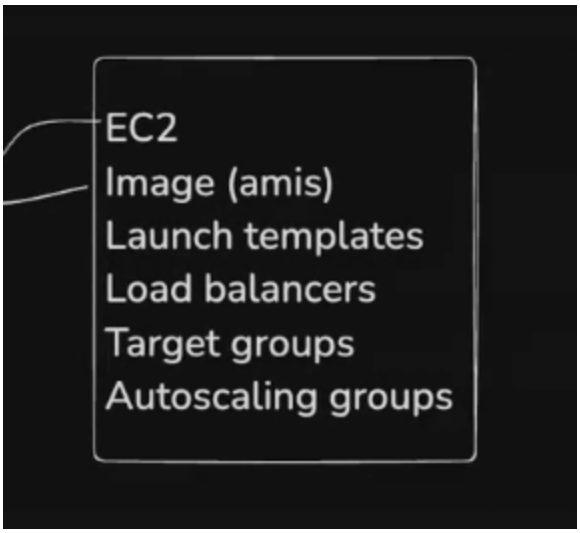
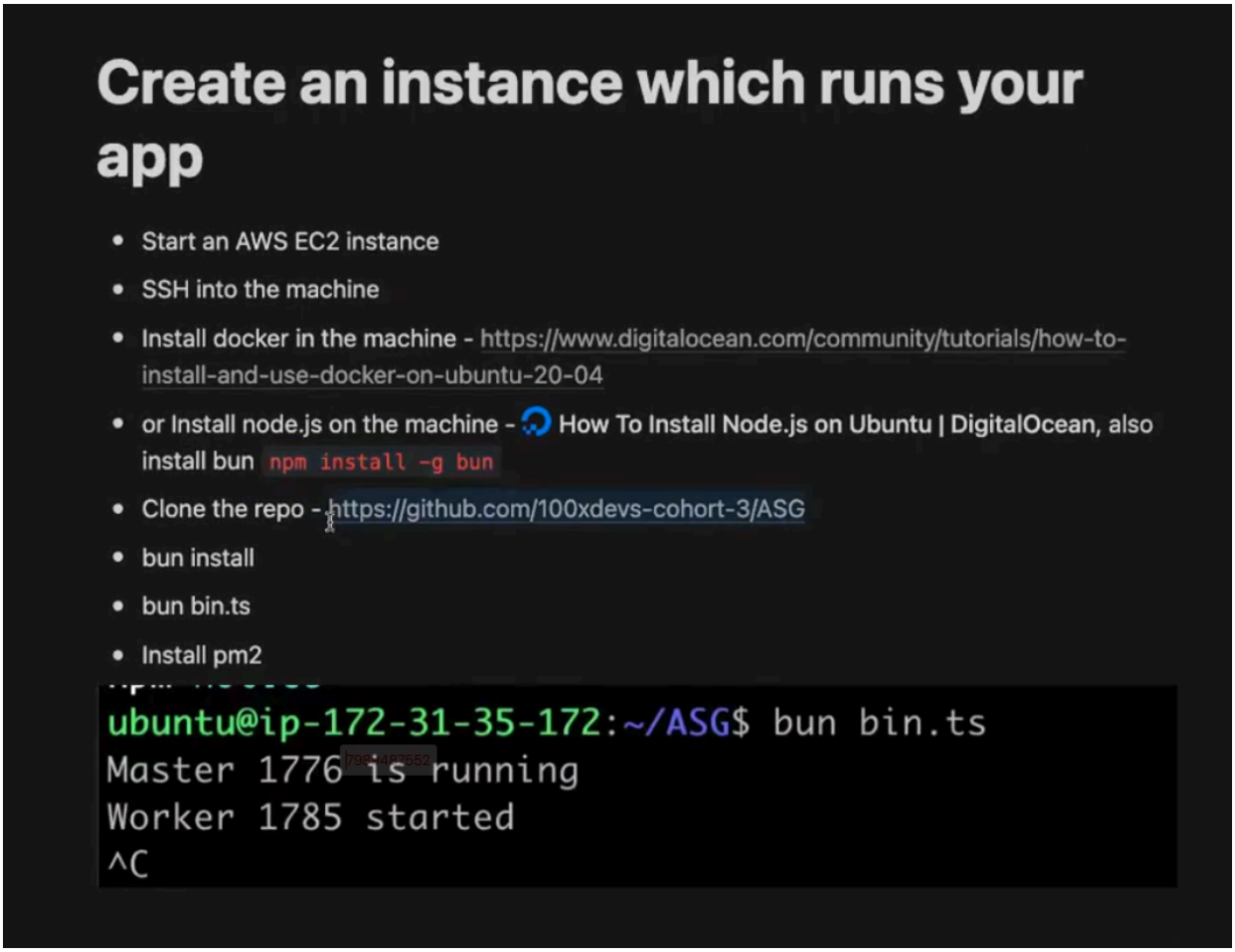
Practical (Steps to create and deploy your application to ASG)

Steps to create and deploy your application to ASG



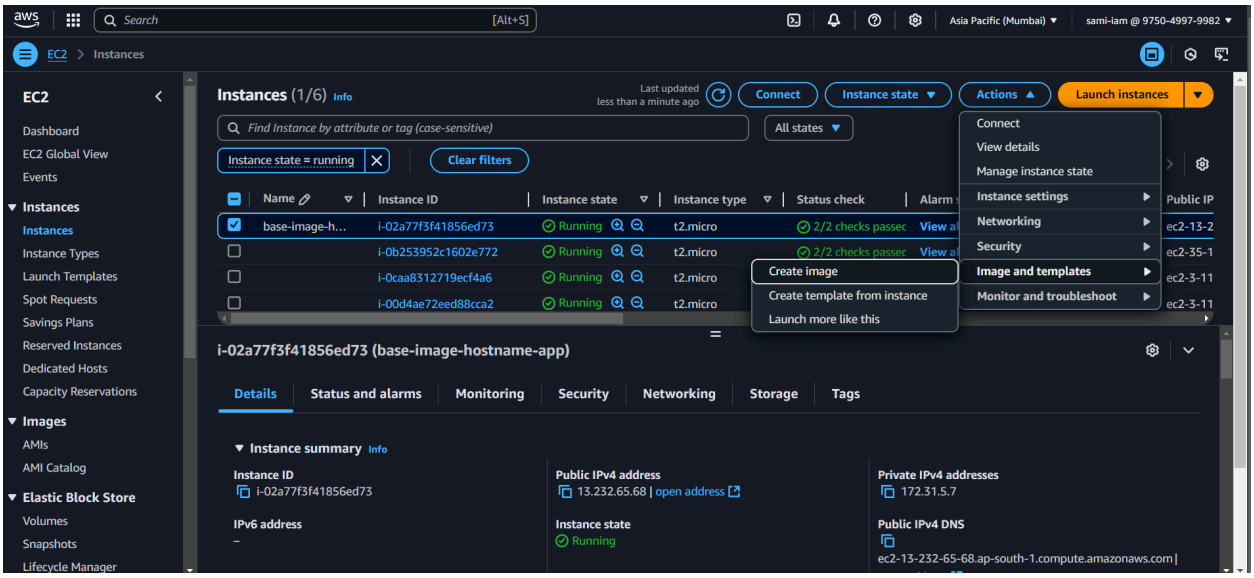
Step 1: Create an EC2 instance to create an image on it.

- create an EC2 instance manually and clone your code over there.
- install all dependences and pm2.
- run the application using pm2




```
pm2 start --interpreter /home/ubuntu/.nvm/versions/node/v22.14.0/bin/bun bin.ts
```

Step 2: Create image from the selected instance



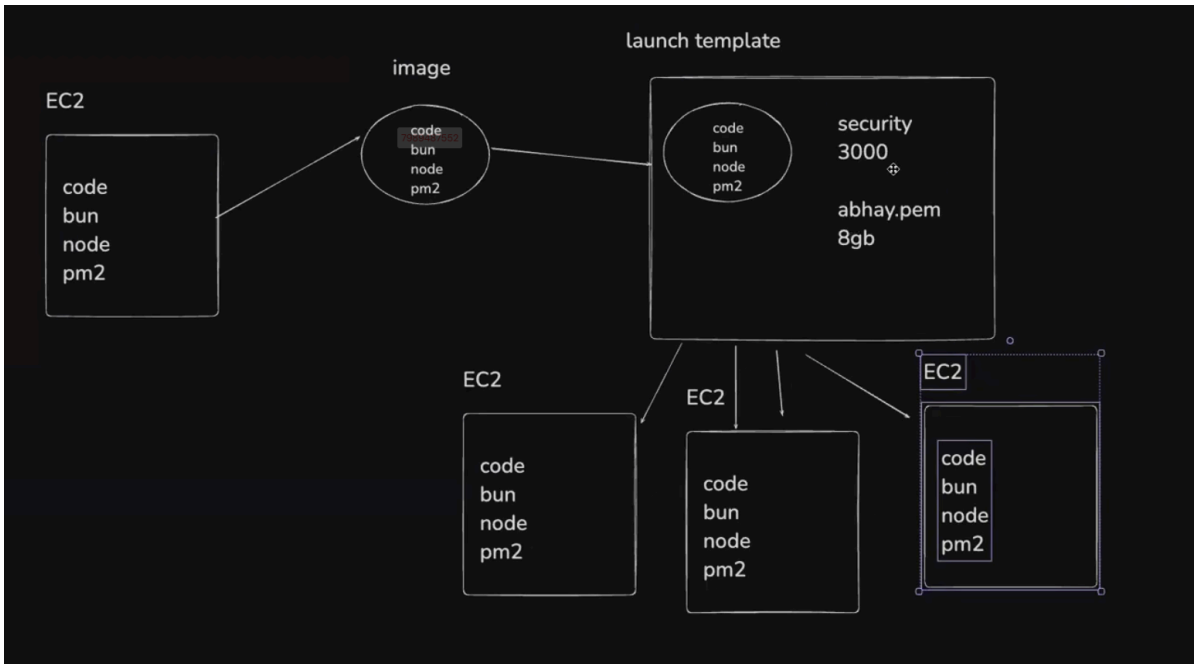
Step 3: Create a Launch Template

- Give name to the template
- Select the recently create Amazon Machine Image (AMI)
- Select instance type (t2.micro)
- Select a key-pair login (for future debugging purposes)
- Expose port 22 for ssh by creating security group from anywhere.
- Now, we need to expose the application's port (here, 3000) specifically to load balancer.
- But we would allow the inbound traffic to port 3000 from anywhere.
- In advanced details:
  - Add the following code to userdata:
  - It means, after the machine starts with the content from the image you selected, what all do you want running on the machine.

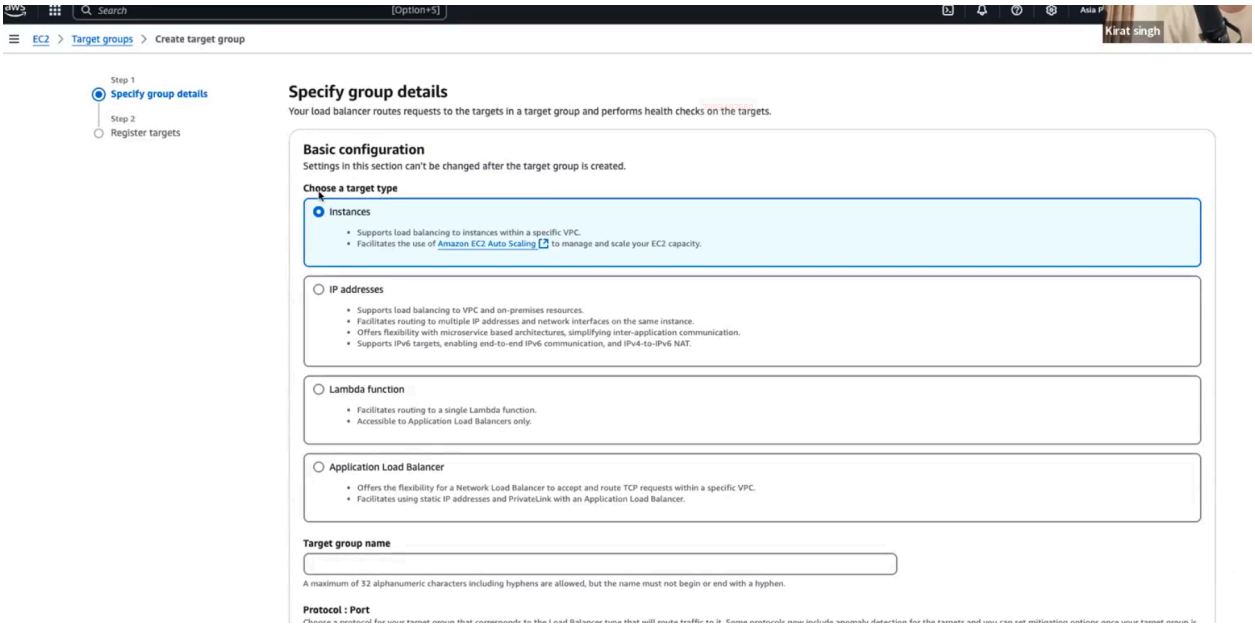
```
#!/bin/bash
cd ~/ASG
export PATH=$PATH:/home/ubuntu/.nvm/versions/node/v22.14.0/bin

# The above export command would add node.js to path
```

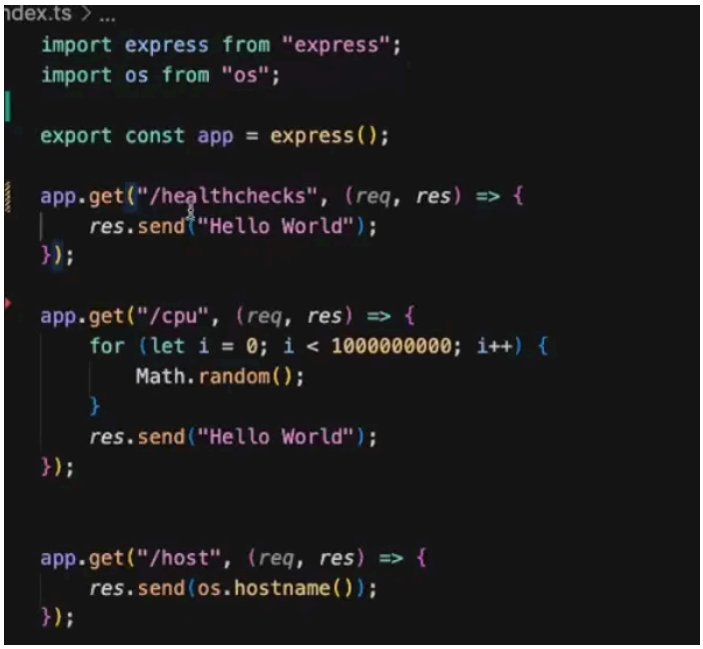
```
npm install -g pm2
pm2 start --interpreter /home/ubuntu/.nvm/versions/node/v22.14.0/bin/bun
/home/ubuntu/ASG/bin.ts
```



Step 4: Create a Target Group



- Give Target group name
- add the application port (in this case, 3000)
- add health check. This is really important in ASGs since the ASG would hit this health check endpoint to determine if the EC2 instance is up and running properly or not.
    - if the ASG recieves a 200 status code, it means the EC2 server is running fine. else, it would terminate that EC2 instance and bring up another.

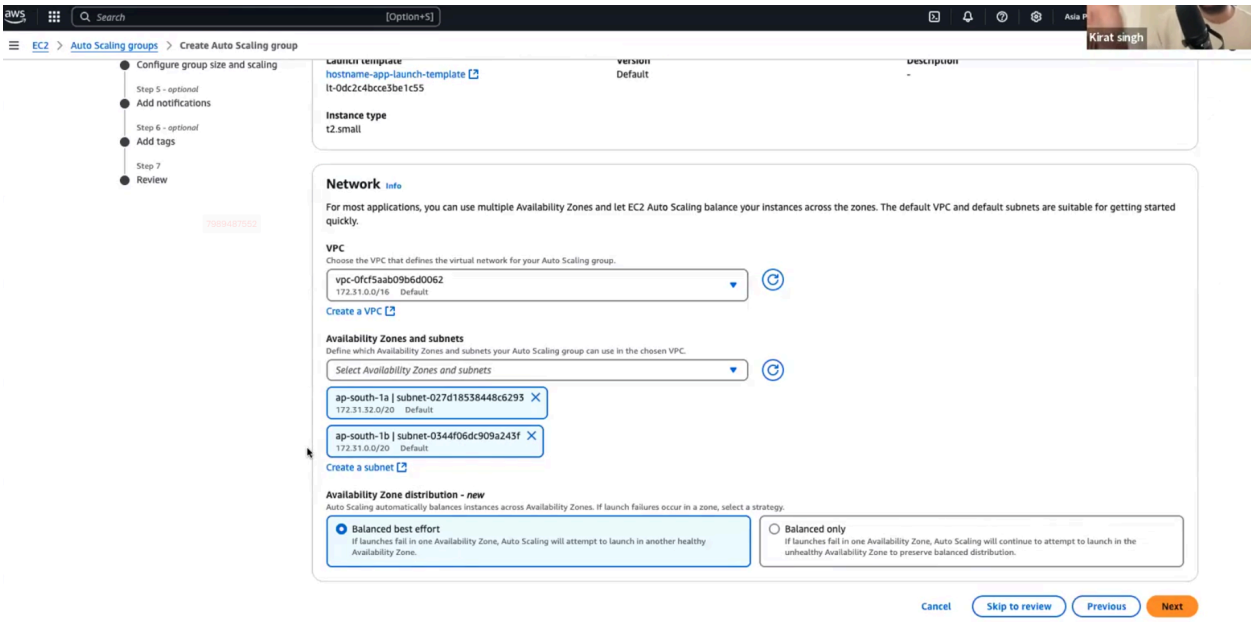


While registering targets on the next step, you should not manually select any machine since it will be managed by ASG.

- Now create the target group.

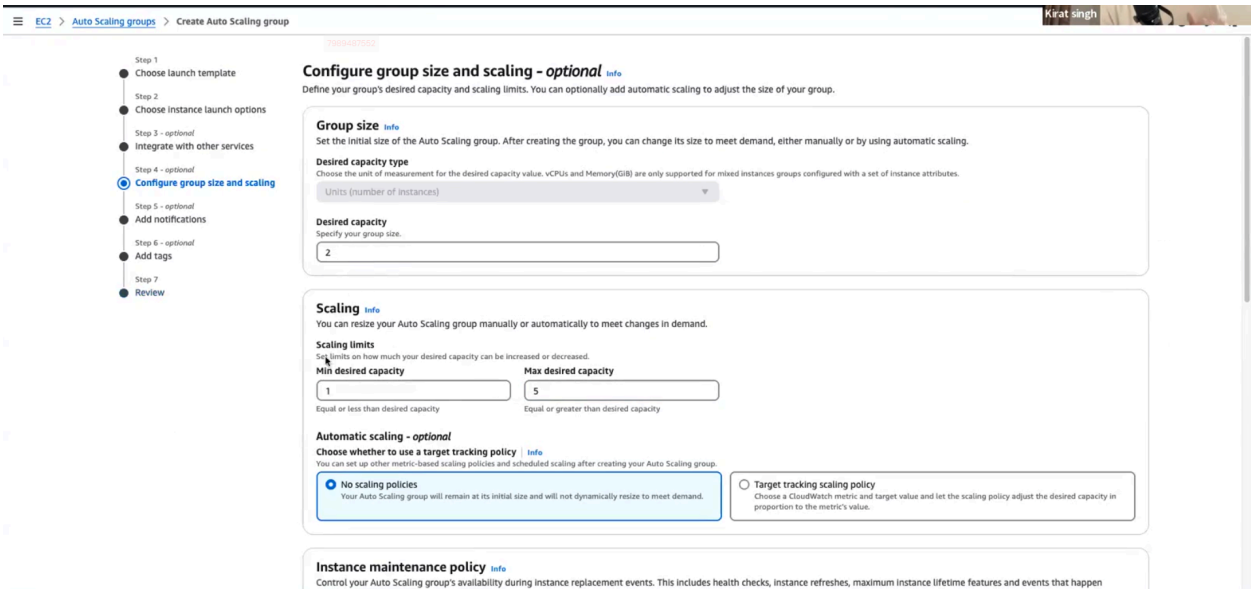
Step 5: Create a ASG

- Give your ASG a name
- select the recently created launch template
- In networks section, select as many availability zones as possible (all 3).



- Now, in the `Integrate with other services`, Attach a new load balancer.
- Select Application Load Balance
- Give your load balancer a name
- the load balancer scheme is `internet-facing`
- the load balancer should listen on port 80 for http.

In `Configure group size and scaling` section, select desired, minimum and maximum instances.



Now, finally, create the auto scaling group.

Add security groups to the load balancer

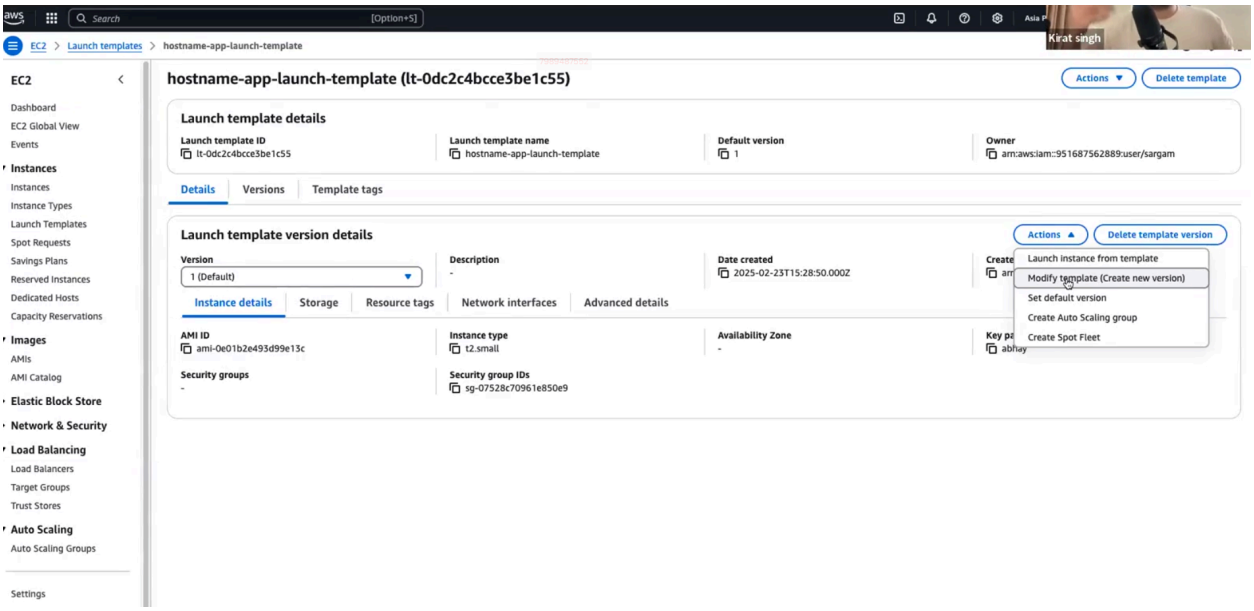
- open port 80 (HTTP) on the load balancer.

Debugging:

If the image does not run on your newly created EC2 instances via ASG, you can ssh into anyone of the EC2 instances spawned via ASG and check the logs by running the following command.

```
cat /var/log/cloud-init.log
cat /var/log/cloud-init-output.log
```

If there was an error in userdata in the launch template or you want to change security group or the instance type, you can edit it by modifying template (create new version)



- Now your template has 2 versions.
- You have to go to ASG, edit the version of the launch template to the latest version (version 2).

After Changing the version of launch template

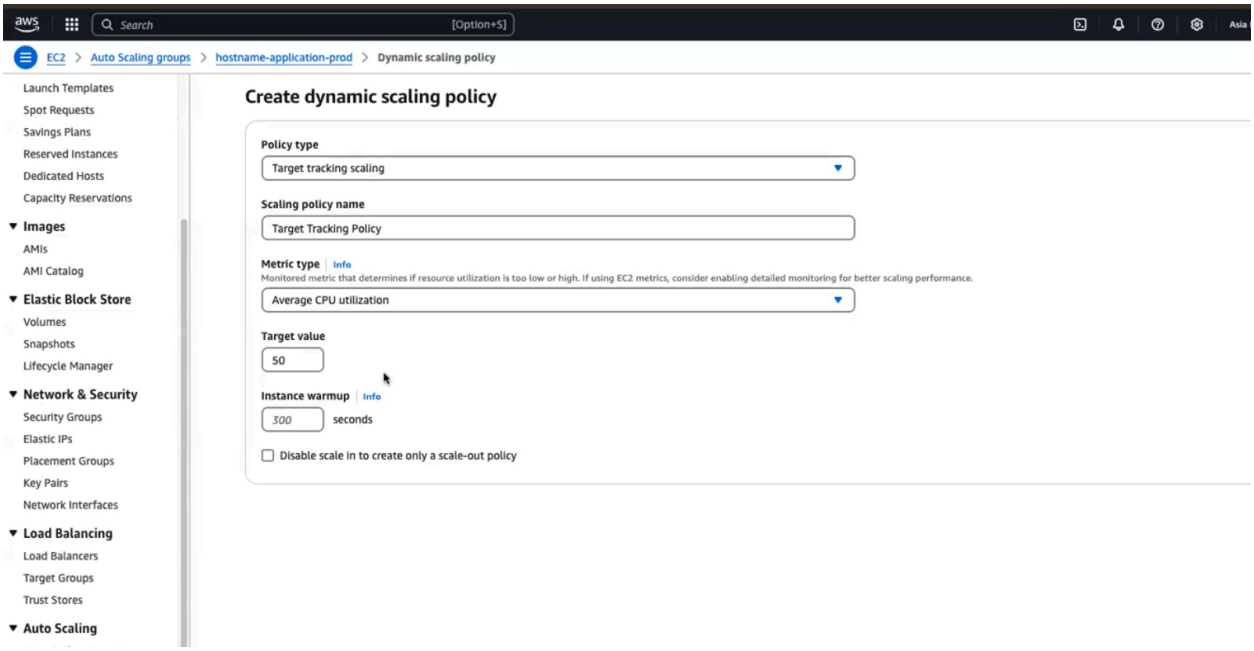
- Now, you need to terminate all the instances spawned via ASG and start the new instances.
- you can do that by simply editing the ASG's desired, minimum and maximum instances values to 0.
- After all the EC2 instances are terminated, change the values of desired, minimum and maximum instances back to normal which will spawn new EC2 instances via ASG.

Debugging tip for Load balancer

- If you get `503 error page` when you hit the load balancer URL, make sure your ASG is properly connected to the load balancer and your load balancer is connected to the target group.
- add security groups to the load balancer to open port 80 (HTTP) on the load balancer.

setup Automatic scaling

- Go to ASG and go to Automatic scaling tab
- You can setup auto scaling based on CPU usage.
    - if the CPU usage hits an average of 50% combining all the running instances on average, then a new EC2 instance will be spawned to distribute the load and bring down the CPU utilization below 50% threshold.
    - You can also setup auto scaling based on no. of incoming requests



Test ASG

- you can test your asg by hitting the resource-intensive endpoint (in our case, it is the /cpu route which run a long loop till some big number 1000000000).
- If you run multiple node.js processes from your local machine and hit this endpoint multiple times, you can see the desired instances (the no. of instances currently running) would increase.