

In this hands-on will check how Auto scaling group works

For this we require-

Launch configuration

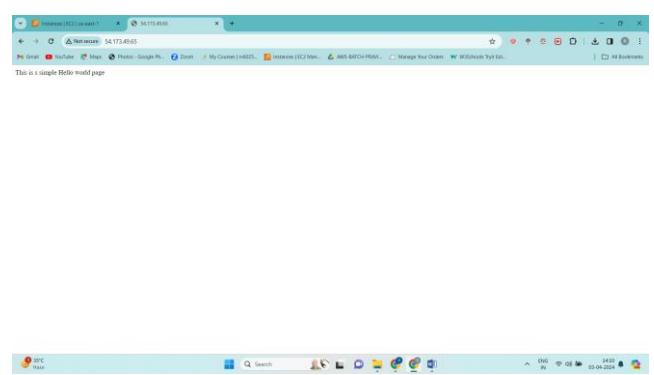
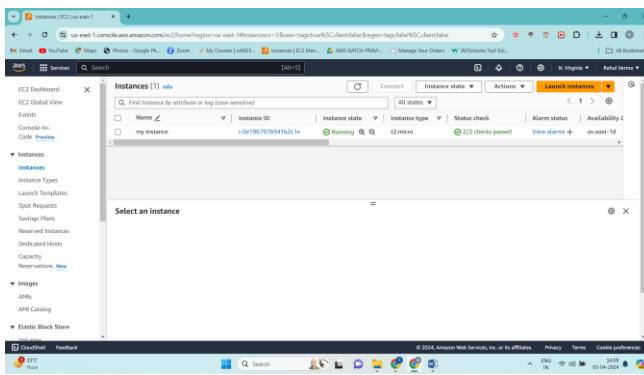
Target group &

Load balancer & cloud watch alarm

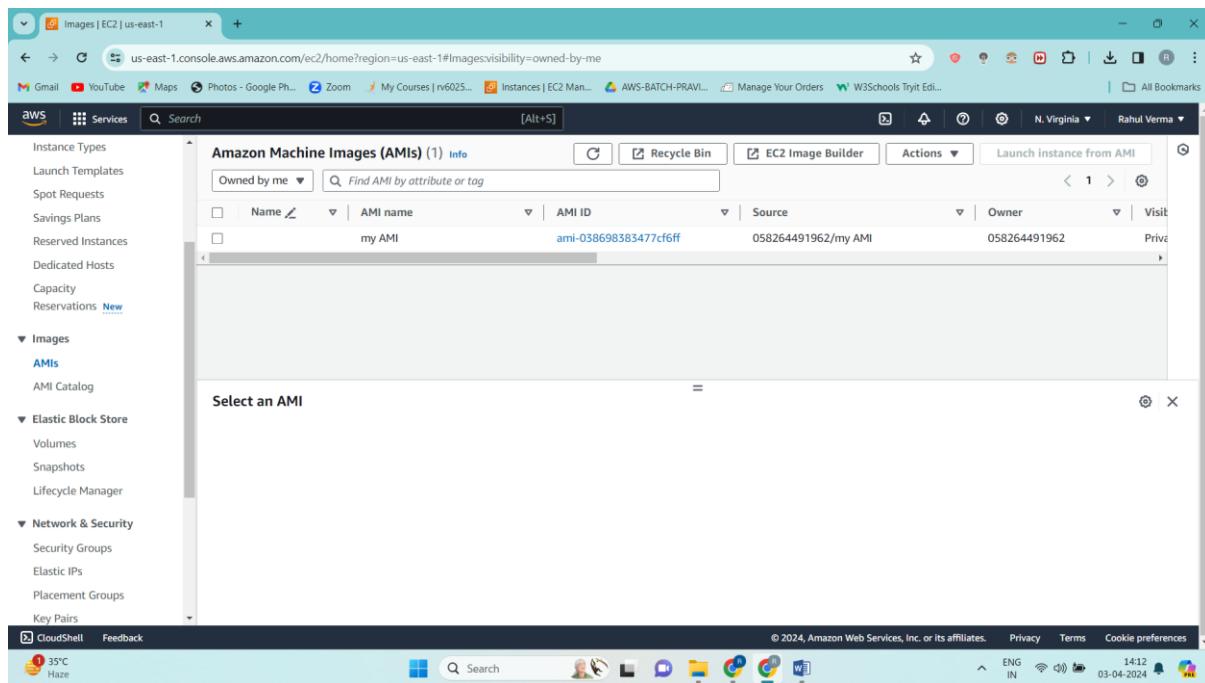
Prerequisite:

- 1) An ec2 instance having web page
- 2) Create image of this instance using AMI

1) Instance having web page



2) AMI of the above Instance

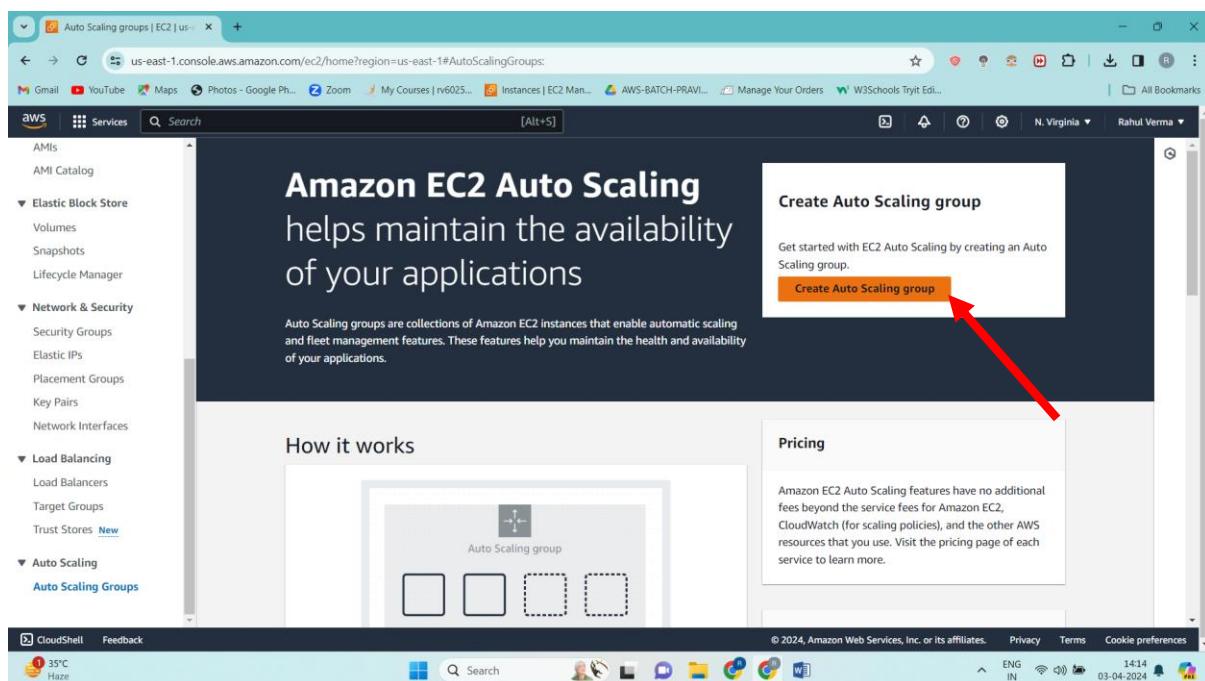


The screenshot shows the AWS EC2 console with the 'Images' section selected. A single AMI named 'my AMI' is listed in the 'Amazon Machine Images (AMIs)' table. The table includes columns for Name, AMI name, AMI ID, Source, Owner, and Privacy. The AMI details are as follows:

Name	AMI name	AMI ID	Source	Owner	Privacy
	my AMI	ami-038698383477cf6ff	058264491962/my AMI	058264491962	Private

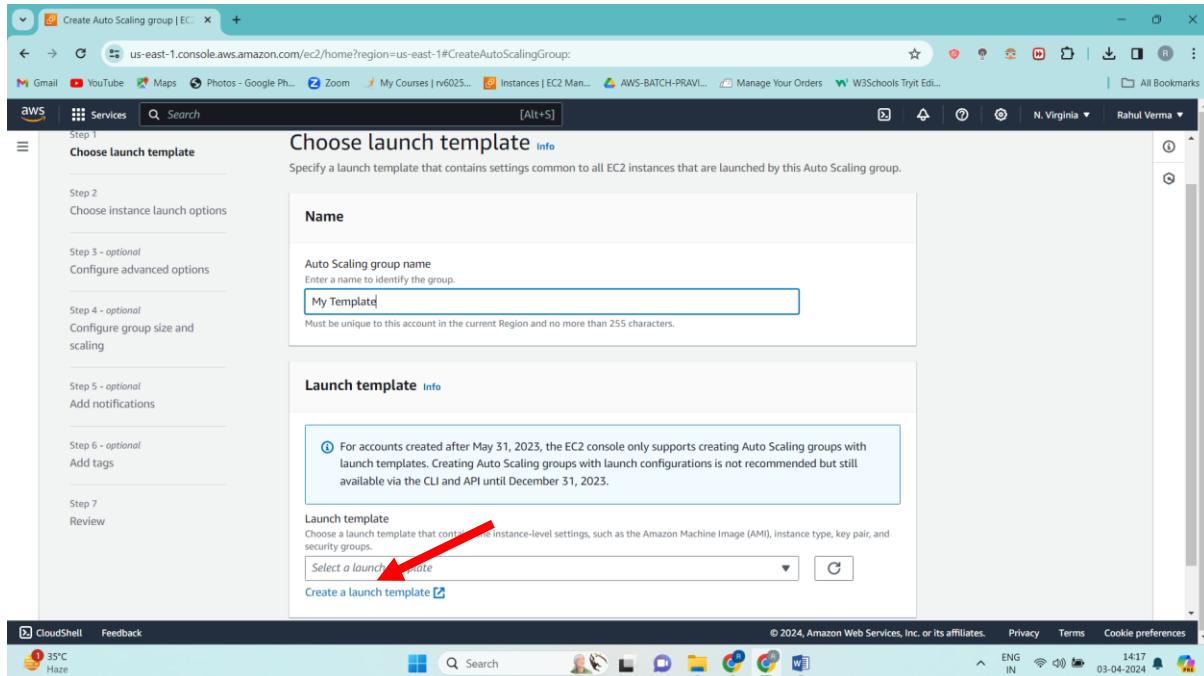
Let's start with creating launch configuration

Step 1: scroll down till bottom and click on Auto scaling and click on Auto scaling groups

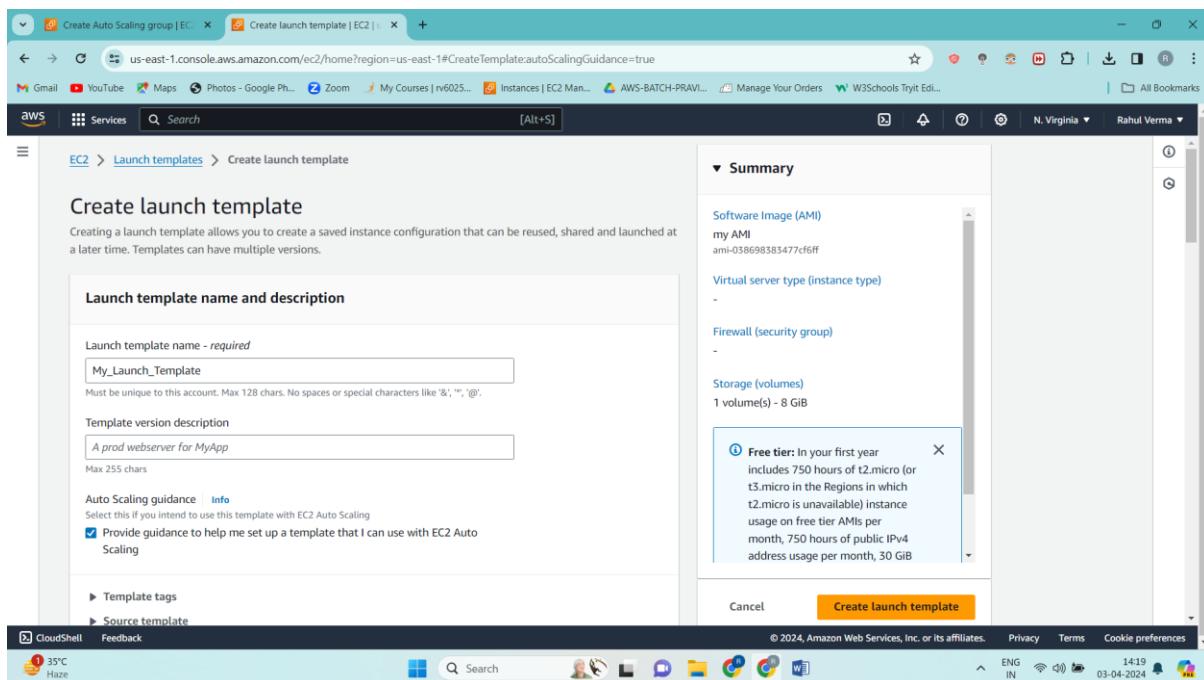


The screenshot shows the AWS EC2 Auto Scaling groups page. A prominent callout box on the right side of the page contains the text 'Create Auto Scaling group' with a red arrow pointing to it. The page also features sections for 'How it works' (a diagram showing an 'Auto Scaling group' with four instances) and 'Pricing' (information about no additional fees). The left sidebar includes options for AMIs, Auto Scaling Groups, and other services like CloudShell and Feedback.

Step 2: Define it's name and click on create a launch template

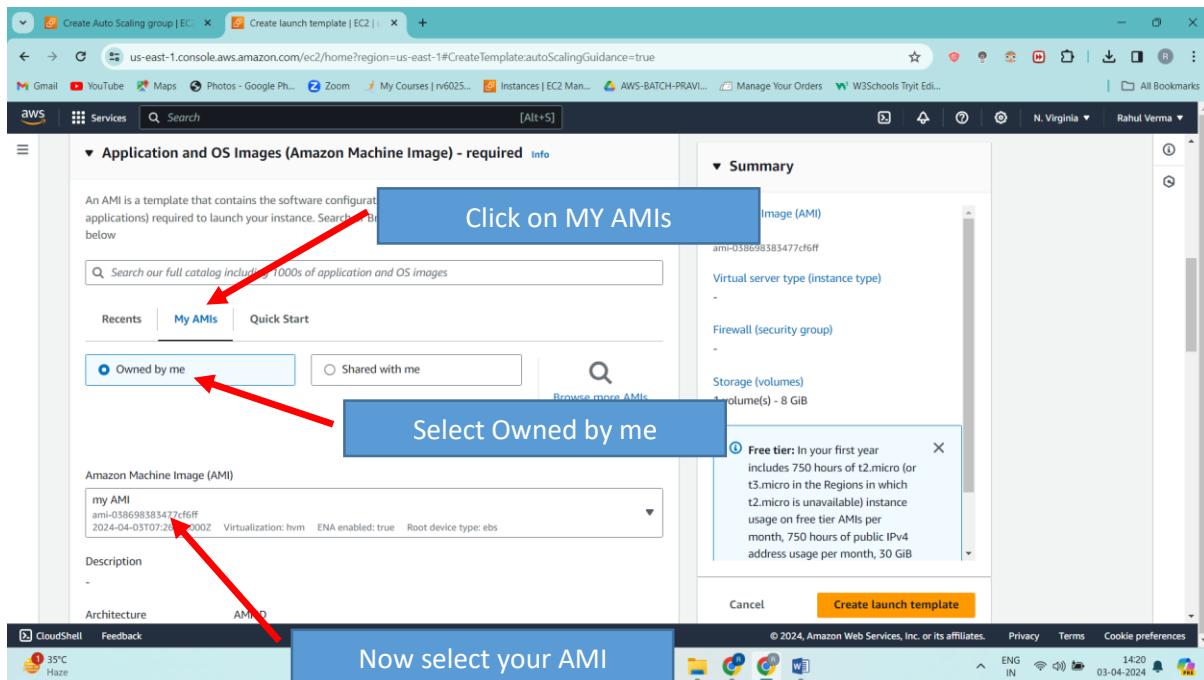


Define template name. I'm giving
My_Launch_Template

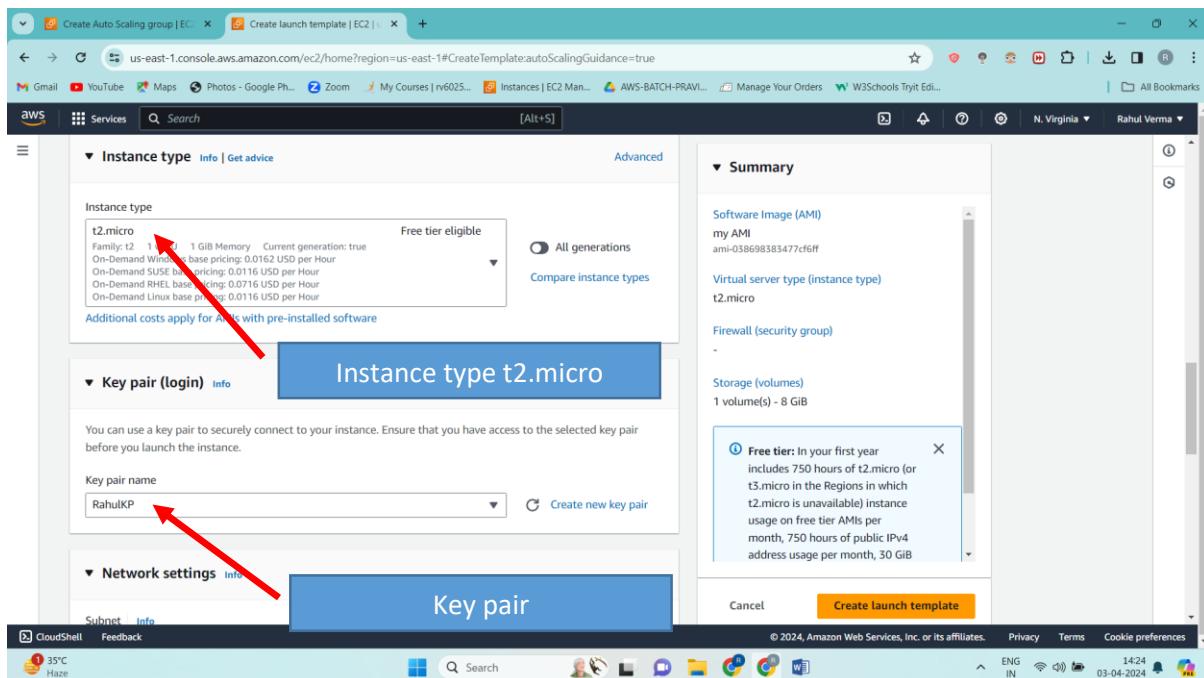


Step 3: Under Application and OS images

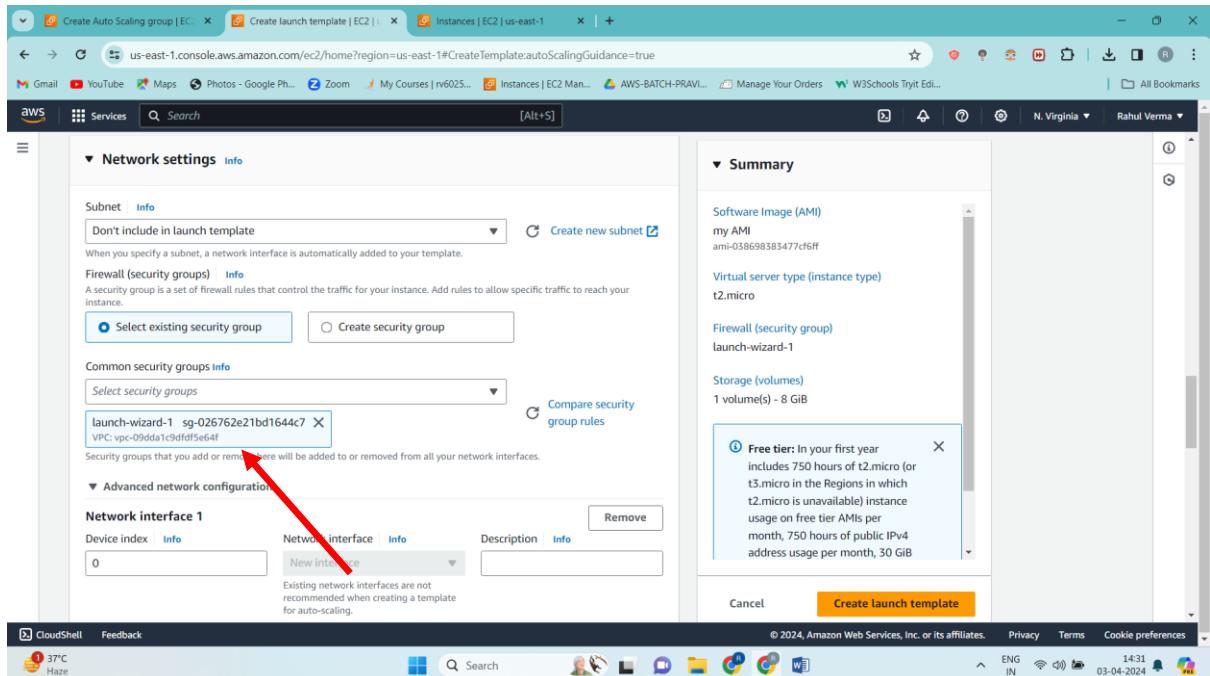
Select your AMI



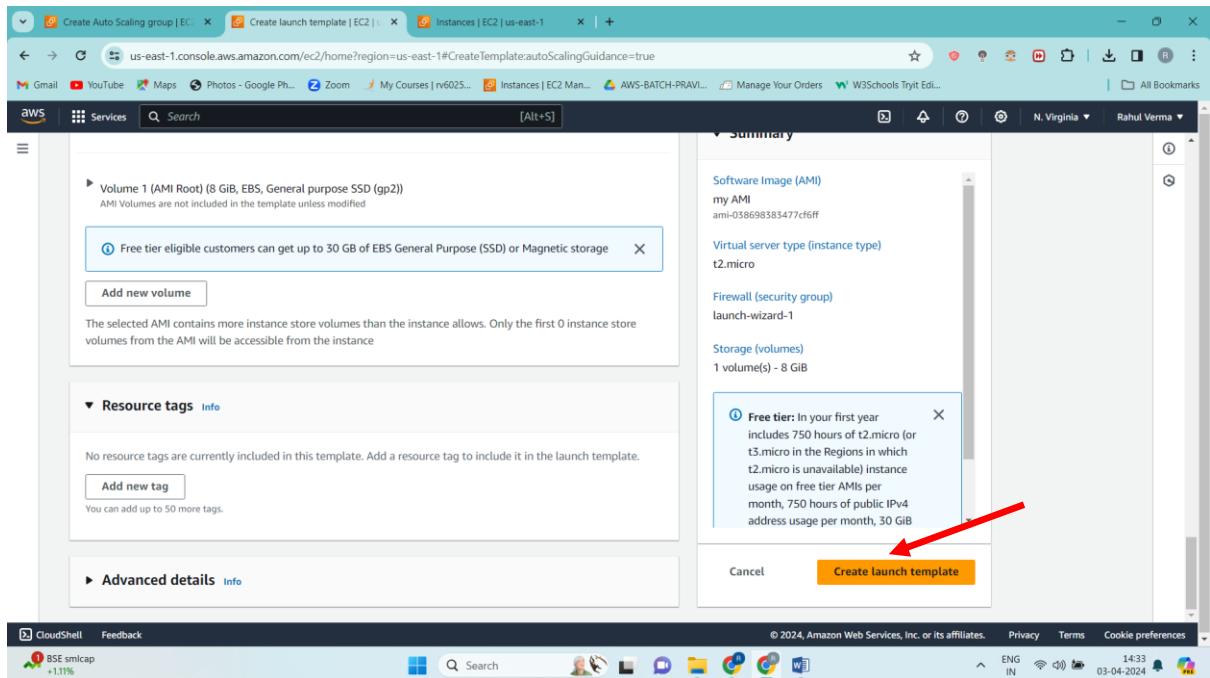
Step 4: Select instance type (for free tier select t2.micro) and select key pair



Step 5: select security group



Rest default setting and just click on create launch template

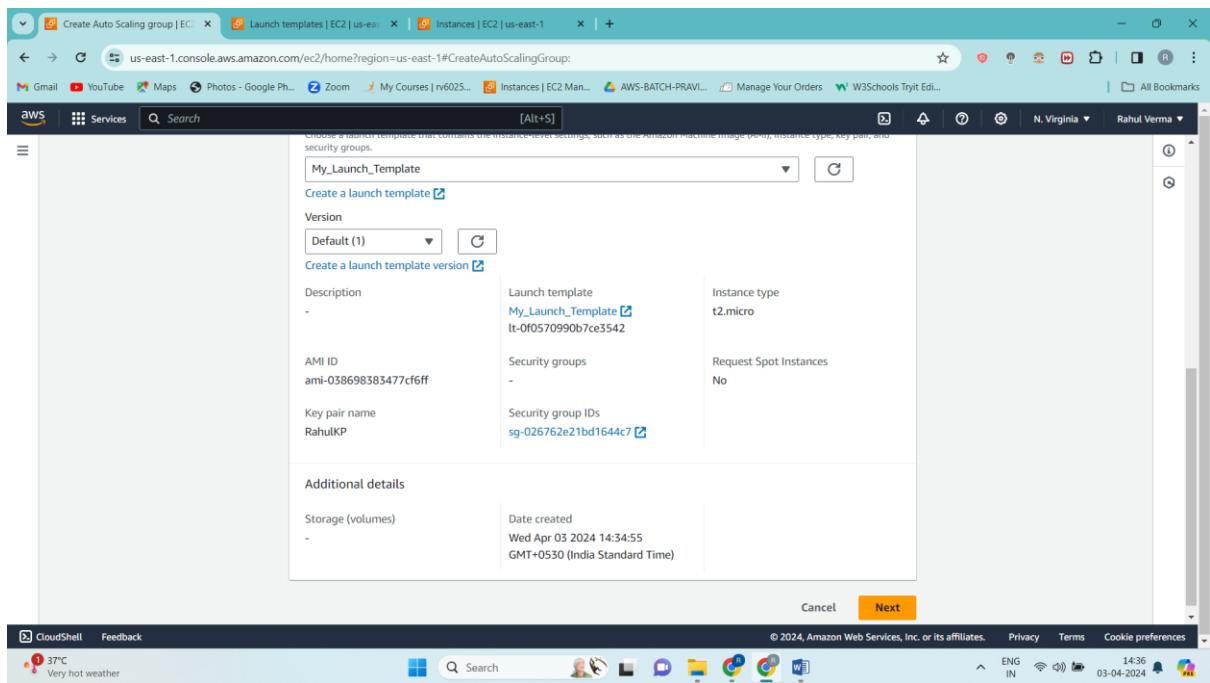


Step 6: now come back to Auto scaling creation page.



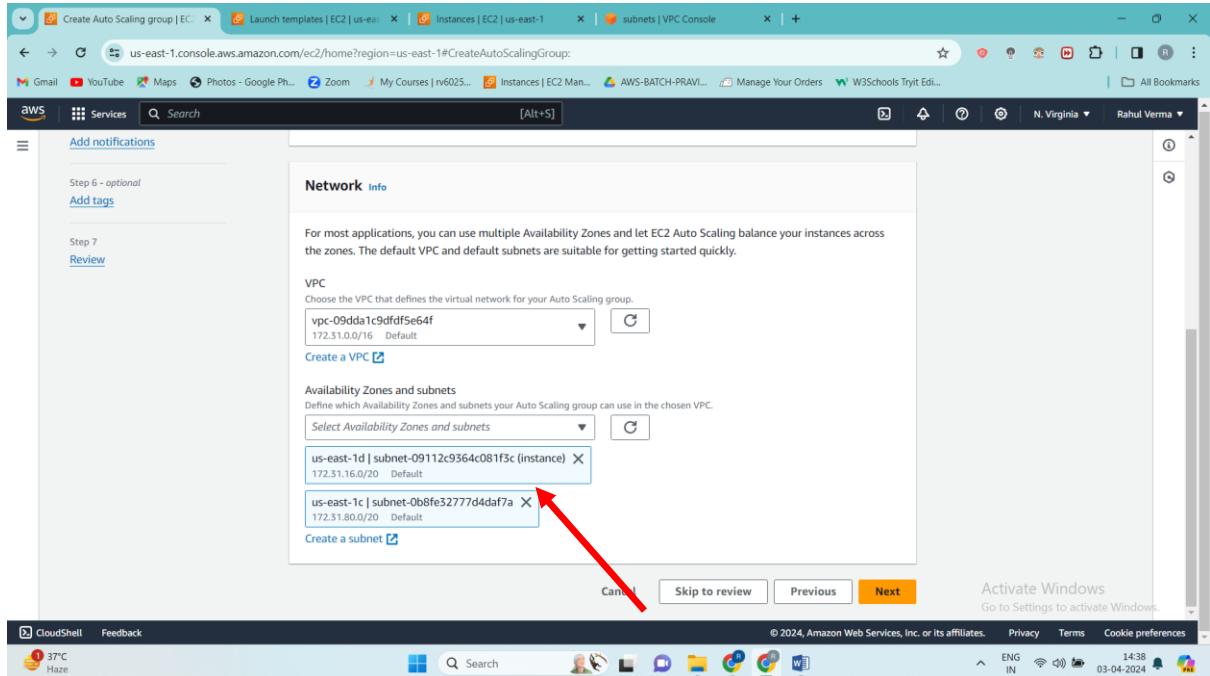
Click on this button

And select your launch template which you have created just now.

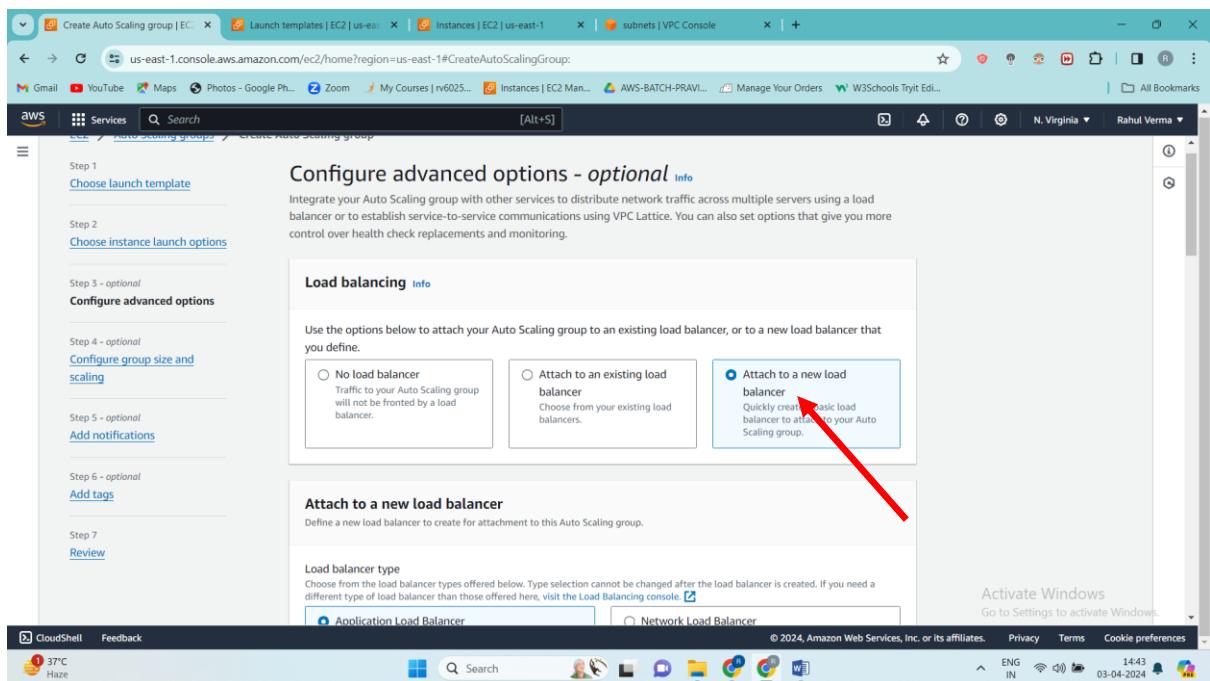


The screenshot shows the AWS CloudFormation Create Stack Wizard Step 2: Set template details. The 'Launch template' dropdown is set to 'My_Launch_Template'. The 'Next Step' button is highlighted in orange.

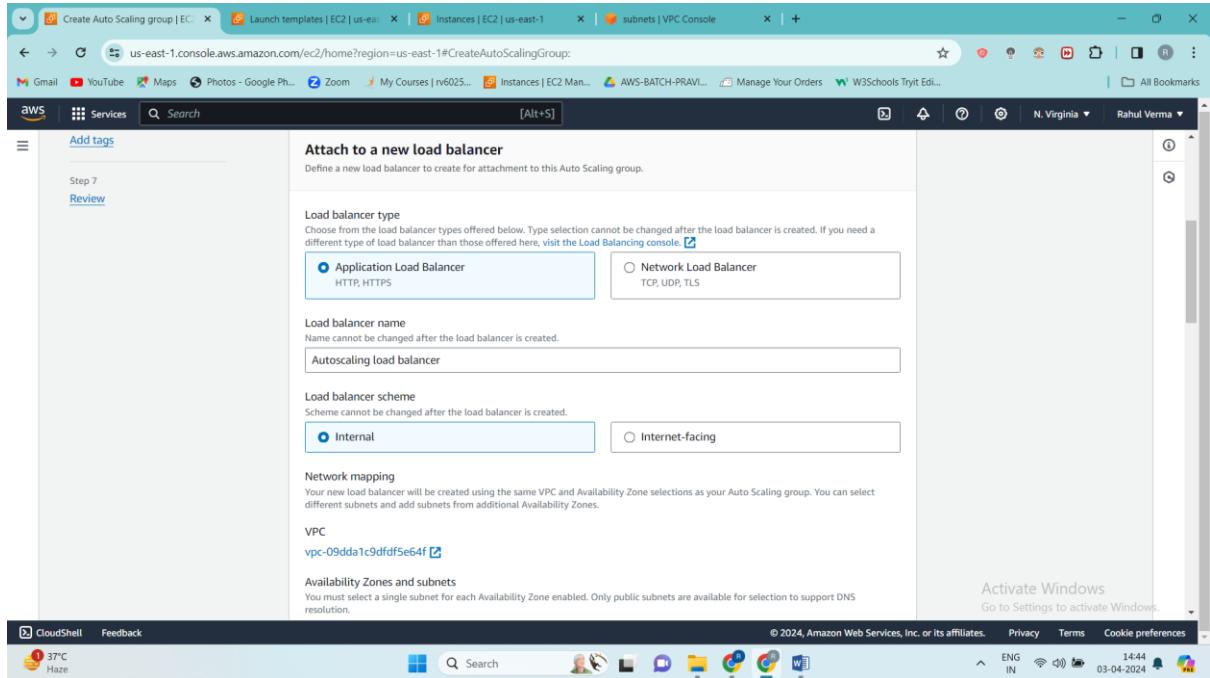
Select availability zone and click on Next



Step 7: now we have to create one Load balancer for that click on attach to a new load balancer

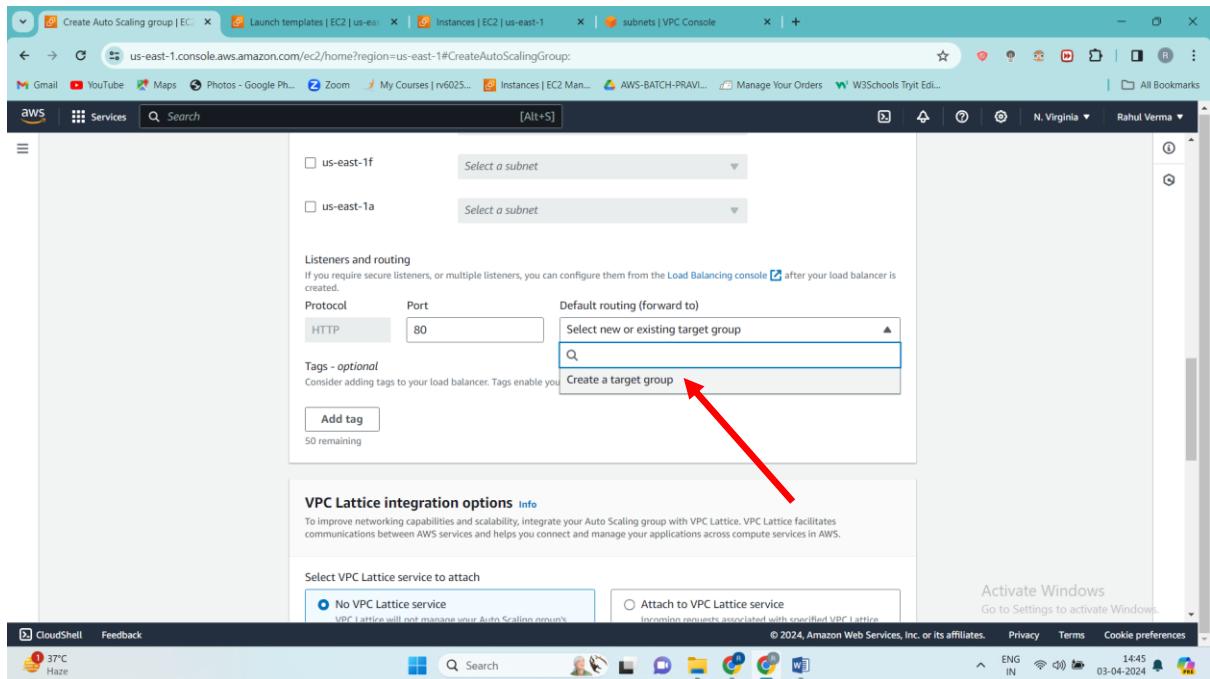


Setup your load balancer

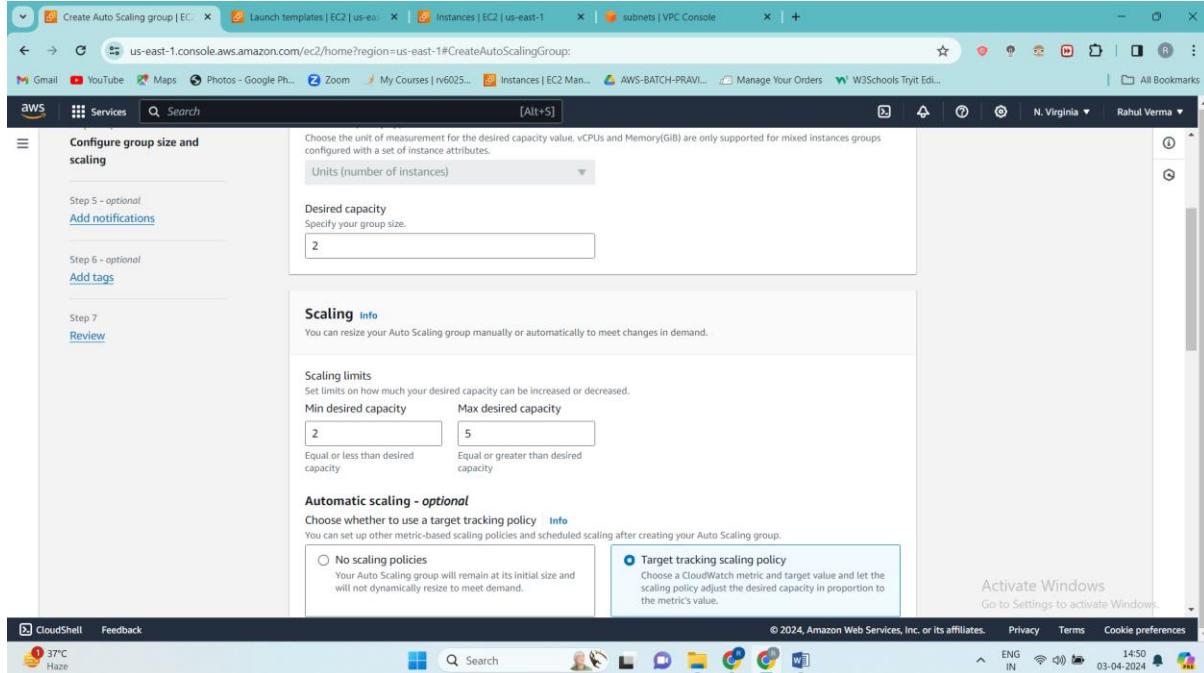


Now create Target group for your load balancer

Click on create a target group and click on next



Step 8: Now configure how much desired instances you want for this demo. I'm giving desired capacity 2 and min 2 instance and max 5 instances



Configure group size and scaling

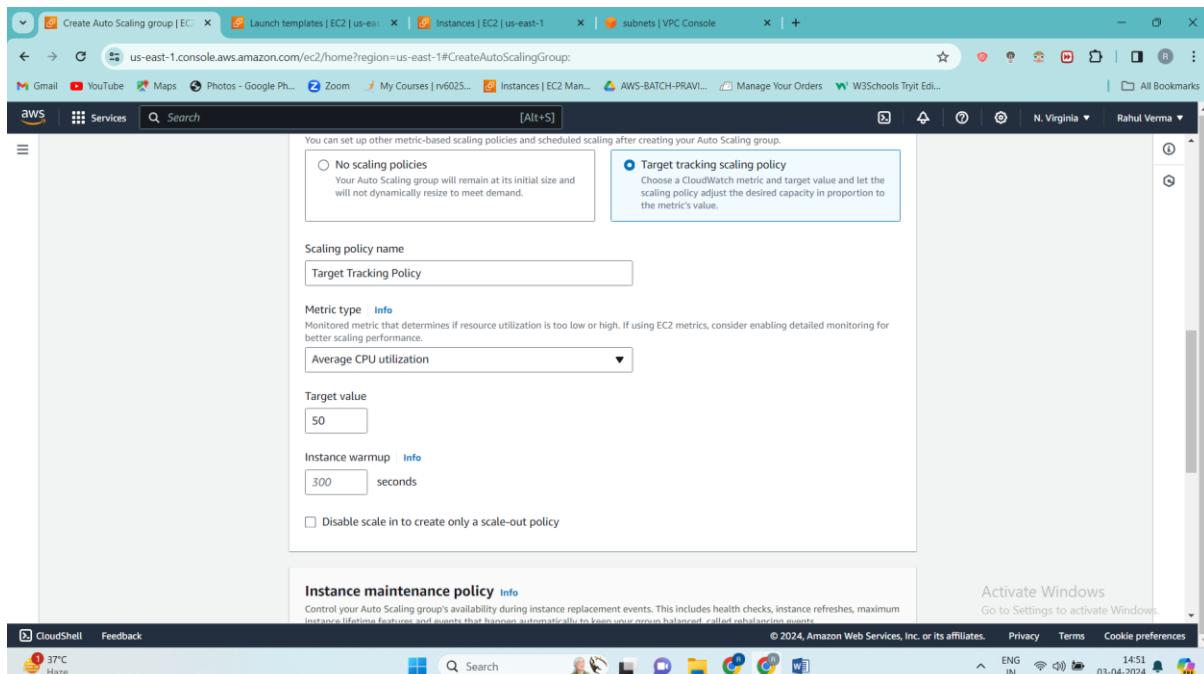
Desired capacity: 2

Min desired capacity: 2

Max desired capacity: 5

Target tracking scaling policy (selected)

Select Target policy.



No scaling policies

Target tracking scaling policy (selected)

Scaling policy name: Target Tracking Policy

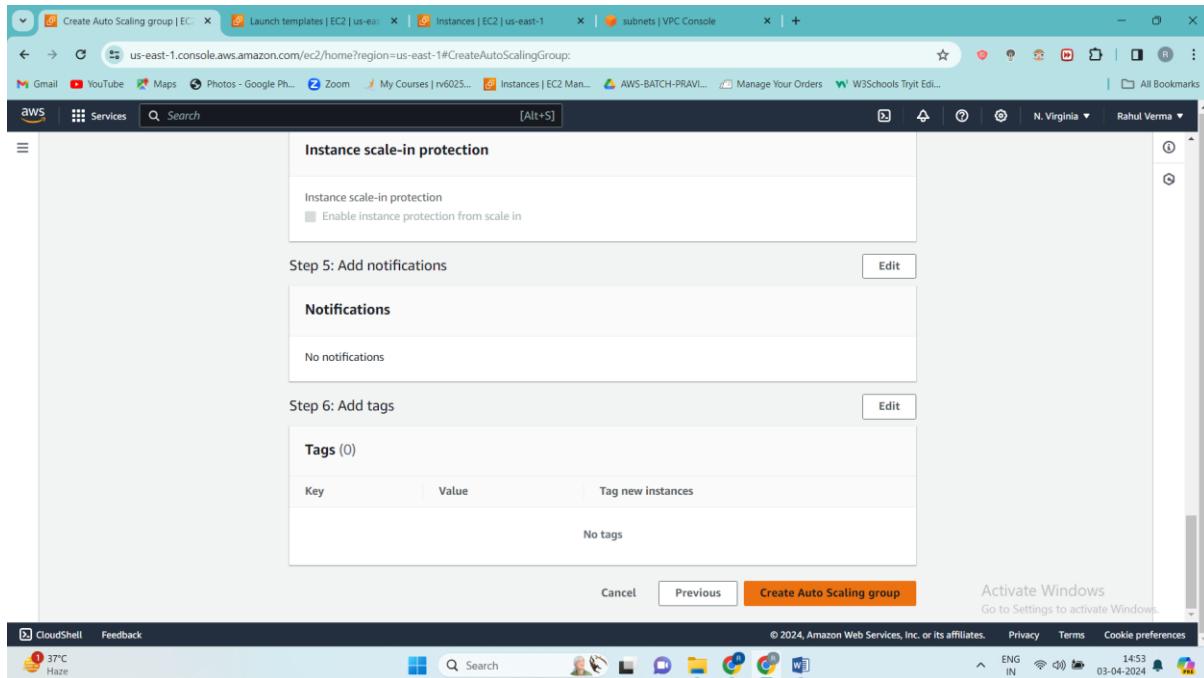
Metric type: Average CPU utilization

Target value: 50

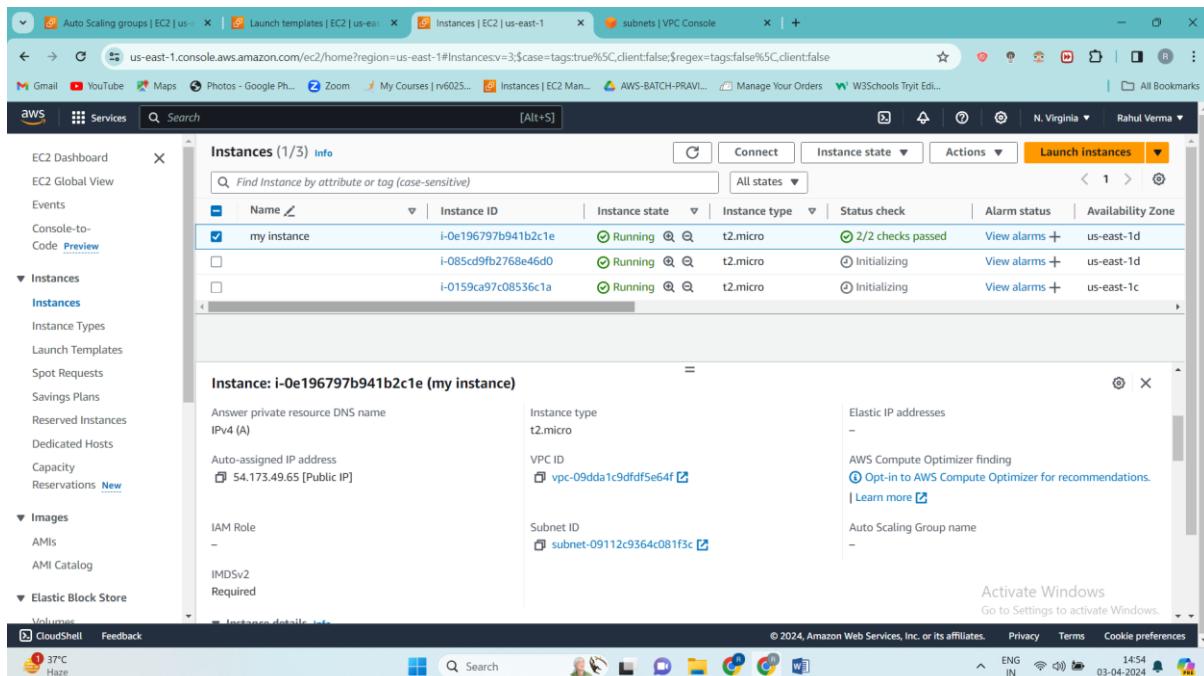
Instance warmup: 300 seconds

Disable scale in to create only a scale-out policy

Now click on Next button until it shows create auto scaling group button and click on it.



Step 9: now you can see 2 instances are created by auto scaling because we have given desired capacity as 2



Now let's connect to one of these instances I have defined them as Auto 1 & Auto 2. Let's connect to Auto 1 and check if our web page is working or not.

Instances (1/3) **Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Auto 1	i-085cd9fb2768e46d0	Running	t2.micro	2/2 checks passed	View alarms	us-east-1d
my instance	i-0e196797b941b2c1e	Running	t2.micro	2/2 checks passed	View alarms	us-east-1d
Auto 2	i-0159ca97c08536c1a	Running	t2.micro	2/2 checks passed	View alarms	us-east-1c

Instance: i-085cd9fb2768e46d0 (Auto 1)

Details | Status and alarms **New** | Monitoring | Security | Networking | **Storage** | Tags

Instance summary **Info**

Instance ID	i-085cd9fb2768e46d0 (Auto 1)	Public IPv4 address	54.87.173.150 Open address
IPv6 address	-	Instance state	Running
Hostname type	IP name: ip-172-31-29-119.ec2.internal	Private IP DNS name (IPv4 only)	ip-172-31-29-119.ec2.internal

We are connected to Auto 1

```

System load: 0.005859375  Processes: 105
Usage of /: 23.3% of 7.57GB  Users logged in: 0
Memory usage: 21%
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

45 updates can be applied immediately.
30 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

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.

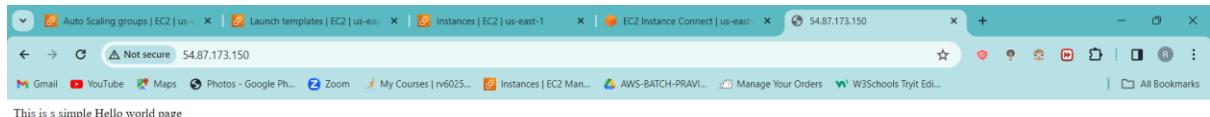
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ip-172-31-29-119:~# [REDACTED]

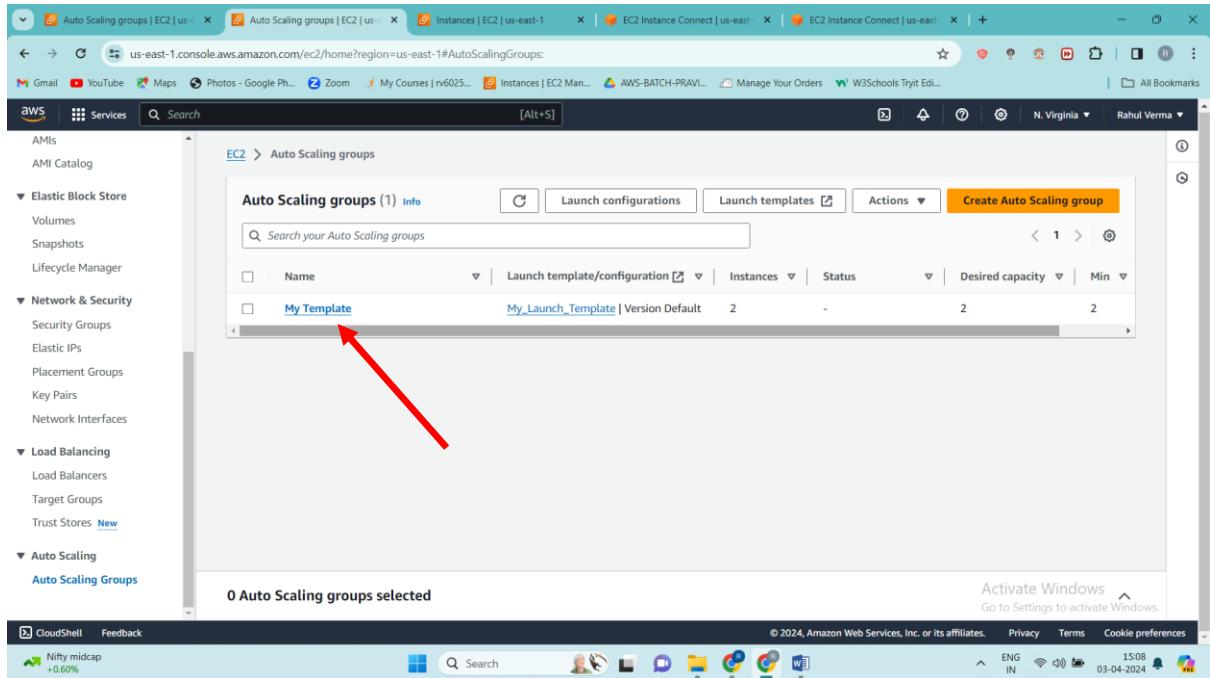
i-085cd9fb2768e46d0 (Auto 1)
PublicIPs: 54.87.173.150  PrivateIPs: 172.31.29.119

```

And our web is also working good

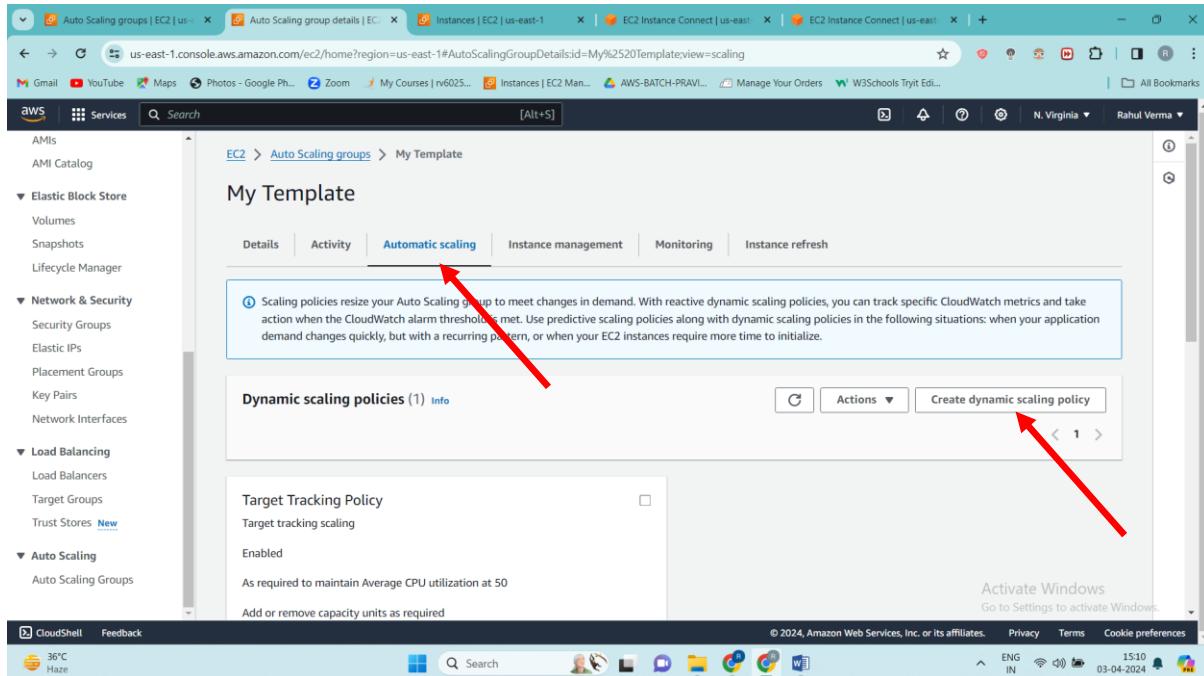


Step 10: now go back to your Auto scaling group click on it



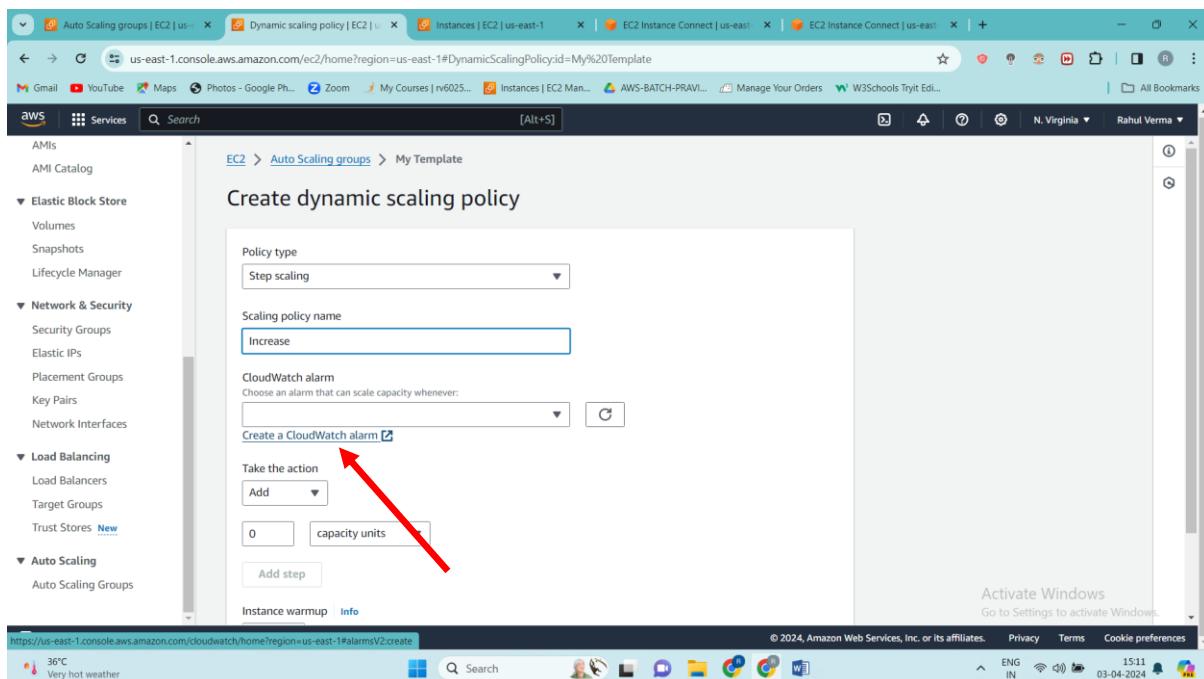
Now go to automatic scaling

Under that click on create dynamic scaling policy



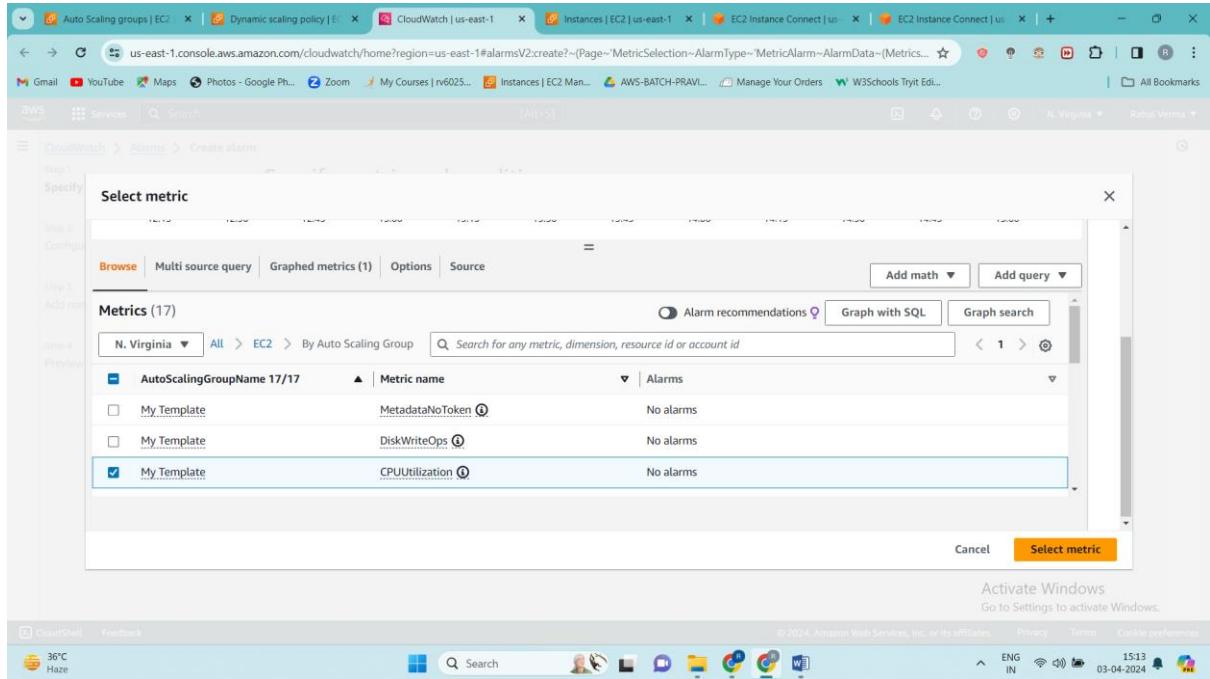
The screenshot shows the AWS EC2 console with the 'Auto Scaling groups' section selected. Under 'My Template', the 'Automatic scaling' tab is active. A red arrow points to the 'Create dynamic scaling policy' button in the top right of the main content area. The page also displays a 'Target Tracking Policy' section with a checkbox for 'Target tracking scaling' and a note about maintaining average CPU utilization at 50.

Step 11: define it's name and click on create a cloud watch alarm.



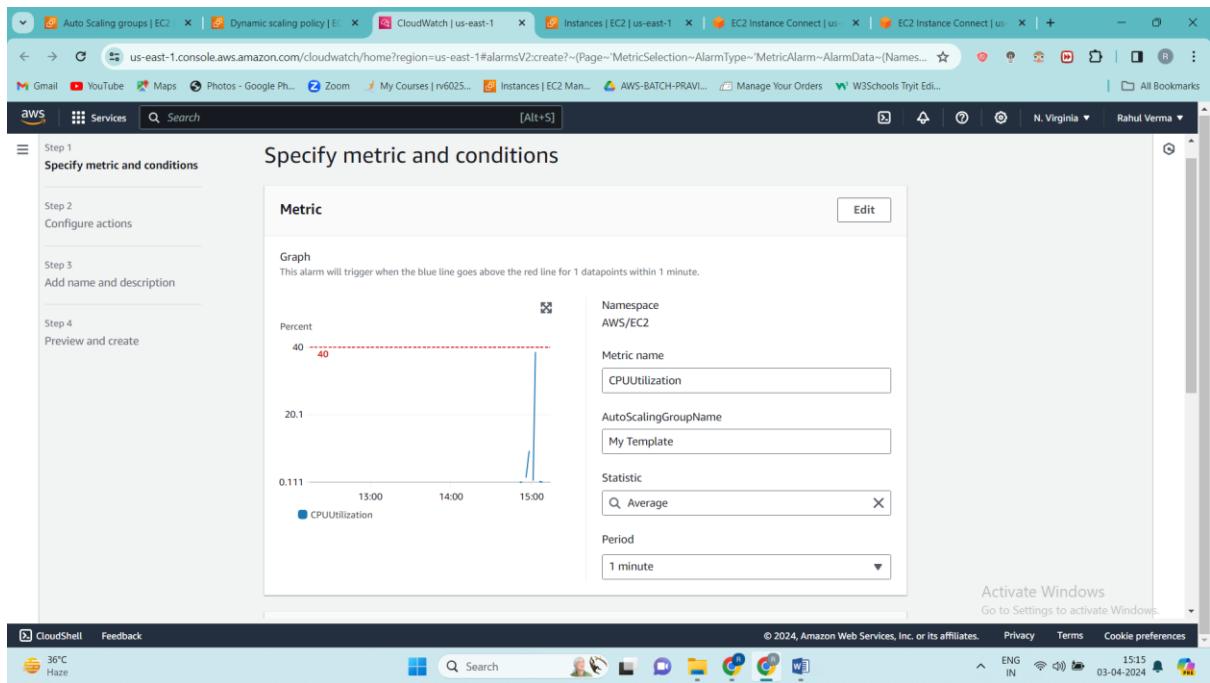
The screenshot shows the 'Create dynamic scaling policy' page. The 'Policy type' is set to 'Step scaling'. The 'Scaling policy name' is 'Increase'. In the 'CloudWatch alarm' section, a red arrow points to the 'Create a CloudWatch alarm' link. The 'Take the action' section shows an 'Add' button and a 'capacity units' input field set to 0. The page also includes an 'Instance warmup' section and a link to 'https://us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#alarmsV2:create'.

Now browse metrics and select My template (auto scaling group name) CPU utilization



The screenshot shows the 'Select metric' dialog box from the AWS CloudWatch Metrics interface. The 'Metrics' section lists several metrics, with 'My Template' selected. The selected metric details are displayed: Metric name is 'CPUUtilization', Namespace is 'AWS/EC2', and Metric name is 'CPUUtilization'. The dialog box has 'Select metric' and 'Cancel' buttons.

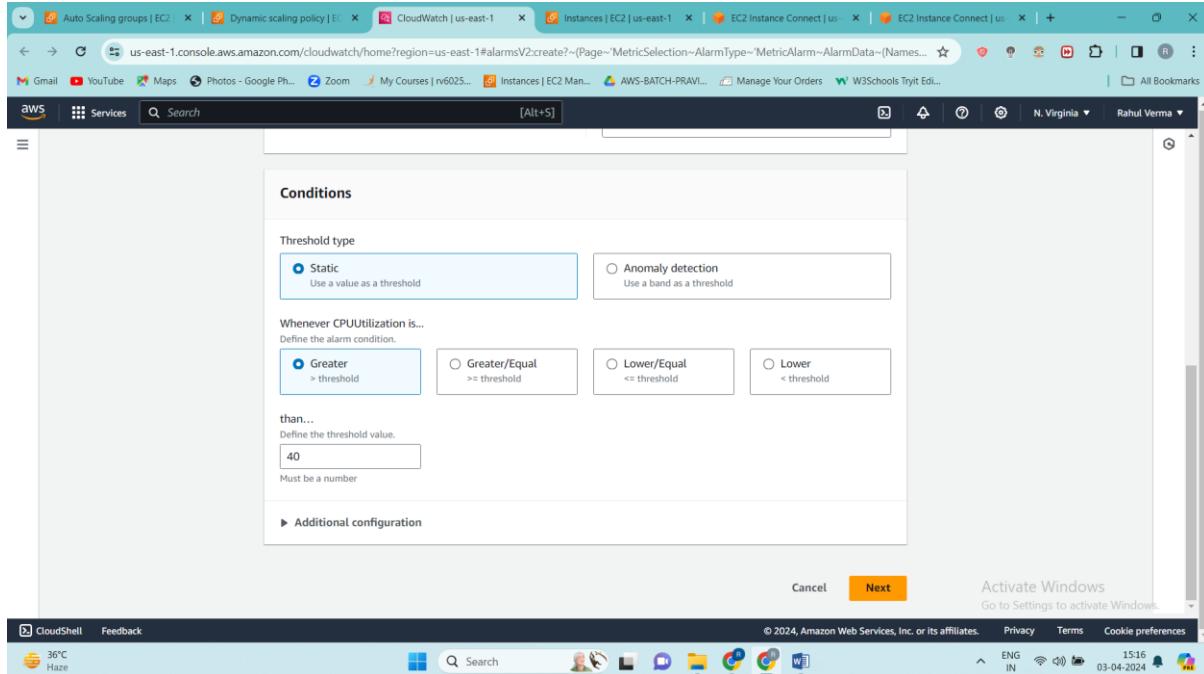
Select the desired settings



The screenshot shows the 'Specify metric and conditions' step in the AWS CloudWatch Metrics creation wizard. The 'Metric' section displays a graph of CPUUtilization over time, with a red dashed line at 40% and a blue line peaking above it. The graph is labeled 'Graph' and 'This alarm will trigger when the blue line goes above the red line for 1 datapoint within 1 minute.' The right side of the dialog shows configuration fields: Namespace (AWS/EC2), Metric name (CPUUtilization), AutoScalingGroupName (My Template), Statistic (Average), and Period (1 minute). The dialog has 'Edit' and 'Next Step' buttons.

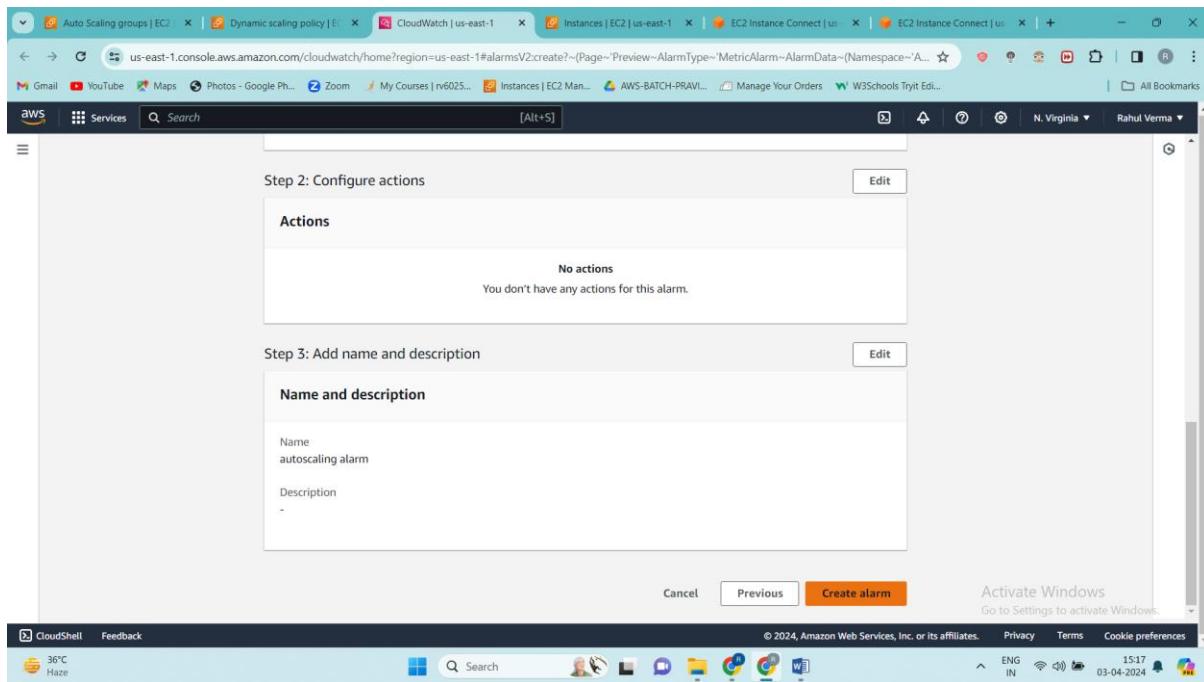
And give some conditions-

If cpu utilization goes more than 40% than alarm will trigger



The screenshot shows the 'Conditions' step of creating a CloudWatch Metrics alarm. The 'Threshold type' is set to 'Static' (selected). The 'Whenever CPUUtilization is...' condition is set to 'Greater' (> threshold), with a value of '40' entered. The 'Next' button is visible at the bottom right.

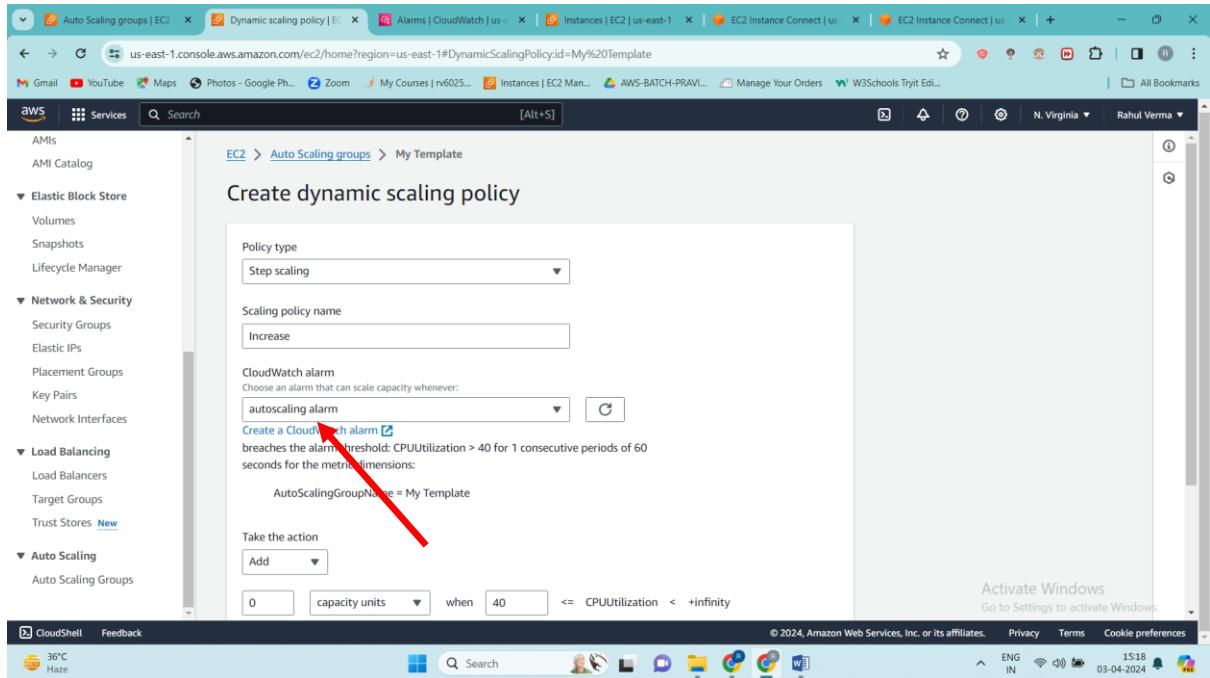
Now preview and click on create alarm



The screenshot shows the 'Step 2: Configure actions' and 'Step 3: Add name and description' steps of creating a CloudWatch Metrics alarm. Both steps show 'No actions' and a name of 'autoscaling alarm'. The 'Create alarm' button is visible at the bottom right.

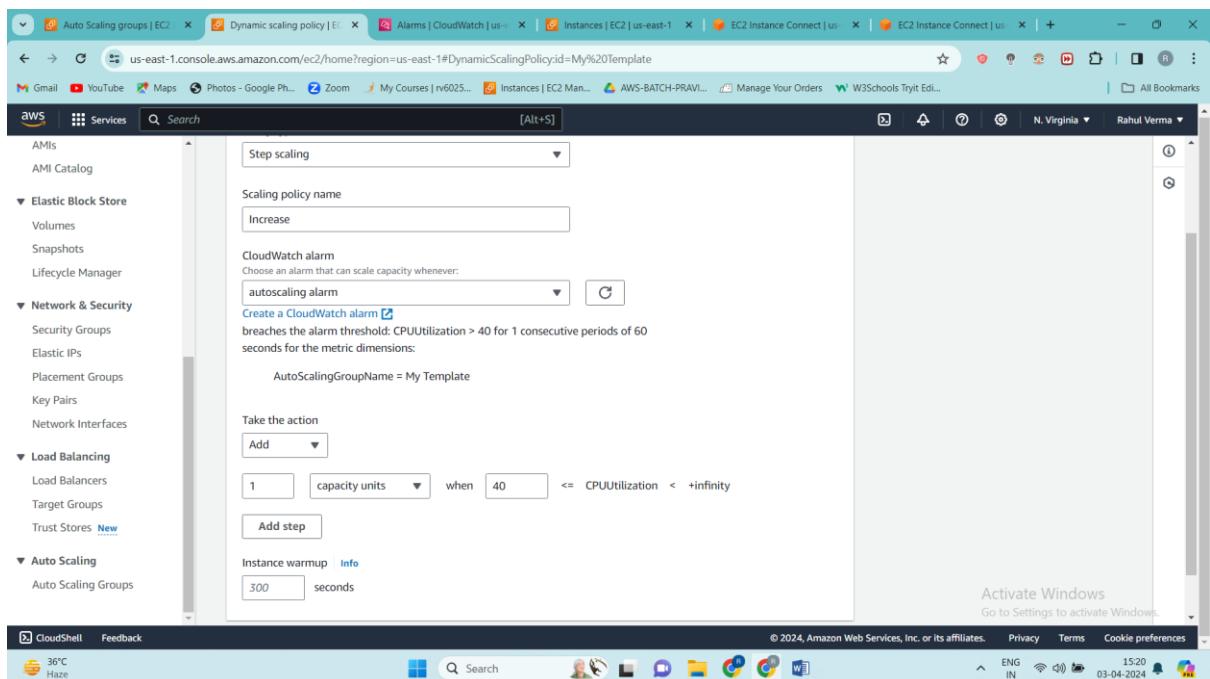
Step 12: now go back to dynamic policy creation page

Select your alarm which you have created just now



The screenshot shows the 'Create dynamic scaling policy' page in the AWS CloudWatch Metrics console. A red arrow points to the 'CloudWatch alarm' dropdown menu, which is set to 'autoscaling alarm'. The policy type is 'Step scaling' and the scaling policy name is 'Increase'. The 'CloudWatch alarm' section shows a threshold of 'CPUUtilization > 40 for 1 consecutive periods of 60 seconds'. The 'Take the action' section shows an 'Add' button, a 'capacity units' input of '1', and a 'when' input of '40'. The status bar at the bottom indicates '36°C Haze'.

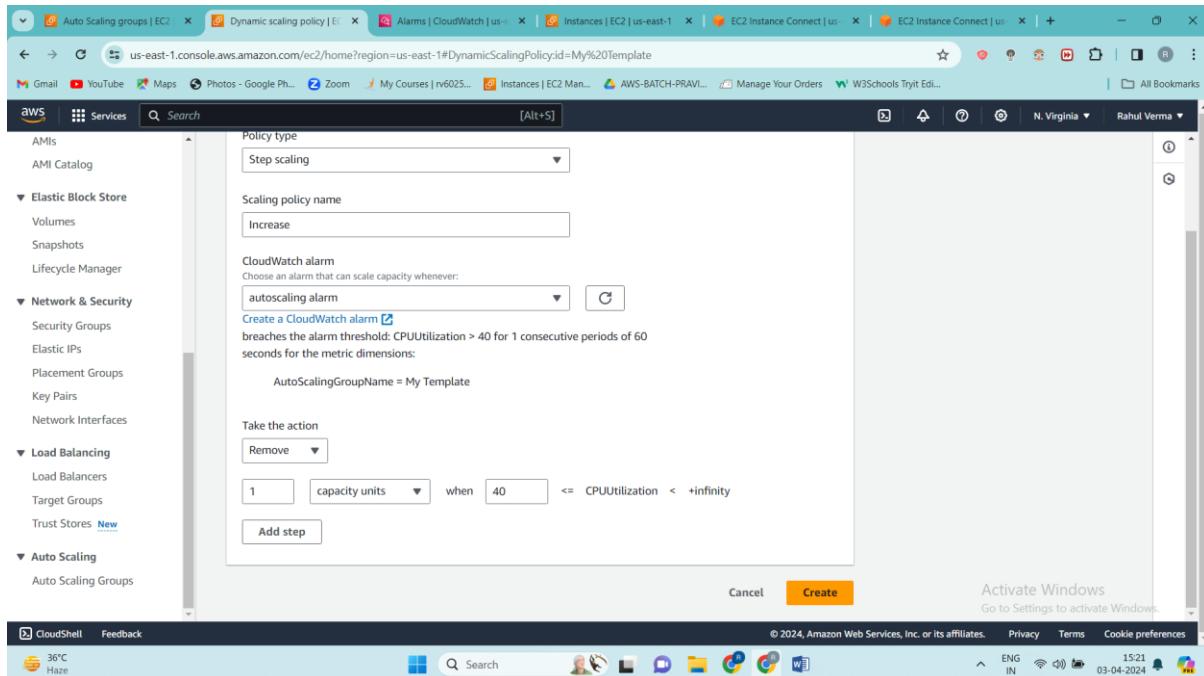
And here we have defined if cpu utilization goes above 40 % it will add one instance.



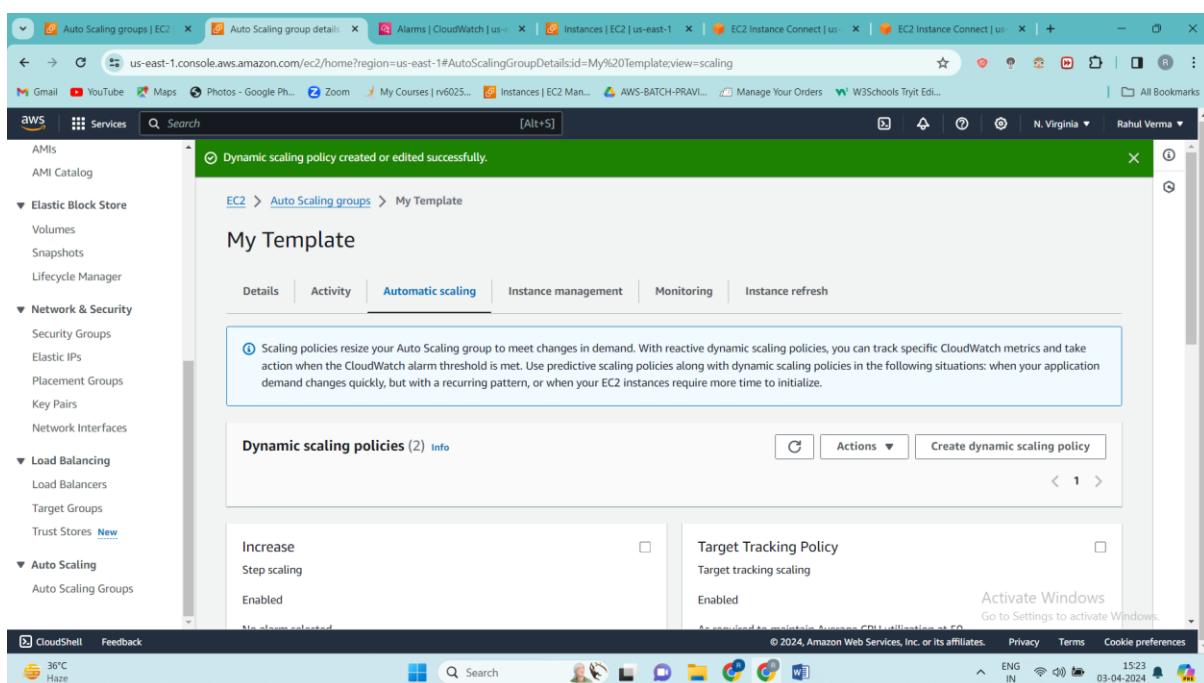
The screenshot shows the 'Create dynamic scaling policy' page in the AWS CloudWatch Metrics console. The 'CloudWatch alarm' dropdown is set to 'Create a CloudWatch alarm' (highlighted with a red arrow). The 'Take the action' section shows an 'Add' button, a 'capacity units' input of '1', and a 'when' input of '40'. The 'Instance warmup' section shows a 'seconds' input of '300'. The status bar at the bottom indicates '36°C Haze'.

Note: same can be done for remove also for that you have to create another cloud watch alarm

Like If cpu utilization goes below 30% it will delete one instance

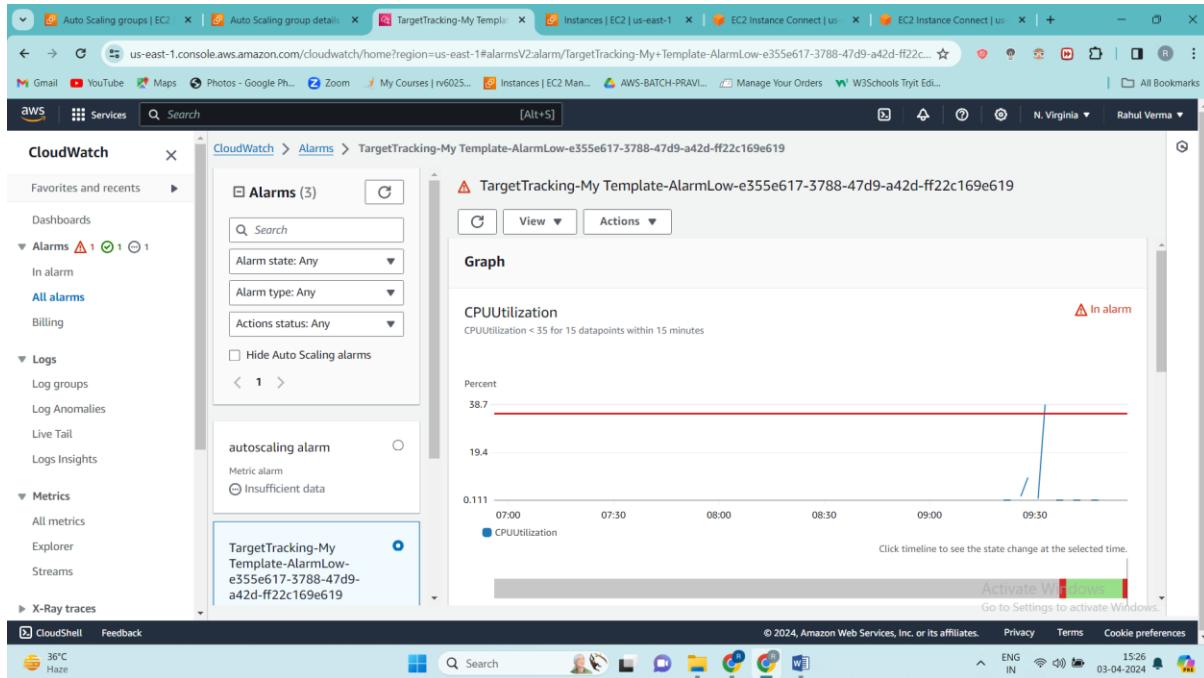


We have created for adding instance only

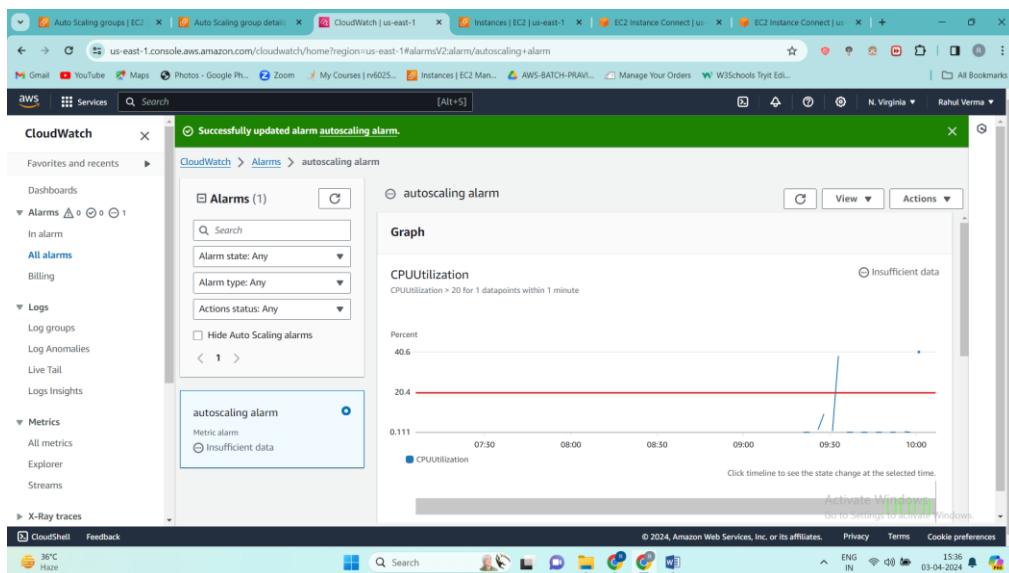


Step 13: let's increase the load so that it will go above 40% cpu utilization and it should create one instance.

Our alarm is triggered now



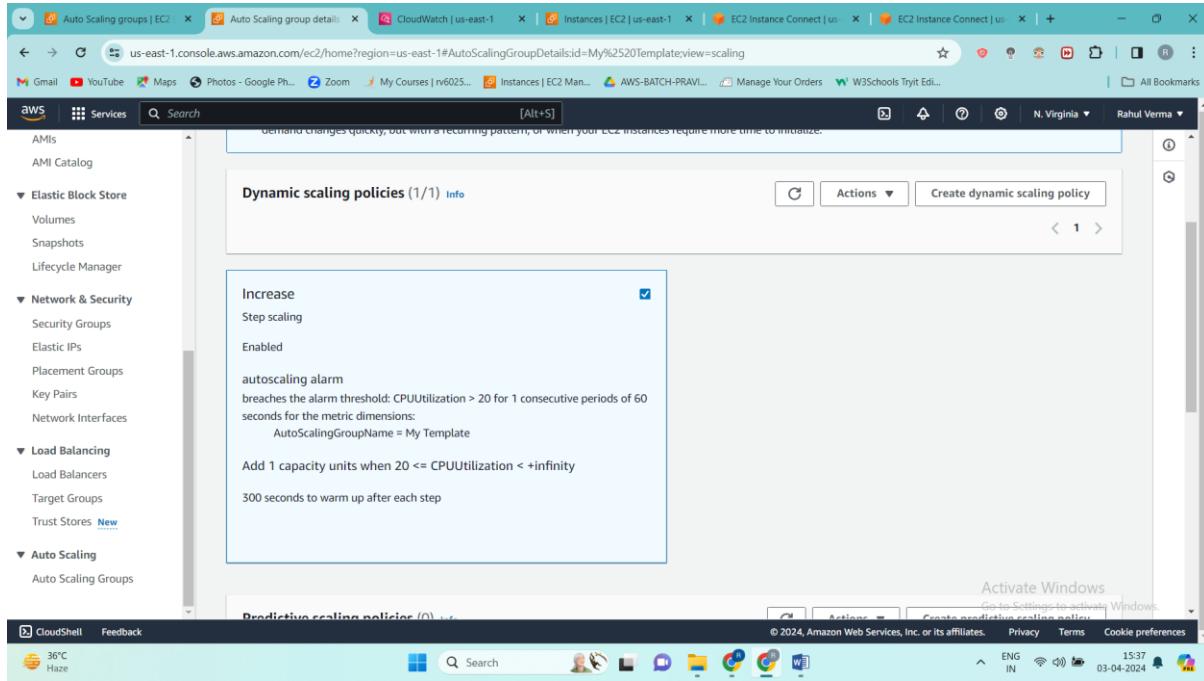
We have updated the alarm to spike the load faster. Instead of 40 we have change it to 20 now.



Use following command to spike the load-

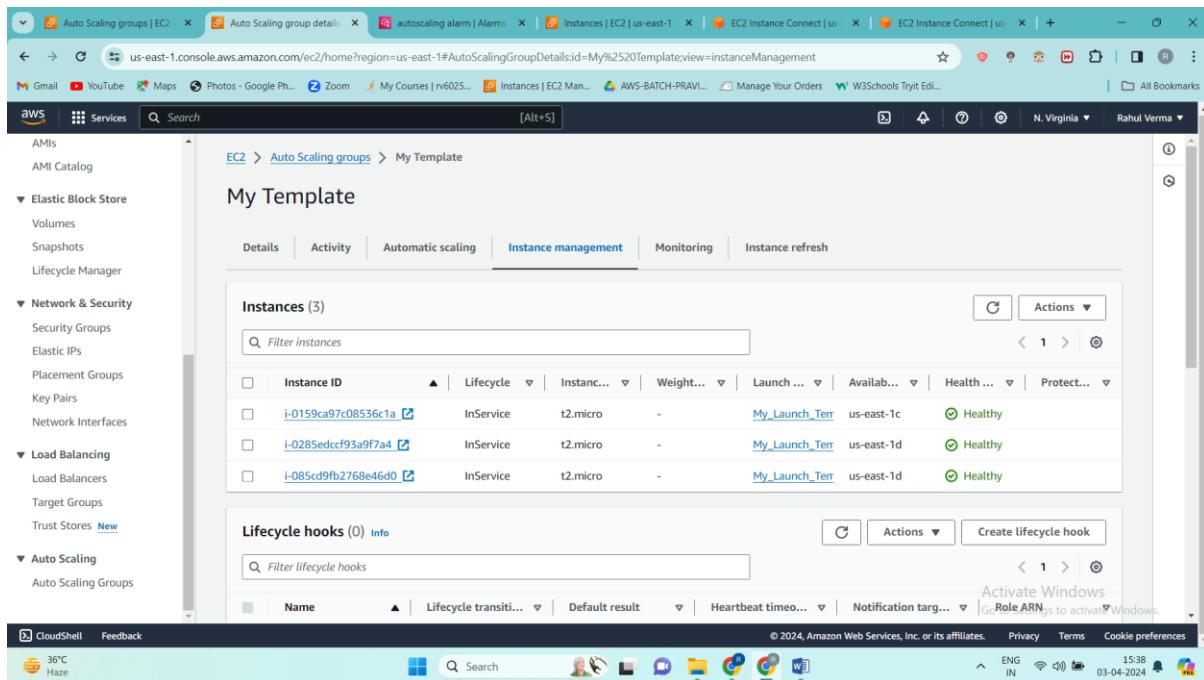
yes > /dev/null

You can see in auto scaling also



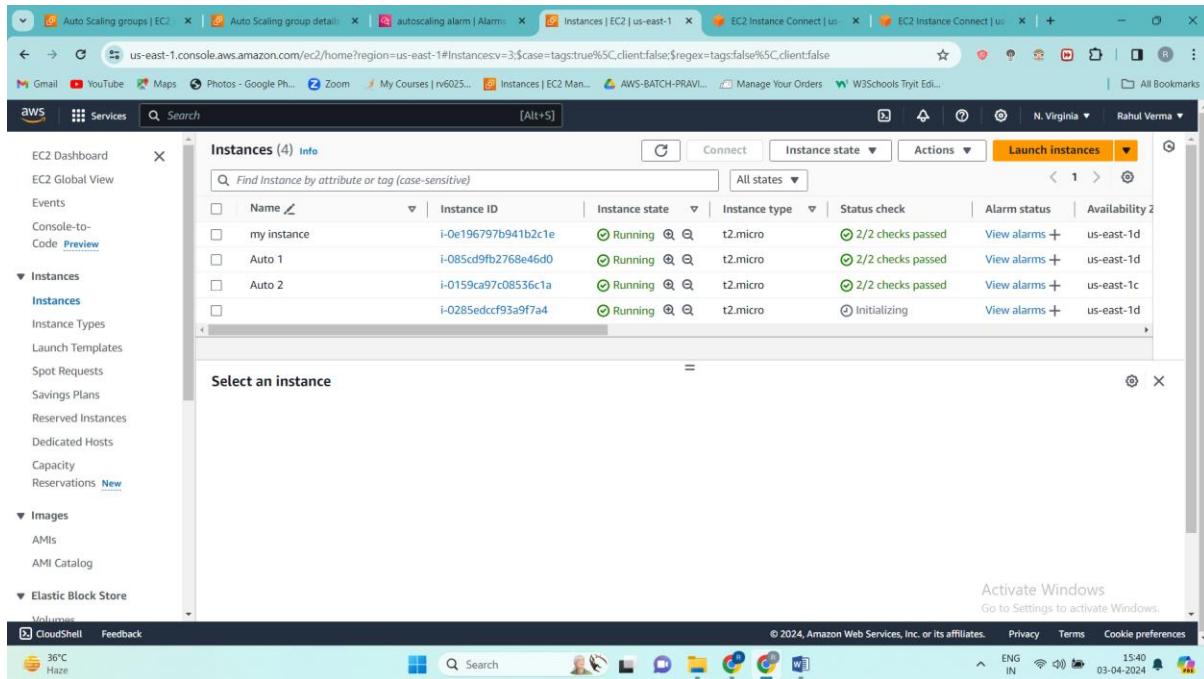
The screenshot shows the AWS CloudWatch Metrics Insights interface. A step scaling policy is displayed, titled 'Increase'. The policy is set to 'Step scaling' and is 'Enabled'. It triggers when 'CPUUtilization > 20' for 60 seconds. The policy adds 1 capacity unit when the condition is met. A 'warm up' period of 300 seconds is specified. The policy is associated with an Auto Scaling group named 'My Template'.

And it's done. It has created third instance



The screenshot shows the AWS Auto Scaling Groups interface for a group named 'My Template'. The 'Instance management' tab is selected. The 'Instances' section shows three instances in the 'InService' state, each with an 'Instance ID' (i-0159ca97c08536c1a, i-0285edccf93a9f7a4, i-085cd9fb2768e46d0), 'Lifecycle' (t2.micro), and 'Health' status (Healthy). The 'Lifecycle hooks' section is empty.

And it's created one instance automatically



The screenshot shows the AWS EC2 Instances page with four instances listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
my instance	i-0e196797b941b2c1e	Running	t2.micro	2/2 checks passed	View alarms	us-east-1d
Auto 1	i-085cd9fb2768e46d0	Running	t2.micro	2/2 checks passed	View alarms	us-east-1d
Auto 2	i-0159ca97c08536c1a	Running	t2.micro	2/2 checks passed	View alarms	us-east-1c
	i-0285edccf93a9f7a4	Running	t2.micro	Initializing	View alarms	us-east-1d

A modal window titled "Select an instance" is open, showing the same four instances. The "my instance" is selected.

Note: To avoid charges after using the resources do delete them.

Thank You