

VPC Project

Accessing a simple Python application from private subnet

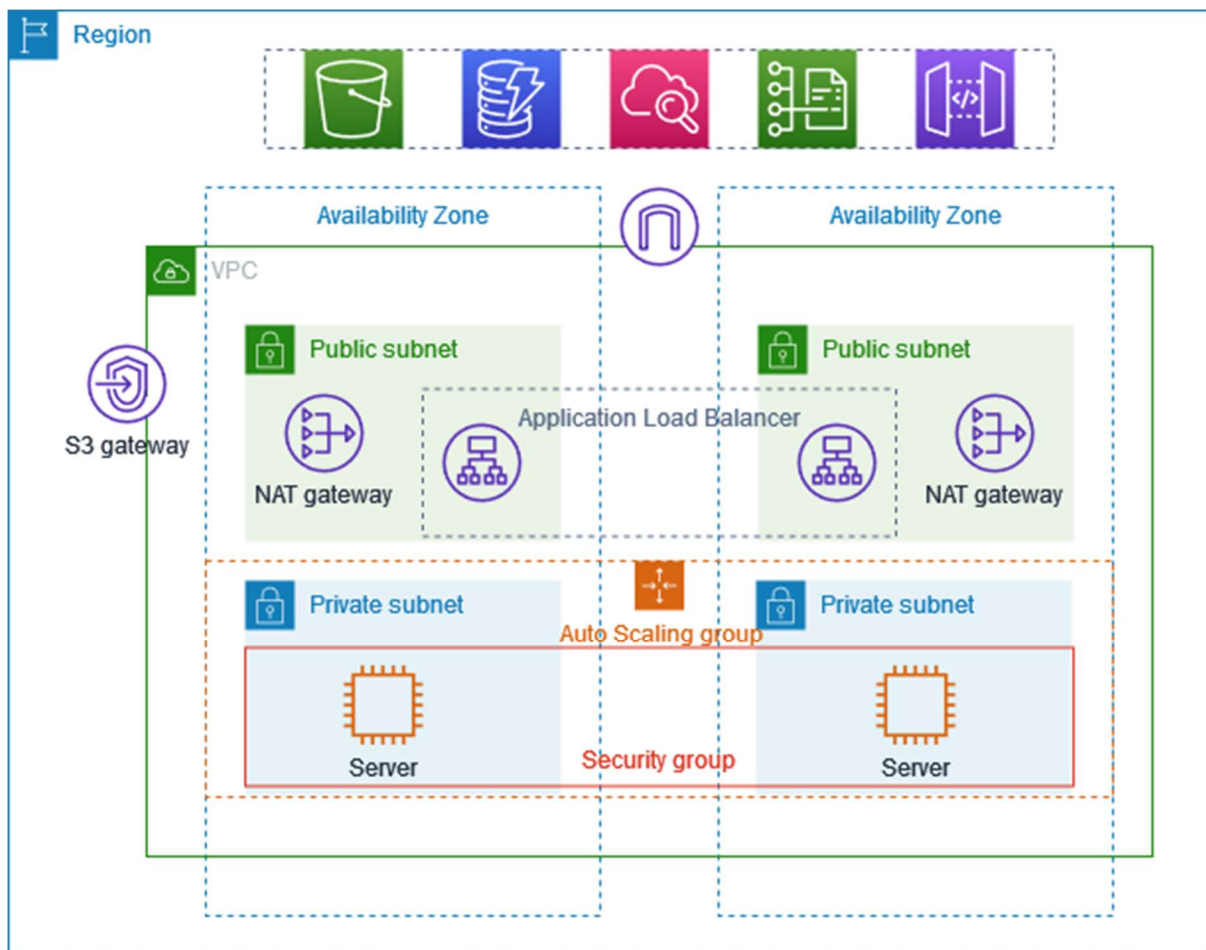
By: Shivam Joshi

<https://www.linkedin.com/in/shivamgjoshi>

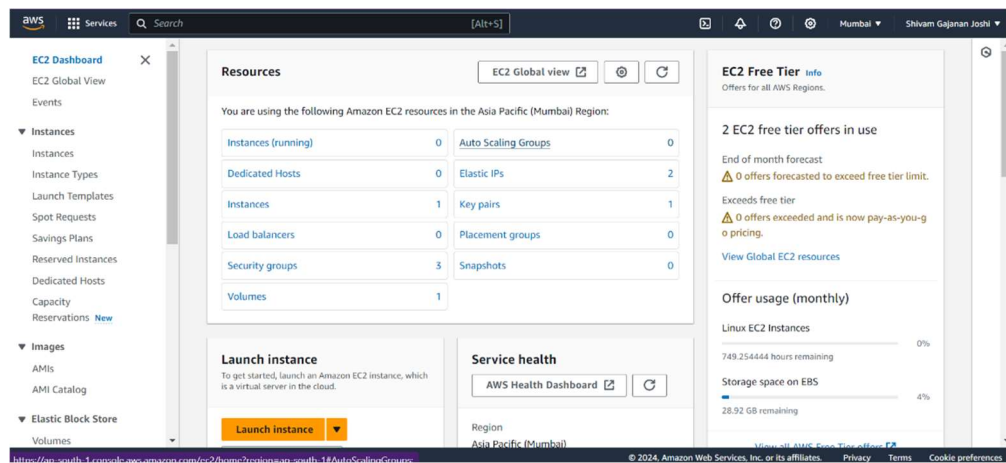
Services Used:

- Application Load Balancer
- Autoscaling groups
- VPC
- EC2

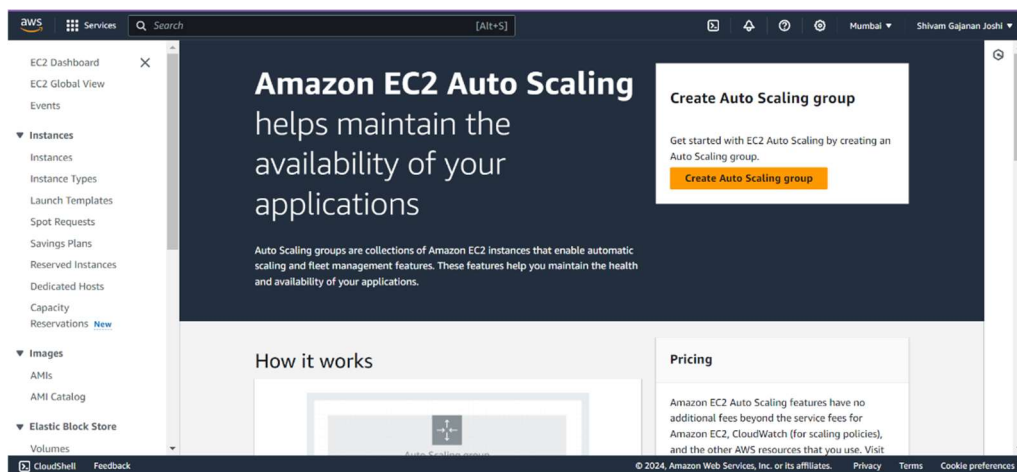
Architecture



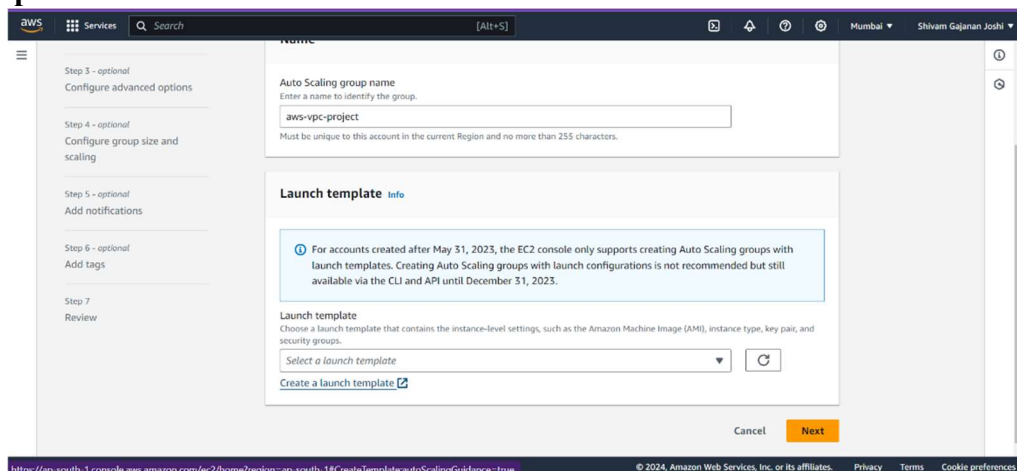
1. Go to AWS console, search EC2. Click on **Auto scaling groups**



2. Create Auto Scaling group



3. We need to specify the EC2 configuration for autoscaling group. Click on **Launch template**



4. Give name for the template and write short description.

Create launch template

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

Launch template name and description

Launch template name - *required*

aws-vpc-project-launch-template

Must be unique to this account. Max 128 chars. No spaces or special characters like %, ", @.

Template version description

Deploying a vpc project

Max 255 chars

Auto Scaling guidance [Info](#)

Select this if you intend to use this template with EC2 Auto Scaling

☒ Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► Template tags

► Source template

Summary

Software Image (AMI)

Canonical, Ubuntu, 24.04 LTS, ...[read more](#)

ami-0f58b397bc5c1f2e8

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per

Cancel **Create launch template**

5. Choose AMI as **Ubuntu**.

Quick Start

Amazon Linux, macOS, **Ubuntu**, Windows, Red Hat, SUSE Li

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

ami-0f58b397bc5c1f2e8 (64-bit (x86)) / ami-0dda4ba9a42839a4b (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Canonical, Ubuntu, 24.04 LTS, amd64 noble image build on 2024-04-23

Architecture

64-bit (x86)

AMI ID

ami-0f58b397bc5c1f2e8 **Verified provider**

Free tier eligible

Instance type [Info](#) | [Get advice](#)

Advanced

Summary

Software Image (AMI)

Canonical, Ubuntu, 24.04 LTS, ...[read more](#)

ami-0f58b397bc5c1f2e8

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per

Cancel **Create launch template**

6. Go with free tier configurations. Choose the key pair name, or create on.

Instance type [Info](#) | [Get advice](#)

Advanced

Instance type

t2.micro **Free tier eligible**

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Linux base pricing: 0.0124 USD per Hour

On-Demand Windows base pricing: 0.017 USD per Hour

On-Demand RHEL base pricing: 0.0724 USD per Hour

On-Demand SUSE base pricing: 0.0124 USD per Hour

Additional costs apply for AMIs with pre-installed software

Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name

VPC project [Create new key pair](#)

Summary

Software image (AMI)

Canonical, Ubuntu, 24.04 LTS, ...[read more](#)

ami-0f58b397bc5c1f2e8

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

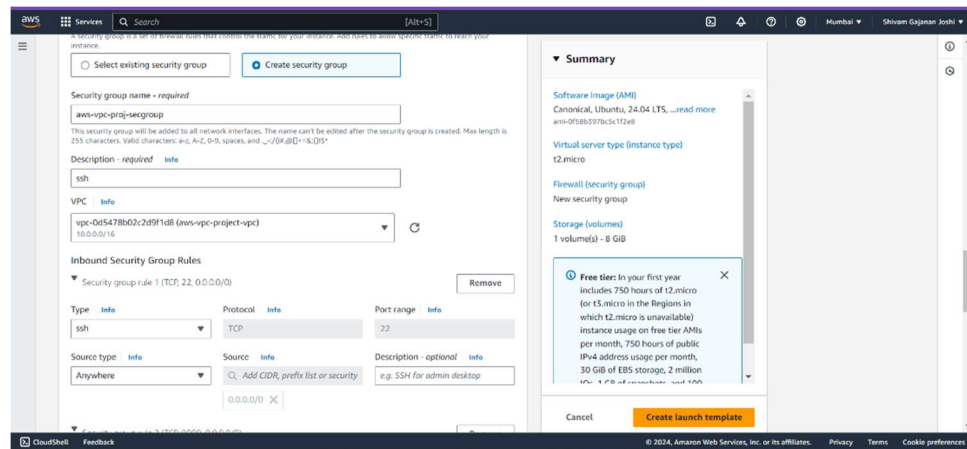
Storage (volumes)

1 volume(s) - 8 GiB

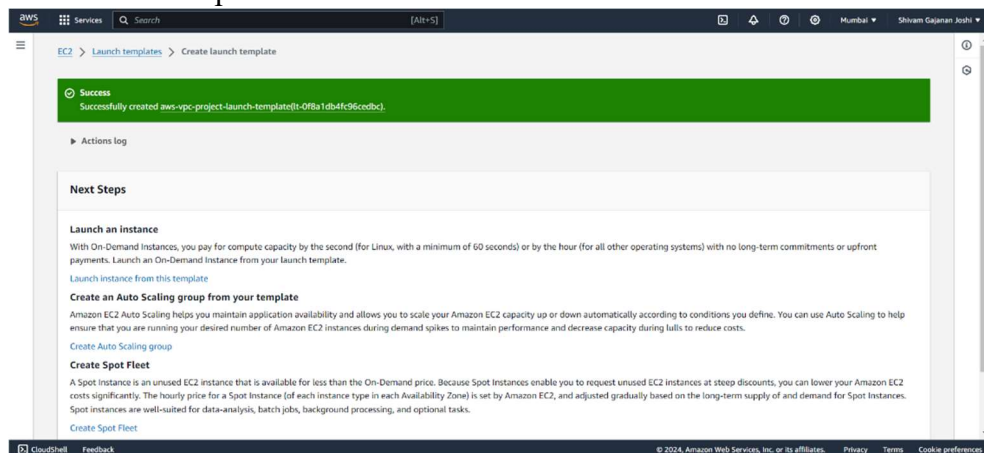
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per

Cancel **Create launch template**

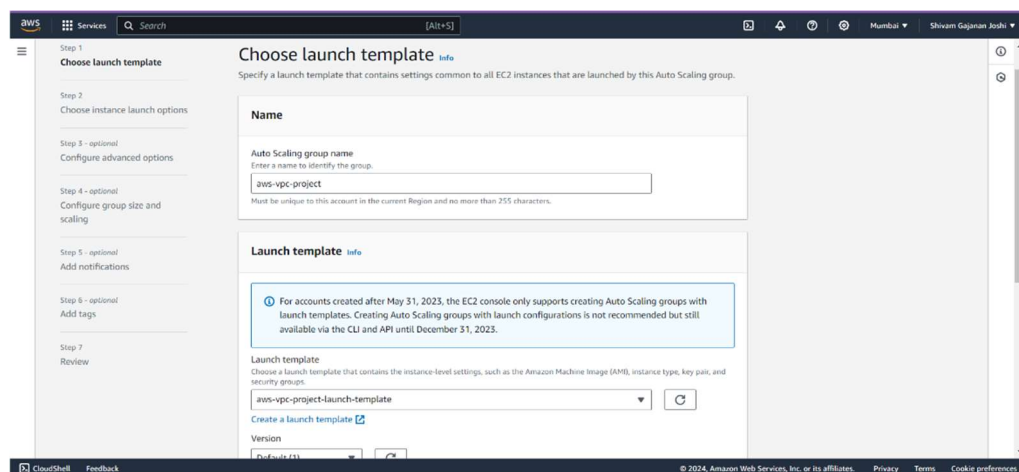
7. Click on **create security group**, and select the **VPC** created for this project. Click on edit **inbound rules**, add **SSH** rule.



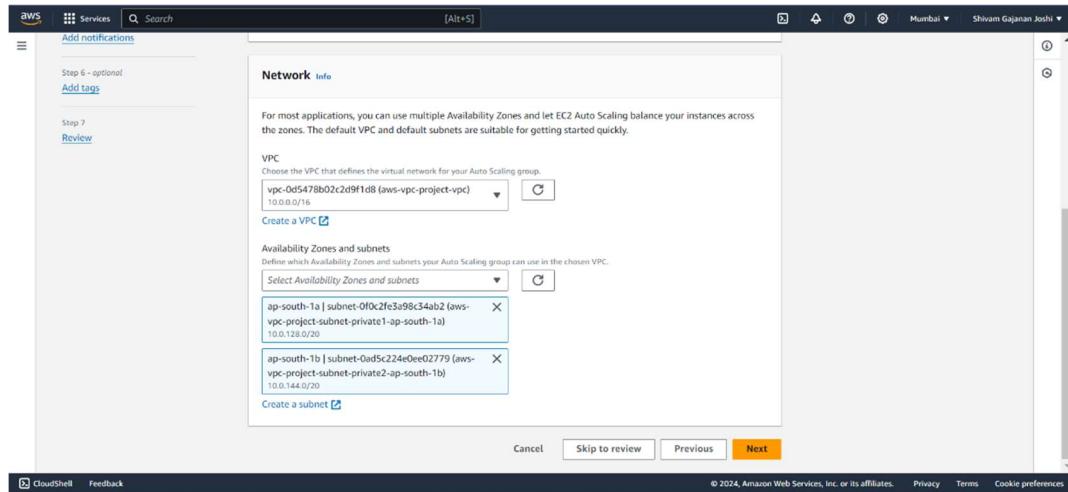
8. Click on **Launch template**.



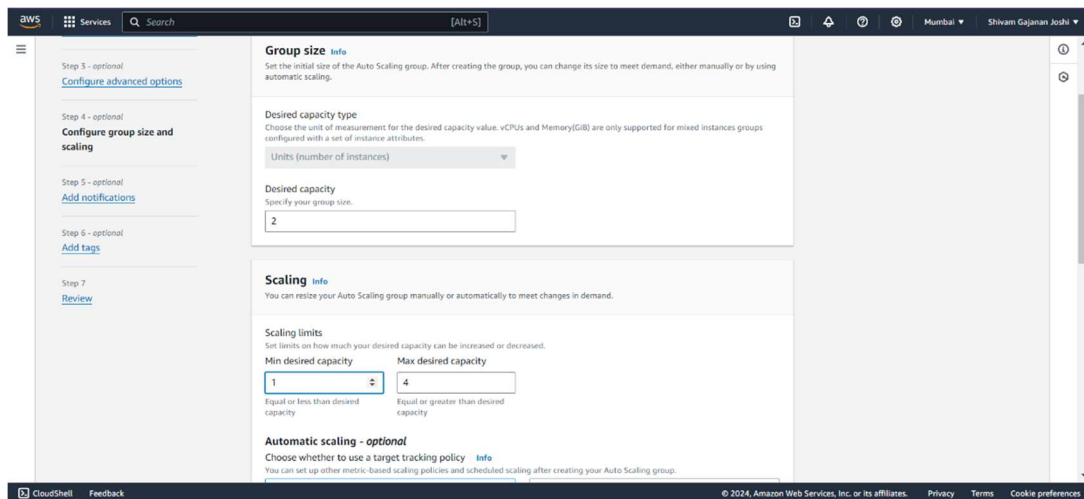
9. Once it is created go to **autoscaling groups**, and select the recently created **template**.



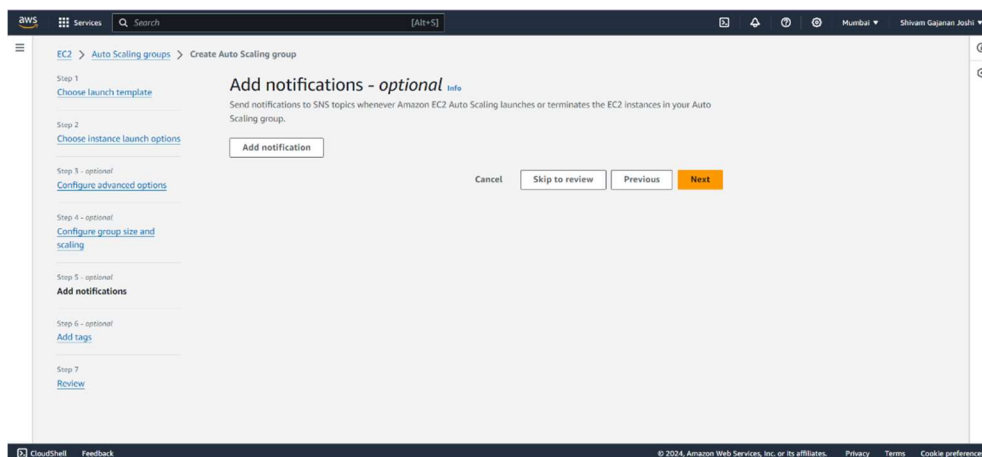
10. Under networking section, select VPC of this project and **select the 2 private subnets.**



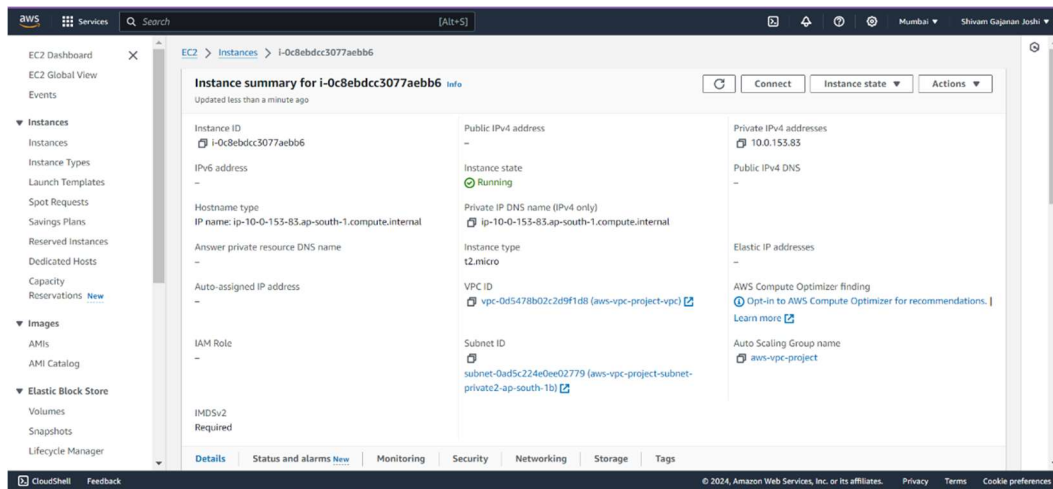
11. Under Group size, select the no. of instance you need in the start, and max capacity instance. These will reach its said limit when the traffic will be more.



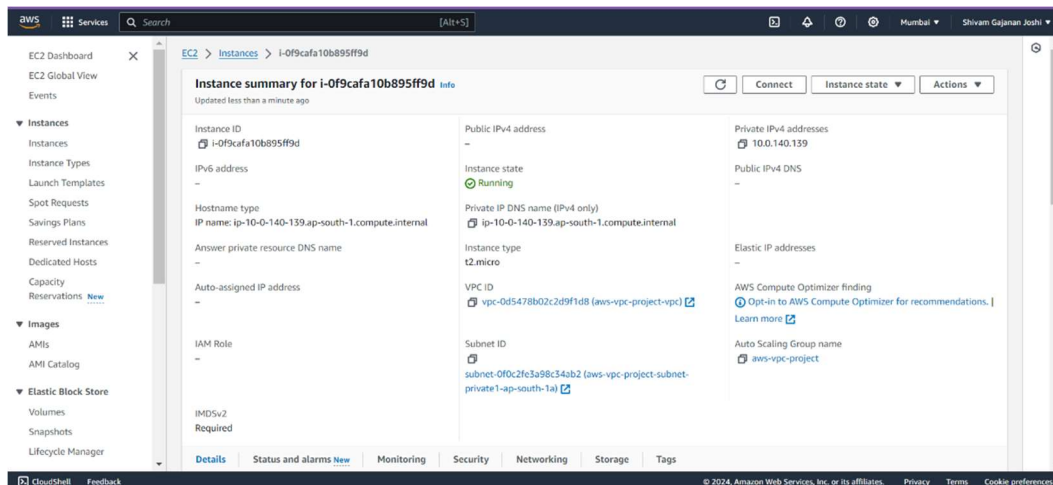
12. Go with default configurations in notifications section. Review and create.



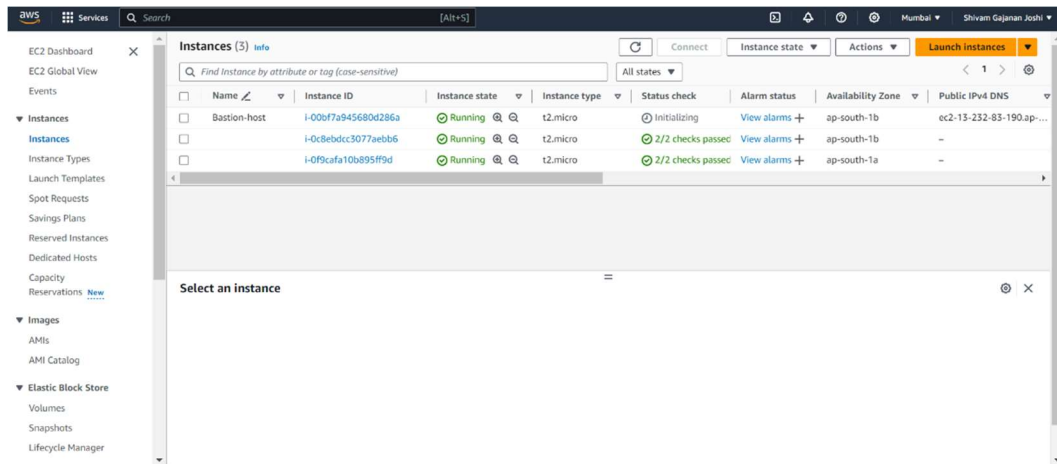
13. Check both the instances and their subnets. This instance is in south1b.



14. Check both the instances and their subnets. This instance is in south1a.



15. Create a Bastion host to access the autoscaling instances, **since they don't have a public IP address, we use a Bastion host machine.** The procedure is same as we follow in creating an EC2 instance, just a minor change in the **networking setting**, while making the configurations select the **VPC in which both the Autoscaling instances are deployed, and enable auto assign IP address option.**



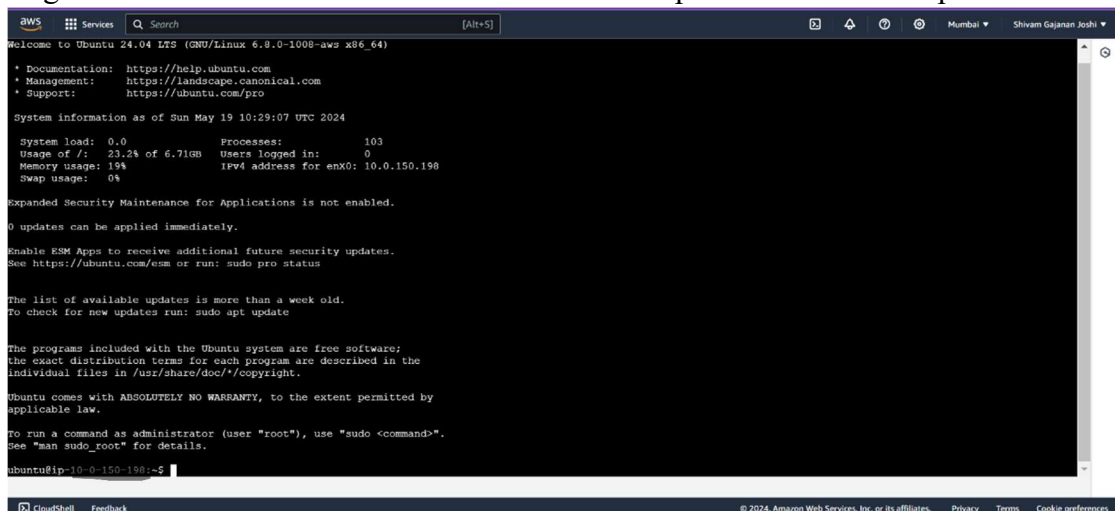
16. Now we need to copy the .pem file from our local machine to our Bastion host machine. For that open the **bash terminal(for windows user)** and type in the below command. The path of .pem may vary, change it accordingly.

\$ scp -i <path of .pem file> ubuntu@<publicip of Bastion>:/home/ubuntu

```
HP@SHIVAM MINGW64 ~
$ scp -i "C:\Users\HP\Downloads\awsprodekeypair.pem" "C:\Users\HP\Downloads\awsprodekeypair.pem" ubuntu@65.2.136.17:/home/ubuntu
C:\Users\HP\Downloads\awsprodekeypair.pem                                100% 1678    49.6KB/s   00:00

HP@SHIVAM MINGW64 ~
$
```

17. Login to the Bastion host machine and check if the .pem file has been copied or not.



18. Now we need to create a sample html file.

```
aws Services Search [Alt+S]
ubuntu@ip-10-0-150-198:~$ cat index.html
<!DOCTYPE html>
<html>
<head>
<title>My python application to demonstrate apps in private subnet</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

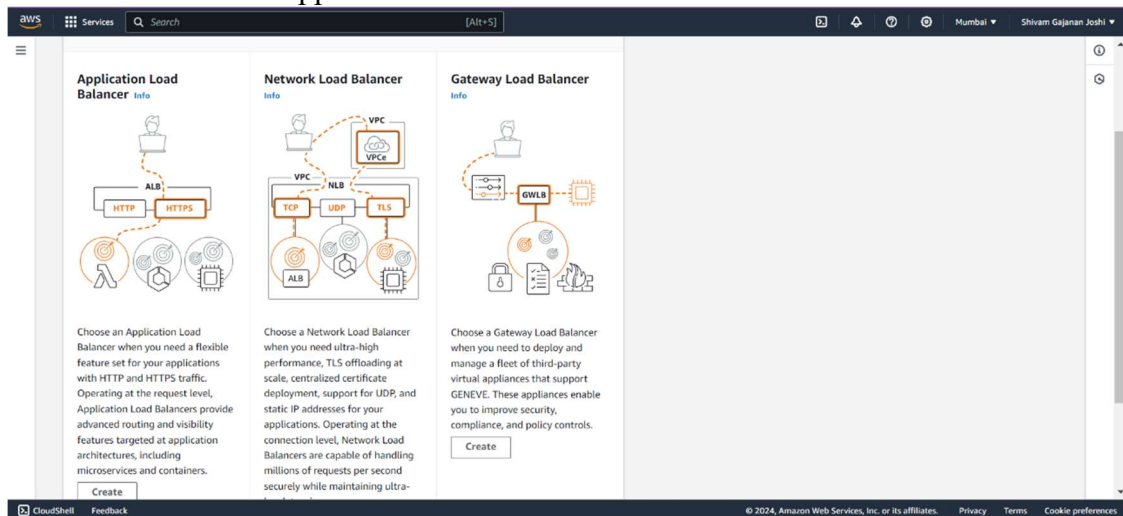
</body>
</html>
ubuntu@ip-10-0-150-198:~$
```

19. Run the application using above command

\$ python3 -m http.service <port number>

```
aws Services Search [Alt+S]
ubuntu@ip-10-0-150-198:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

20. It's time to create an Application Load Balancer



21. Follow the below steps

The screenshot shows the 'How Application Load Balancers work' page in the AWS Management Console. The 'Basic configuration' section is active. The 'Load balancer name' field contains 'aws-prod-eg-lb'. The 'Scheme' is set to 'Internet-facing'. The 'IP address type' is set to 'IPv4'.

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
aws-prod-eg-lb
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info
Scheme can't be changed after the load balancer is created.

☒ **Internet-facing**
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)

☐ **Internal**
An internal load balancer routes requests from clients to targets using private IP addresses. Compatible with the IPv4 and Dualstack IP address types.

IP address type Info
Select the type of IP addresses that your subnets use. Public IPv4 addresses have an additional cost.

☒ **IPv4**
Includes only IPv4 addresses.

☐ **Dualstack**
Includes IPv4 and IPv6 addresses.

☐ **Dualstack without public IPv4**
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with Internet-facing load balancers only.

22. Select the VPC created for this project.

The screenshot shows the 'Network mapping' section of the AWS Management Console. The 'VPC' dropdown is set to 'aws-production-eg-vpc'. Two subnets are selected: 'ap-south-1a (aps1-az1)' and 'ap-south-1b (aps1-az3)'.

Network mapping Info
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC Info
Select the virtual private cloud (VPC) for your targets or you can [create a new VPC](#). Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

aws-production-eg-vpc
vpc-02aa02efe8862197f
IPv4 VPC CIDR: 10.0.0.0/16

Mappings Info
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

☒ **ap-south-1a (aps1-az1)**

Subnet
subnet-051430e17a00b95dc
aws-production-eg-subnet-public1-ap-south-1a

IPv4 address
Assigned by AWS

☒ **ap-south-1b (aps1-az3)**

Subnet
subnet-0abd501430c373733
aws-production-eg-subnet-public2-ap-south-1b

IPv4 address
Assigned by AWS

23. In the security groups, add rule to allow traffic via port 8000 (port mentioned in previous step).

The screenshot shows the 'Security groups' and 'Listeners and routing' sections of the AWS Management Console. The 'Security groups' dropdown is set to 'aws-prod-eg'. The 'Listeners and routing' section shows a listener for HTTP on port 80, with the default action set to 'Forward to'.

Security groups Info
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

Security groups
Select up to 5 security groups

aws-prod-eg
sg-02869605db4a34b91 VPC: vpc-02aa02efe8862197f

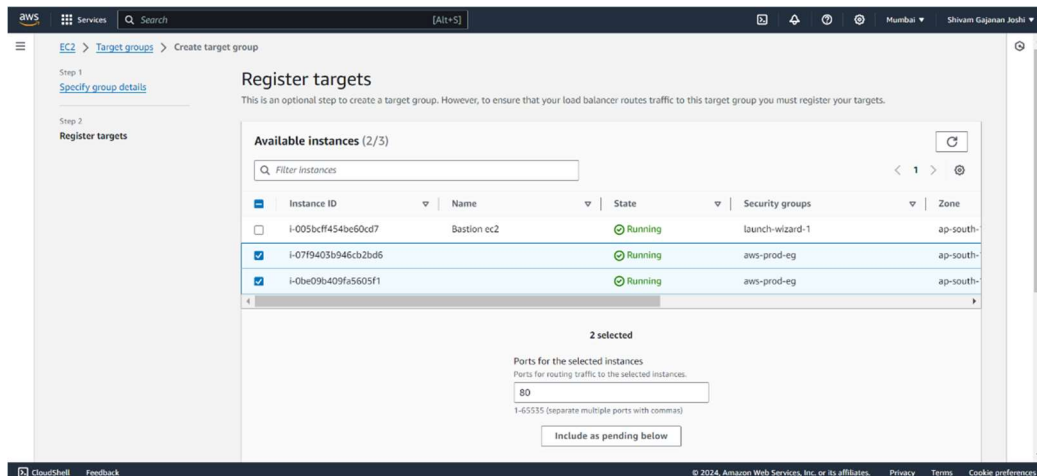
Listeners and routing Info
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

Listener HTTP:80 Remove

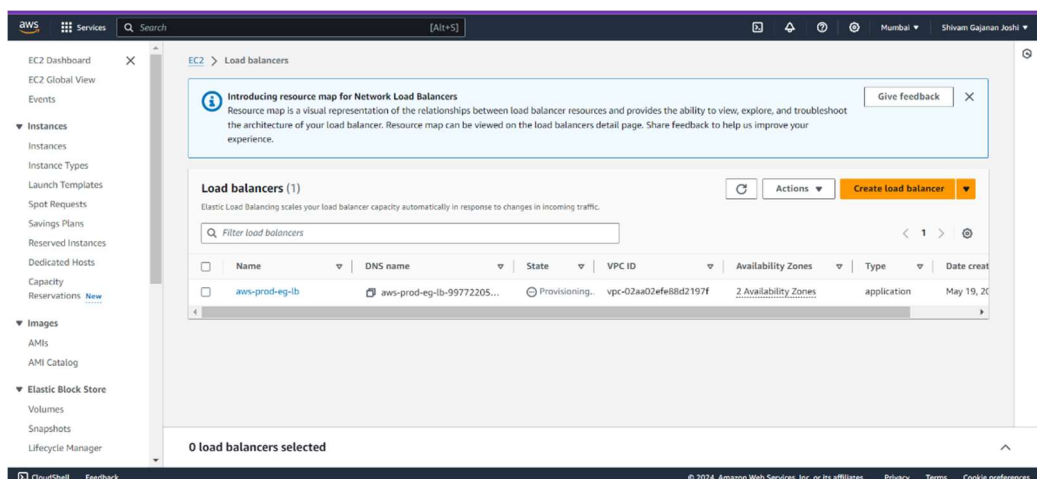
Protocol: HTTP Port: 80 Default action: Forward to Select a target group
1-85535 [Create target group](#)

Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

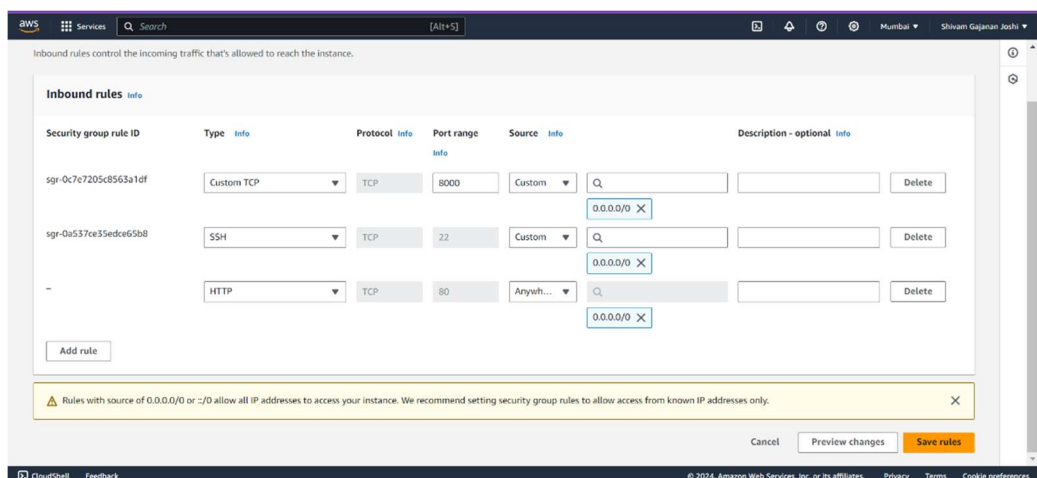
24. Now add the autoscaling machines into the target group and click **Include as pending** below.



25. Create load balancer.



26. If you face issue with port number or access denied, change the security group rules.



27. Access the application using the LB DNS. It should display the output of HTML file.

My First AWS PROJECT to demonstrate apps in private subnet

NOTE: If you see 502 Bad Gateway, it is because aws shows the successful output of healthy instances.

Please clean up your resources to save cost.