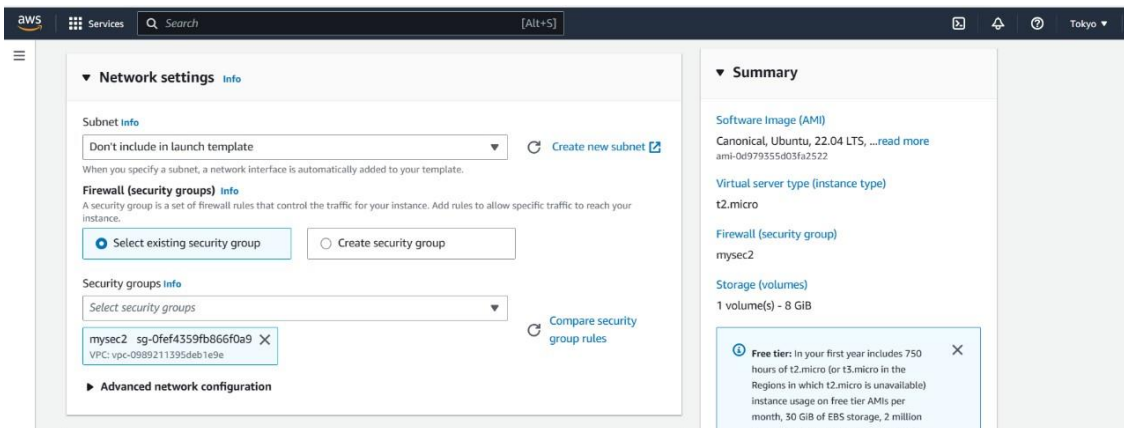
9. Then click on Ubuntu.



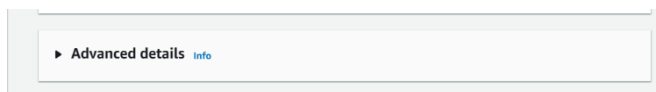
10. Under Instance type select t2.micro type of configuration. Then enter key pair name.



11. Then in Network settings select existing security group then select Security groups.



12. Then click on Advanced details.



13. scroll down and then in User data write some 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 YourRepositoryURLhere

cd YourRepositoryNamehere/

npm install

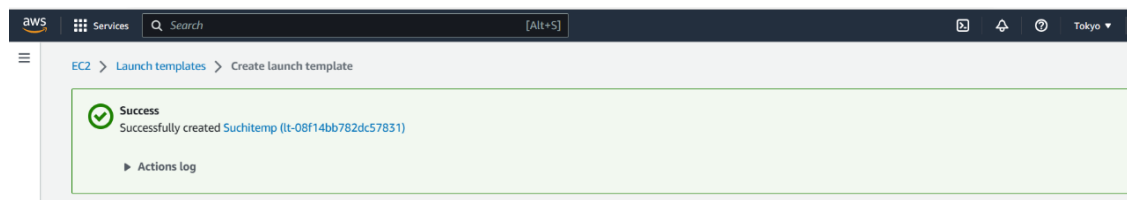
node index.js
```

Then click on Launch Instance.

```
User data - optional Info
Enter user data in the field.

#!/bin/bash
apt-get update
apt-get install -y nginx
systemctl start nginx
systemctl enable nginx
apt-get install -y
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
apt-get install -y nodejs
git clone https://github.com/kumari-suchi/Myrep1.git
cd Myrep1
npm install
node index.js
```

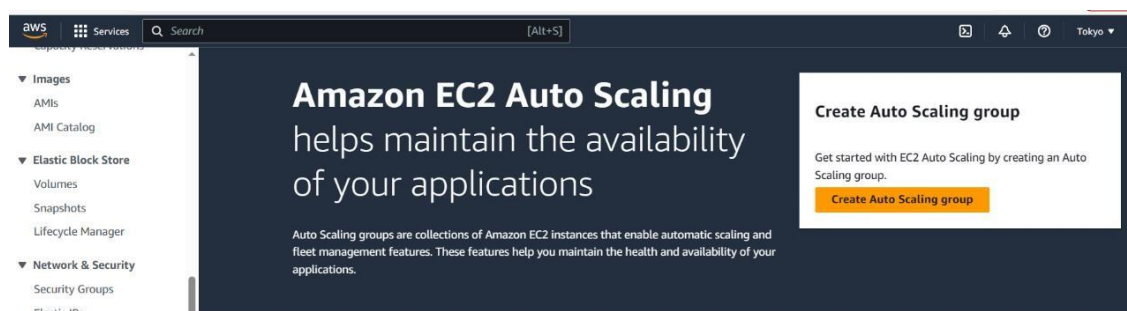
14. Then successfully created Launch templates.



15. Then again go to EC2 Dashboard the click on Auto Scaling and then Auto Scaling Groups.



16. Then click on Create Auto Scaling group.



17. Then enter Auto Scaling group name.

Step 2: Choose instance launch options

Step 3 - optional: Configure advanced options

Step 4 - optional: Configure group size and scaling policies

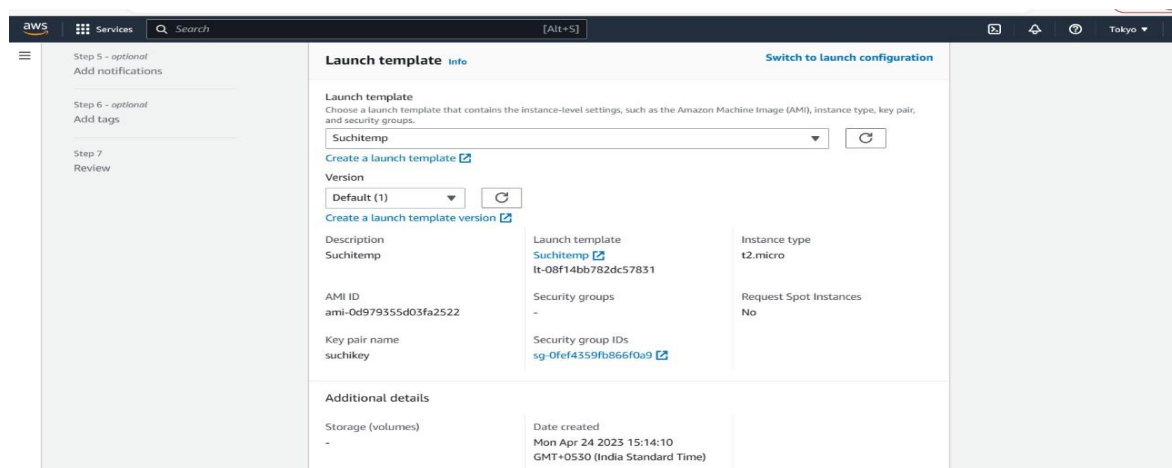
Name

Auto Scaling group name
Enter a name to identify the group.

SuchiAutoScaling

Must be unique to this account in the current Region and no more than 255 characters.

18. Then select Launch template which is you have created then in version select Default(1).Then click on next.



19. After that, Under Availability Zones and Subnets select all the zones that appear.

Step 1
Choose launch template or configuration

Step 2
Choose instance launch options

Step 3 - optional
Configure advanced options

Step 4 - optional
Configure group size and scaling policies

Step 5 - optional
Add notifications

Step 6 - optional
Add tags

Step 7
Review

Choose instance launch options [info](#)

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

Network [info](#)

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.

☒ ap-northeast-1a | subnet-099f4273f89103a9b
172.31.32.0/20 Default

☒ ap-northeast-1c | subnet-07ecbdc2014f714e
172.31.0.0/20 Default

☒ ap-northeast-1d | subnet-0f39630393b0f2919
172.31.16.0/20 Default

Select Availability Zones and subnets

ap-northeast-1a | subnet-099f4273f89103a9b X

ap-northeast-1c | subnet-07ecbdc2014f714e X

ap-northeast-1d | subnet-0f39630393b0f2919 X

Create a subnet [link](#)

20. Then click on next.

Step 5 - optional
Add notifications

Step 6 - optional
Add tags

Step 7
Review

Choose instance launch options [info](#)

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

ap-northeast-1a | subnet-099f4273f89103a9b X

ap-northeast-1c | subnet-07ecbdc2014f714e X

ap-northeast-1d | subnet-0f39630393b0f2919 X

Create a subnet [link](#)

Instance type requirements [info](#)

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template
[Suchitemp](#)
lt-08f14bb782dc57831

Version
Latest

Description
Suchitemp

Instance type
t2.micro

Cancel Skip to review Previous **Next**

21. Now Under Load Balancers select the Attach to a New Load balancer option.

Load balancing [info](#)

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

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

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

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

22. Now select Internet-Facing under Load balancer scheme.

Attach to a new load balancer

Define a new load balancer to create for attachment to this Auto Scaling group.

Load balancer type
Choose from the load balancer types offered below. Type selection cannot be changed after the load balancer is created. If you need a different type of load balancer than those offered here, visit the [Load Balancing console](#).

☒ Application Load Balancer
HTTP, HTTPS

☐ Network Load Balancer
TCP, UDP, TLS

Load balancer name
Name cannot be changed after the load balancer is created.
SuchiAutoScaling-1

Load balancer scheme
Scheme cannot be changed after the load balancer is created.
☐ Internal ☒ Internet-facing

23. Under Listeners and Routing enter the port no. of the project and select Create target group followed by giving the target group a name

The screenshot shows the 'Listeners and routing' configuration page in the AWS console. It includes a 'Protocol' dropdown set to 'HTTP', a 'Port' input field with '4000', and a 'Default routing (forward to)' dropdown set to 'Create a target group'. Below this is a 'New target group name' input field containing 'SuchiAutoScaling-1'. There is also a 'Tags - optional' section with an 'Add tag' button and a note that 50 tags are remaining.

24. Now click on the next button. After clicking on the Next button, a new page will open. Under Group Size mention:

- Desired Capacity = 2
- Minimum Capacity = 2
- Maximum Capacity = 3

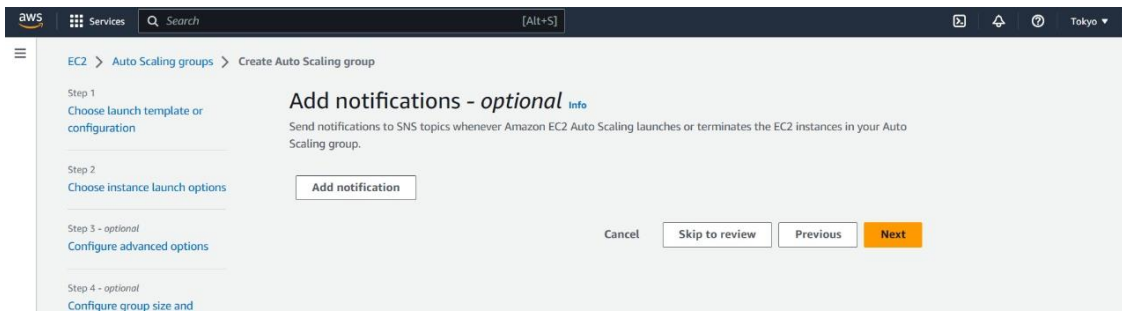
The screenshot shows the 'Configure group size and scaling policies' page in the AWS console. It includes a sidebar with steps 1 through 7. The main content area is titled 'Group size - optional' and contains three input fields: 'Desired capacity' set to '2', 'Minimum capacity' set to '2', and 'Maximum capacity' set to '3'.

25. Now under Scaling policies Choose the Target Tracking Scaling policy option.

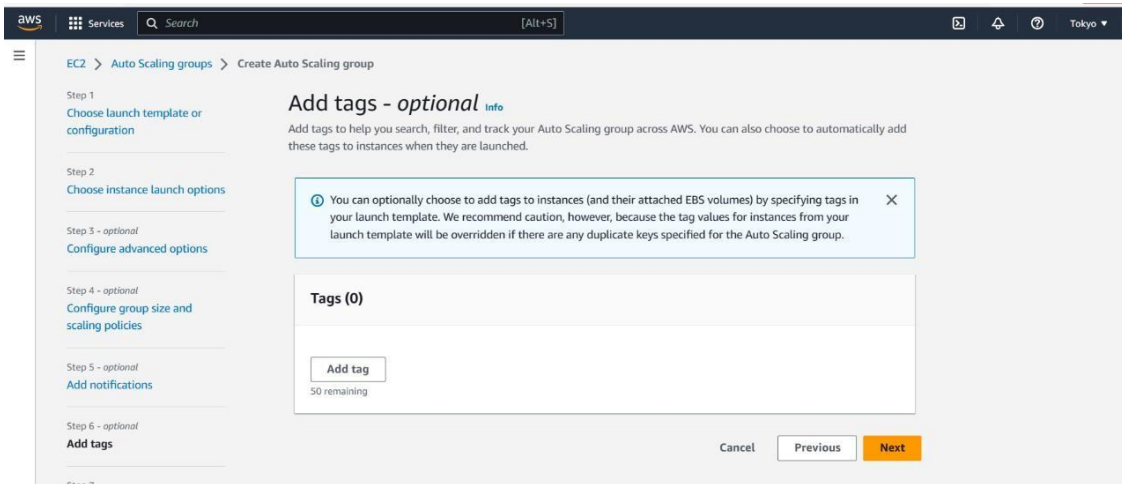
- Select the metric type as Average CPU utilization.
- Set Target Value to 50.
- Set Warm-Up time to 300 seconds under Instances Need then click on next.

The screenshot shows the 'Scaling policies' page in the AWS console. It includes a sidebar with steps 1 through 7. The main content area is titled 'Scaling policies - optional' and contains a radio button selection for 'Target tracking scaling policy'. Below this are input fields for 'Scaling policy name' (set to 'Target Tracking Policy'), 'Metric type' (set to 'Average CPU utilization'), 'Target value' (set to '50'), and 'Instances need' (set to '300' seconds warm up before including in metric). There is also a checkbox for 'Disable scale in to create only a scale-out policy'.

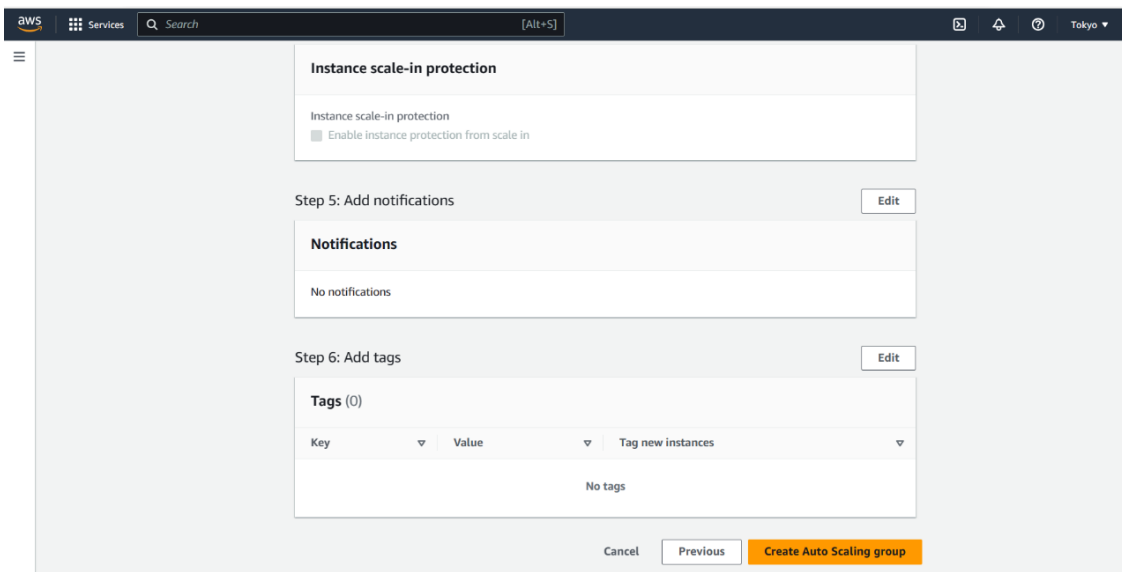
26. Then click on next.



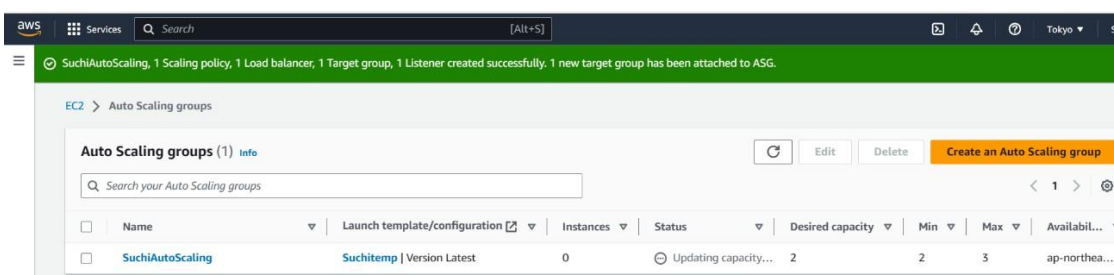
27. Then again click on next.



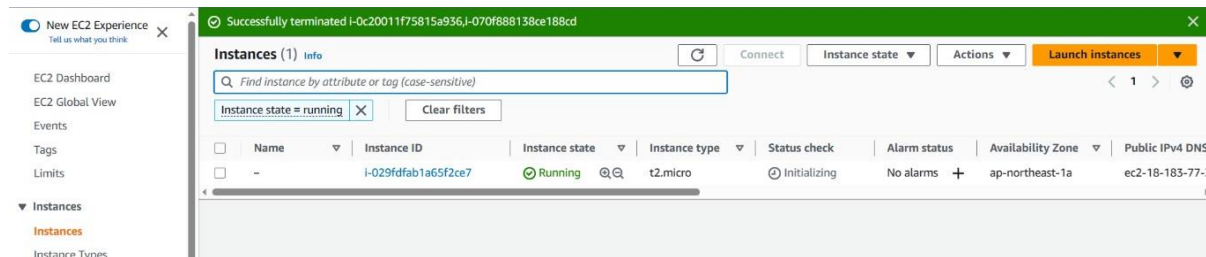
28. Then click on Create Auto Scaling group.



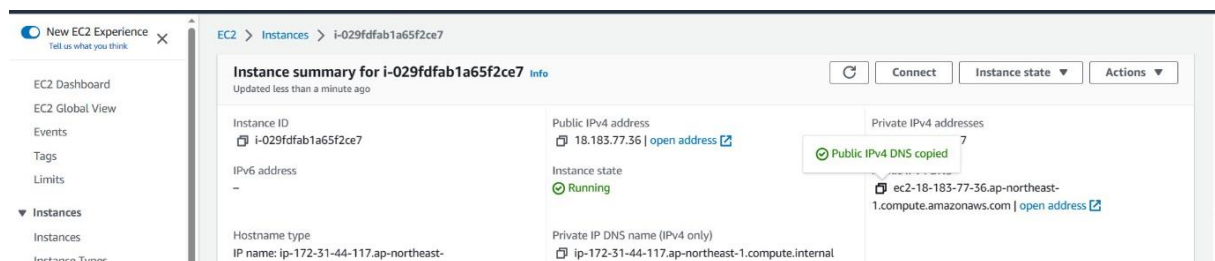
29. Then successfully Auto Scaling group created.



30. Return to the Instances Page using the Left side Nav bar. After some you will notice a new instance server will appear automatically! To help finding it more easily we need to activate the instance running filter. Click on the search box below the Instances Section Heading. Start typing running. Select the option “Instance state = running” option in the suggestion dropdown. The filter will be activated. After some few seconds of refreshing we will be able to see two new servers are running. Then click on Instance ID.



31. Then Copy its public IPV4 DNS.



32. Paste it in another browser.



33. Now to access our project webpage we need to append the port no. (4000) of our project with a “:”

Hello,I am Suchi Kumari

34. We will now OVERLOAD THE SERVER INSTANCES by running scripts and increasing CPU utilization value above the threshold that we specified during Configuration of the Auto scaling group.

35. For it we will use:

- Use Bitwise SSH client for instance 1.
- Use direct connect terminal in AWS for instance 2

36. For Instance-1:

- Copy the public IPv4 address

- Open Bitwise SSH client.
- Paste the IP and select/specify the necessary options.
- Now Log-In to your server.
- Open the new Terminal.
- Now enter the command:
 - nano su1.sh
- After the command a new nano Editor window will open. Type the following in it.

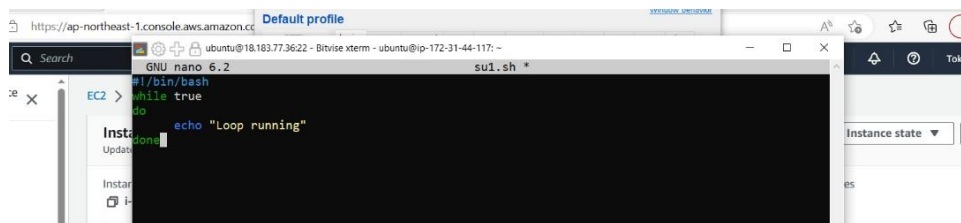
```
#!/bin/bash
```

```
while true
```

```
do
```

```
    echo "Loop running"
```

```
done
```



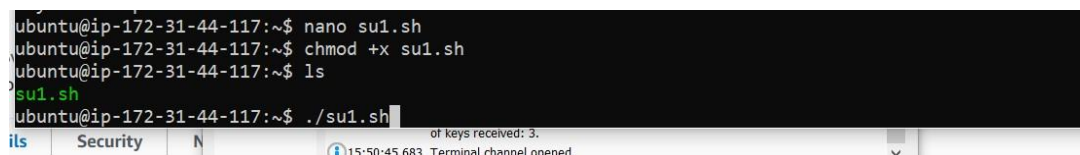
- Now, to save and close the shell script we need to press the following shortcuts and keys sequentially:

Ctrl+X

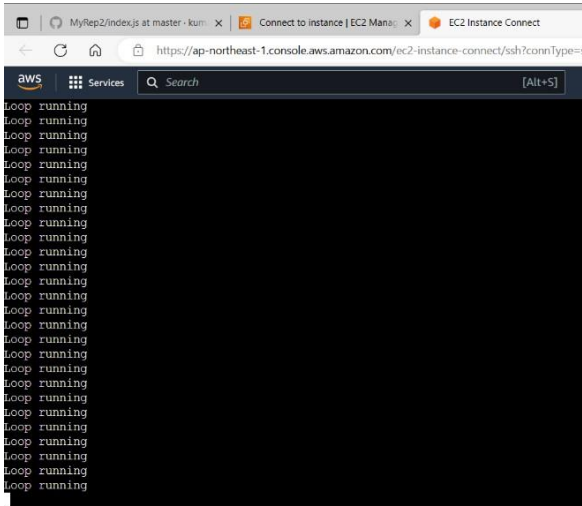
Y

Enter

- Now you will be returned back to the terminal.
- Now type the following commands:
 - chmod +x su1.sh
 - ./su1.sh (Used to execute the su1.sh script).



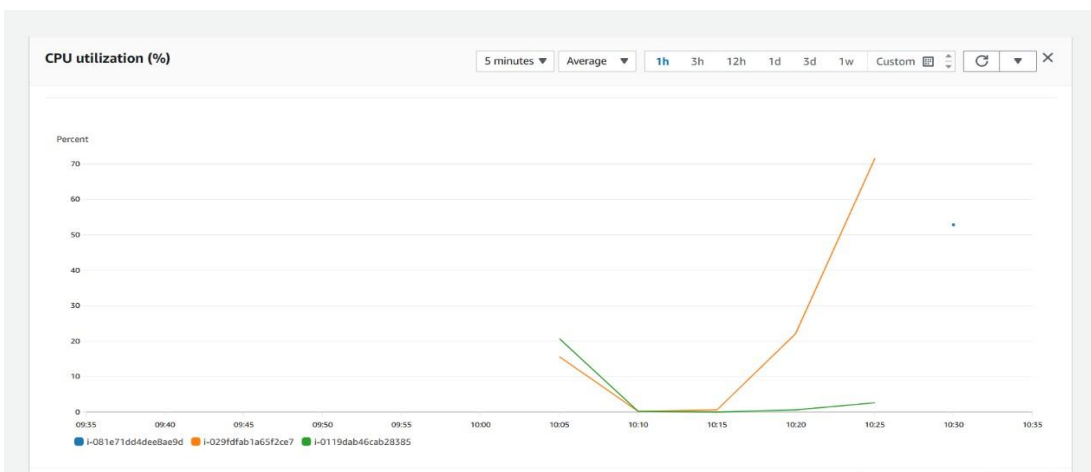
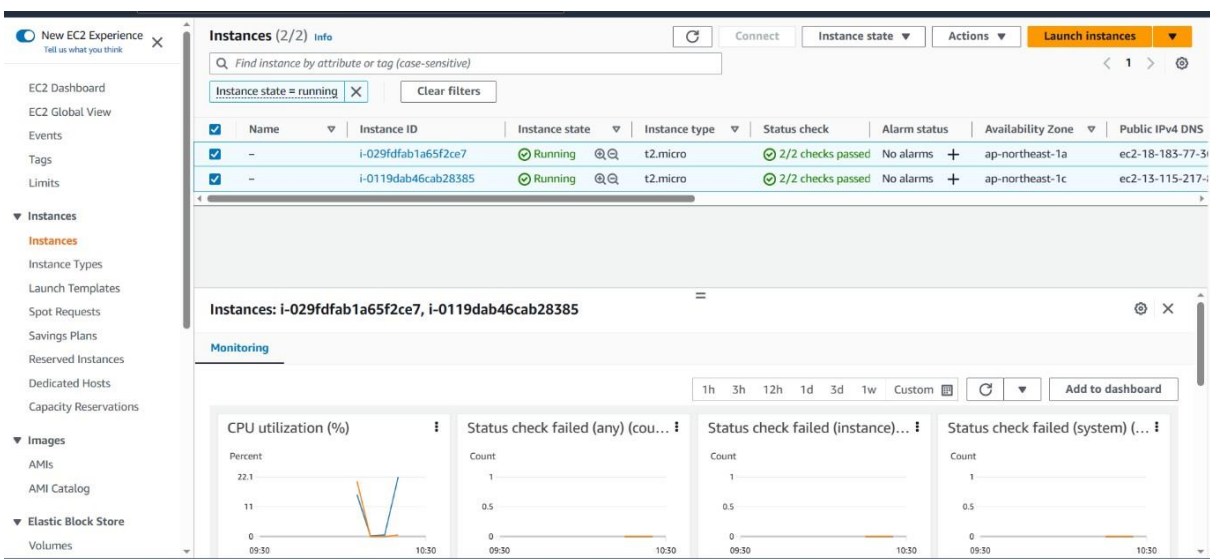
- Now the script will start running infinitely!
- Do not close the terminal. Keep it minimized.



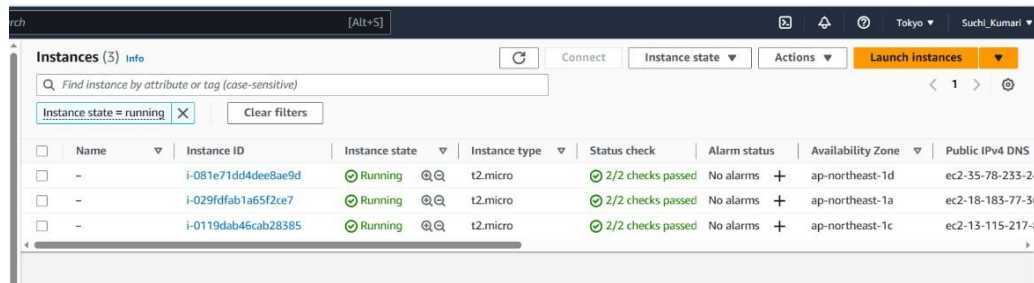
38. Now go to the instances page. Select both the instances.

Click on the instances white bar at the bottom of the page. Now drag the two bars to expand the view. We are interested only in the CPU utilization graph. Click on the maximize icon by hovering over the graph as shown in the fig to maximize this graph.

Our 1st instance has already crossed over 50% utilization. That's why we can see already a new third instance has been initiated by our auto-scaling group to compensate for the overload.

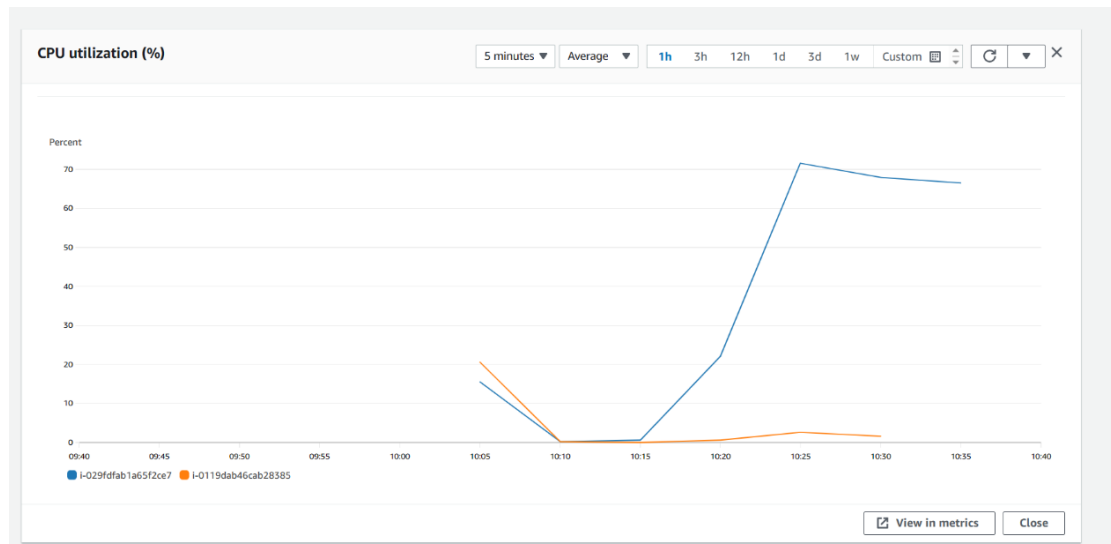


39. There can only be 3 servers running at a time for us as specified in our Auto Scaling group when we were creating it. Hence, we have reached our maximum limit of instances running concurrently.



The screenshot shows the AWS Management Console 'Instances' page. It displays a table with 3 instances, all in a 'Running' state. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. The instances are t2.micro type and are distributed across three availability zones: ap-northeast-1d, ap-northeast-1a, and ap-northeast-1c.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-081e71dd4dee8ae9d	Running	t2.micro	2/2 checks passed	No alarms	ap-northeast-1d	ec2-35-78-233-2...
-	i-029fdfab1a65f2ce7	Running	t2.micro	2/2 checks passed	No alarms	ap-northeast-1a	ec2-18-183-77-3...
-	i-0119dab46cab28385	Running	t2.micro	2/2 checks passed	No alarms	ap-northeast-1c	ec2-13-115-217-...



Hence, our Auto-Scaling Group can handle instance overloading by providing new instances to handle the overloading.

Now observe that whenever we close or terminate any instance then a new instance gets created. Hence, we cannot delete them if we want to delete them finally.

Then first delete Auto Scaling groups

Then delete Load balancer

Then delete Target group

Then You will find that all the instances created by the Auto-Scaling group will automatically be terminated.