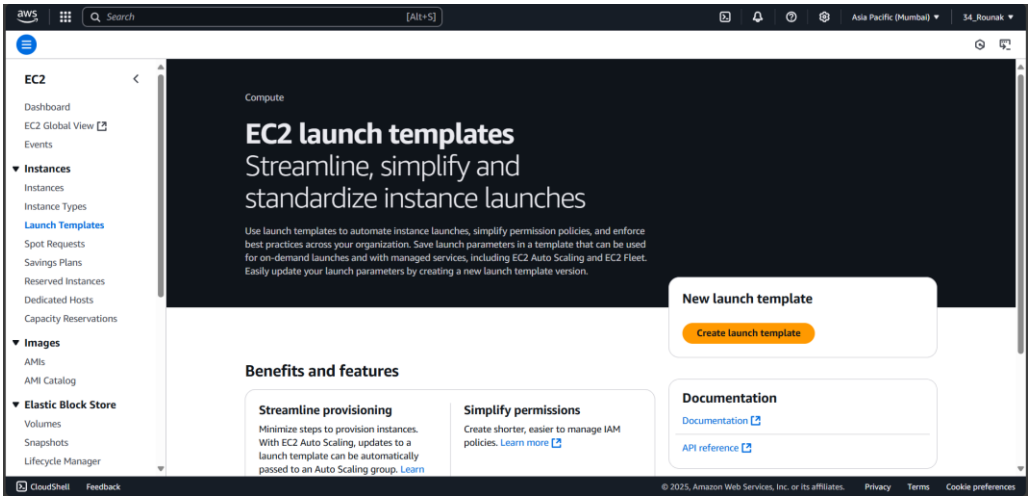
Assignment number: 11

Problem definition: Build scaling plans in AWS that balance the load on different EC2 instances.

Step 1: Sign in to your AWS account as the root user.

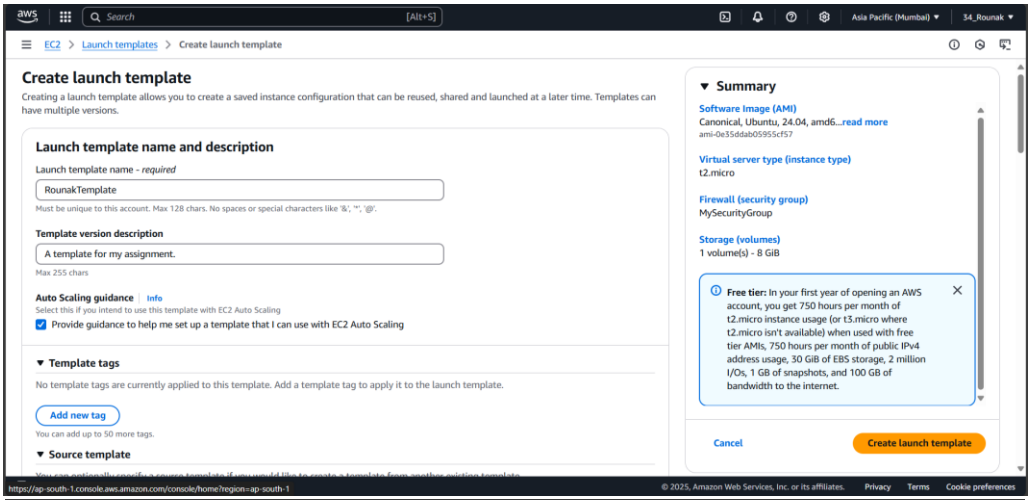
Step 2: Log in to the AWS Management Console, use the search bar to search for "EC2," and click on the EC2 service. Under **Instances**, select **Launch Templates** and click **Create launch template.**

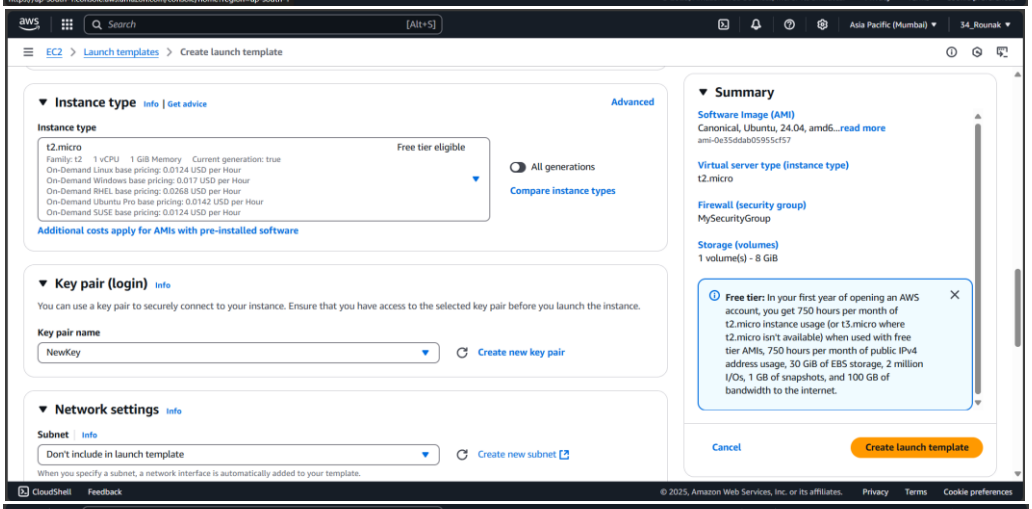


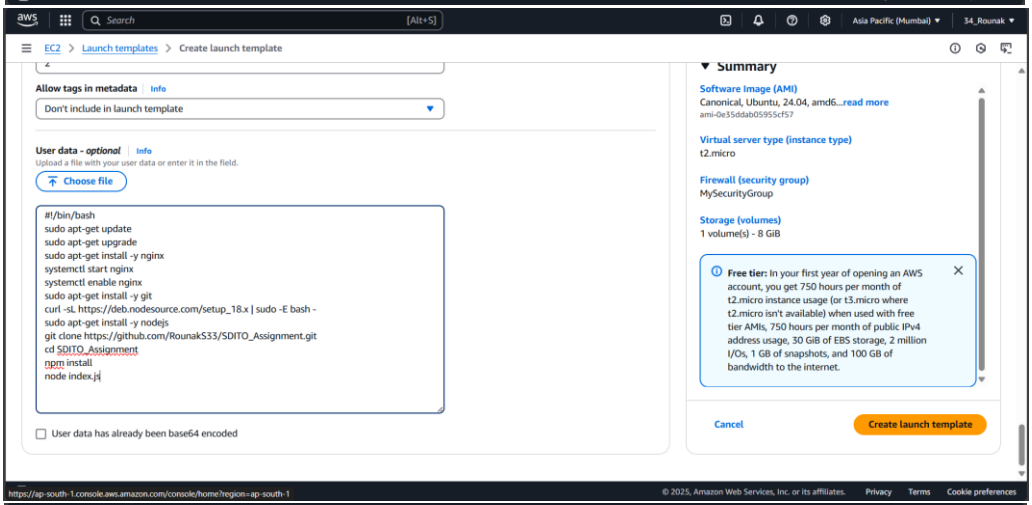
Step 3: Enter **Name** and **Description. Under Auto Scaling guidance,** check **Provide guidance to help me set up a template that I can use with EC2 Auto Scaling. Under Quick Start,** choose your **OS AMI** as **Ubuntu** and **Instance type** as **t2.micro** (free-tier)**.** Under **Key pair (login),** choose **Choose an existing key pair.** Under **Network settings, Security group,** choose **Select existing security group** and select a security group that allows inbound port 4000**.** Expand **Advanced details,** scroll to **User data,** and paste the following commands in sequential order:

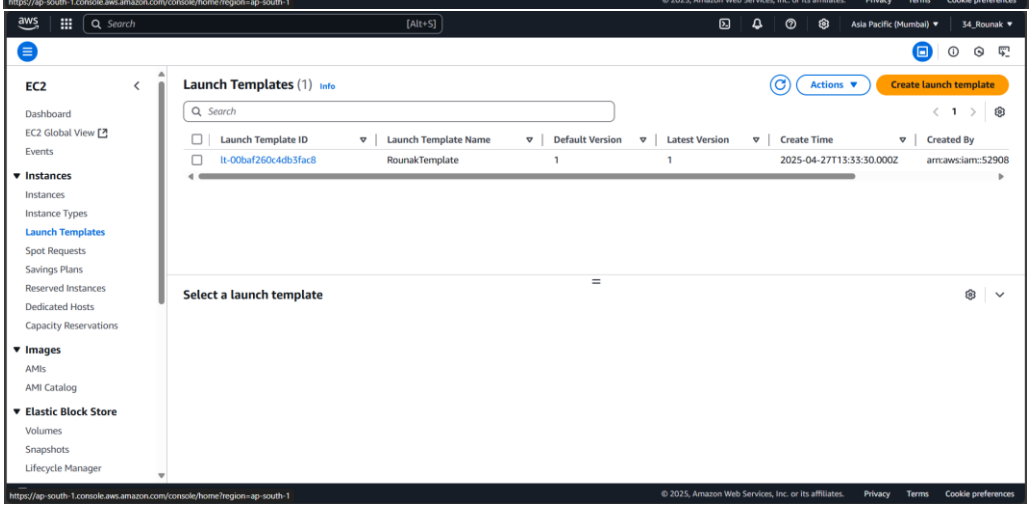
```
#!/bin/bash
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install -y nginx
systemctl start nginx
systemctl enable nginx
sudo apt-get install -y git
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
git clone https://github.com/RounakS33/SDITO_Assignment.git
cd SDITO_Assignment
npm install
node index.js
```

Click **Create launch template**.

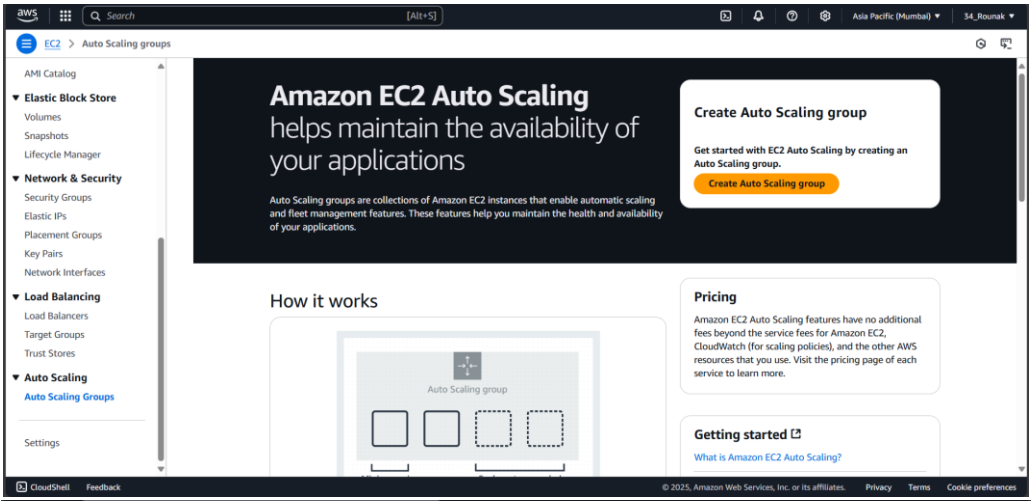

Create launch template
Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.
Launch template name and description
Launch template name - required
RounakTemplate
Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.
Template version description
A template for my assignment.
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
No template tags are currently applied to this template. Add a template tag to apply it to the launch template.
Add new tag
You can add up to 50 more tags.
Source template
Summary
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e35ddab05955cf57
Virtual server type (instance type)
t2.micro
Firewall (security group)
MySecurityGroup
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Create launch template

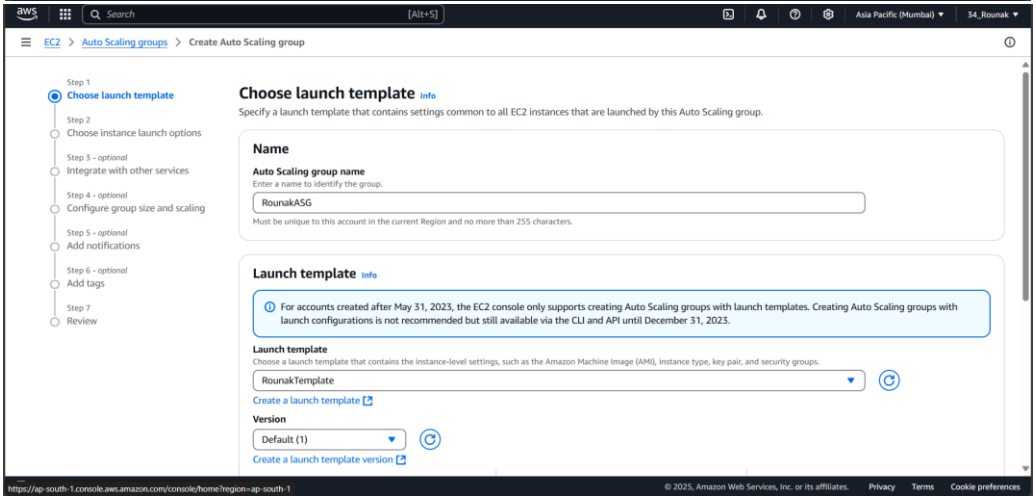

Instance type | Info | Get advice
Advanced
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0142 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour
All generations
Compare instance types
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
NewKey
Create new key pair
Network settings | Info
Subnet | Info
Don't include in launch template
Create new subnet
When you specify a subnet, a network interface is automatically added to your template.
Summary
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e35ddab05955cf57
Virtual server type (instance type)
t2.micro
Firewall (security group)
MySecurityGroup
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Create launch template


Allow tags in metadata | Info
Don't include in launch template
User data - optional | Info
Upload a file with your user data or enter it in the field.
Choose file
#!/bin/bash
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install -y nginx
systemctl start nginx
systemctl enable nginx
sudo apt-get install -y git
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
git clone https://github.com/RounakS33/SDITO_Assignment.git
cd SDITO_Assignment
npm install
node index.js
User data has already been base64 encoded
Summary
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e35ddab05955cf57
Virtual server type (instance type)
t2.micro
Firewall (security group)
MySecurityGroup
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Create launch template

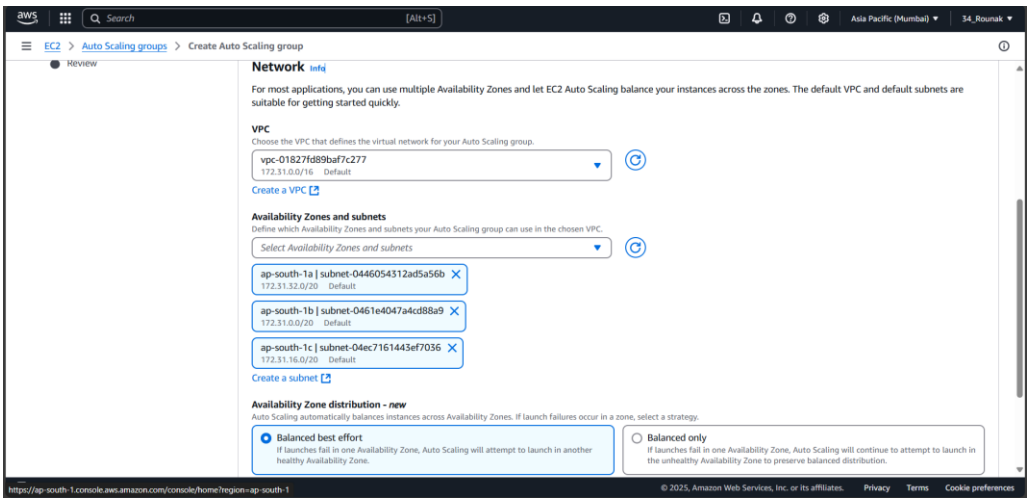

EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Launch Templates (1) Info
Actions
Create launch template
Launch Template ID
Launch Template Name
Default Version
Latest Version
Create Time
Created By
lt-00baf260c4db3fac8
RounakTemplate
1
1
2025-04-27T13:33:30.000Z
arn:aws:iam::52908
Select a launch template

Step 4: In EC2 console, under “**Auto Scaling Groups”** click “**Create Auto Scaling group”.** Enter a name (e.g. RounakASG) and select the launch template you just created. Click **Next**.

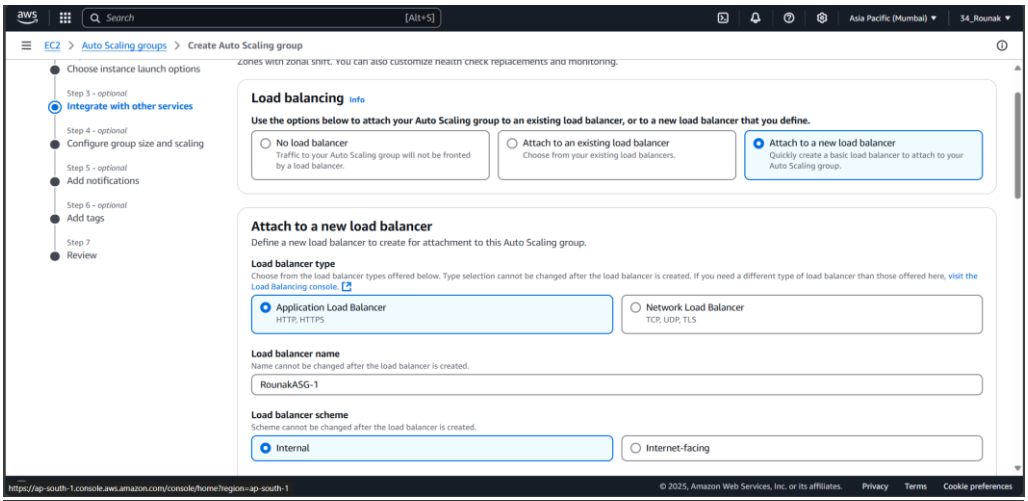


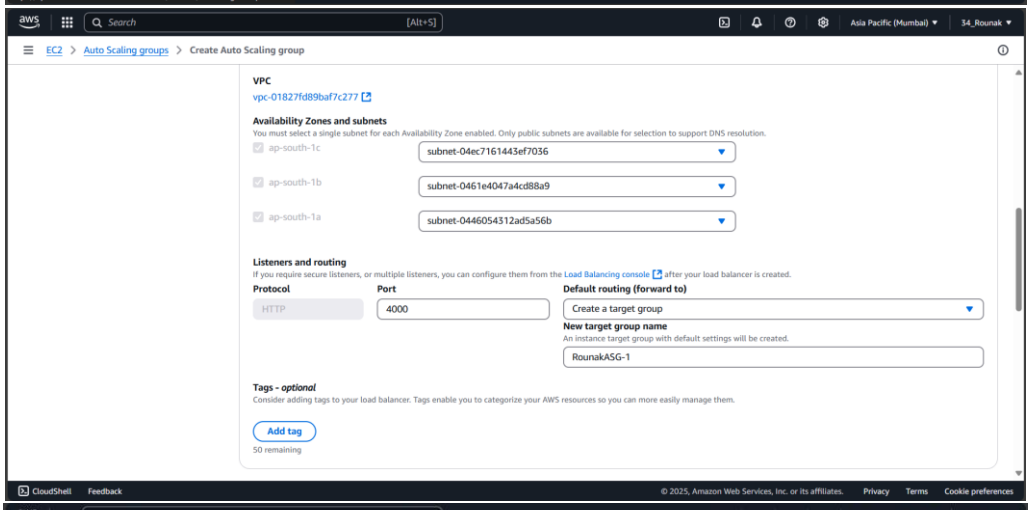


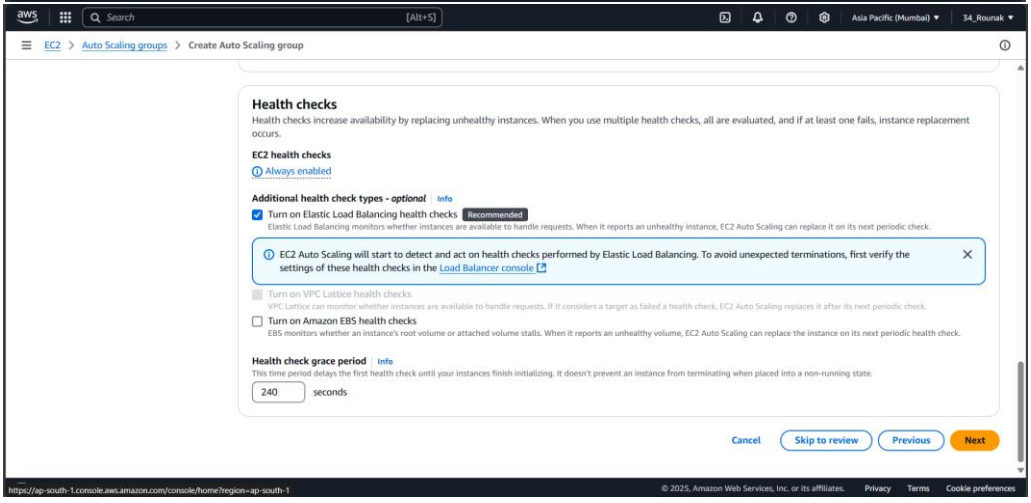
Step 5: Choose your VPC and at least two subnets in different Availability Zones for high availability. Click **Next**.



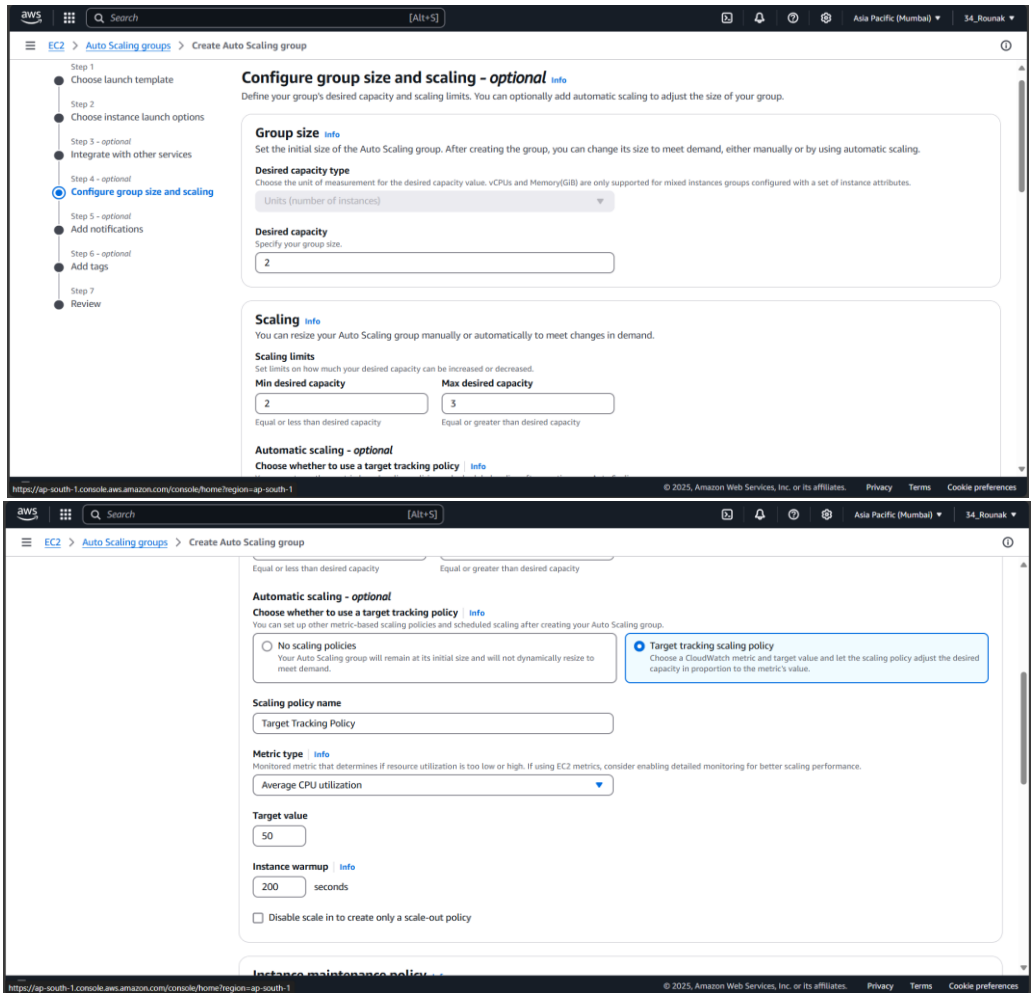
Step 6: Under **Load balancing**, choose **Attach a network load balancer**. Under **Attach to a new load balancer**, choose **Load balancer type** as **Application Load Balancer** and **Load balancer** scheme as **Internal**. When defining the **target group**, set protocol HTTP and port **4000**. Under **Health checks** check the box **Turn on Elastic Load Balancing health checks** and set **Health check grace period** to 240 s**.** Click on **Next.**



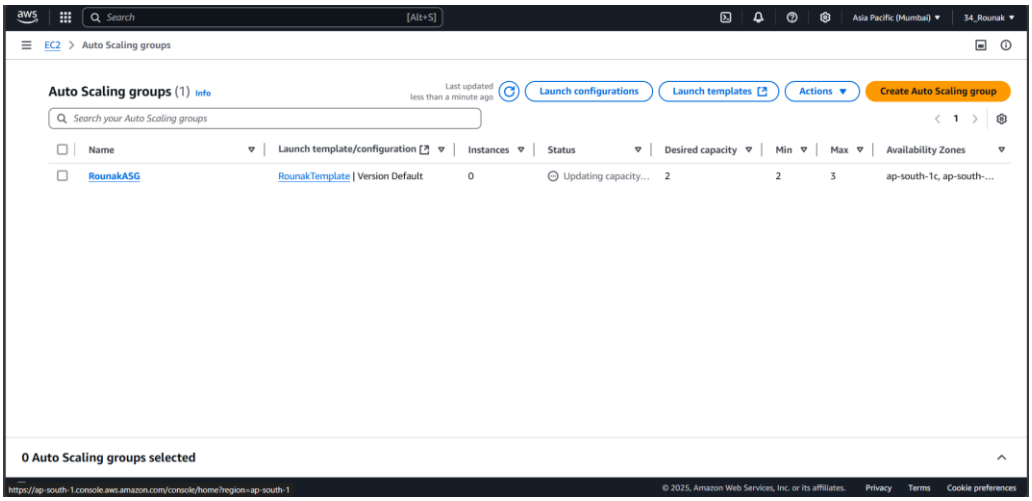


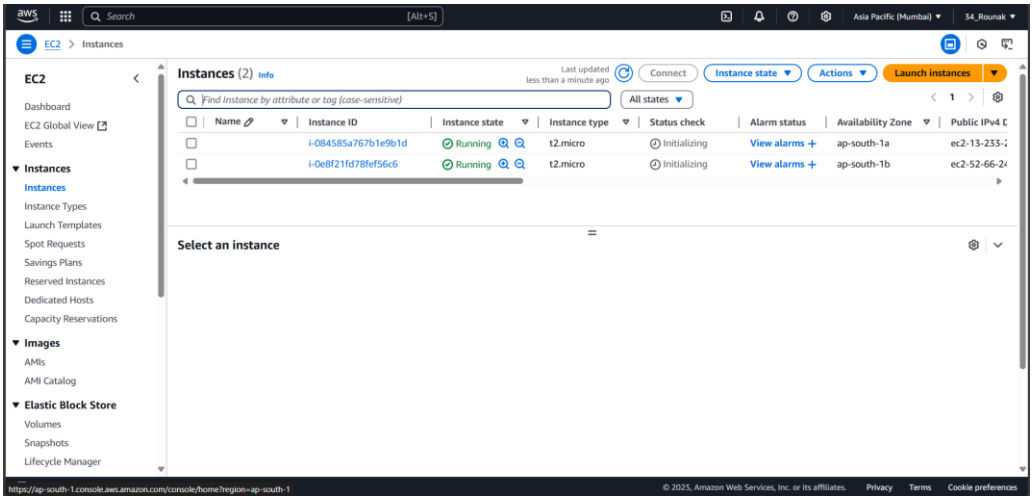


Step 7: Under **Group size** set **Desired capacity** as 2. Under **Scaling** set **Min desired capacity** as 2 and **Max desired capacity** as 3. Under **Automatic scaling**, choose **Target tracking policy**, set average CPU target to **50%**, and **Instance warm-up** to **200 s. Click Next.**

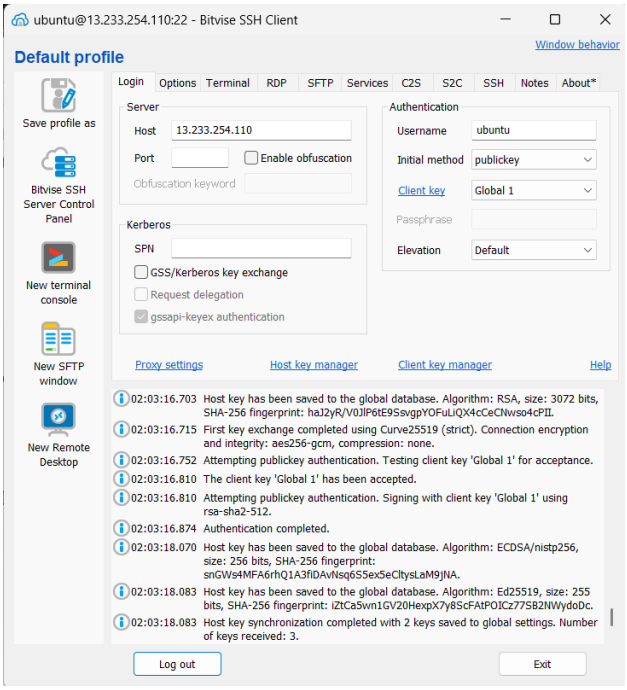


Step 8: Skip Steps 5–6 by clicking **Next**, then in Step 7 click **Create Auto Scaling group**. In **EC2 → Instances**, you'll now see two new instances launched by the ASG.





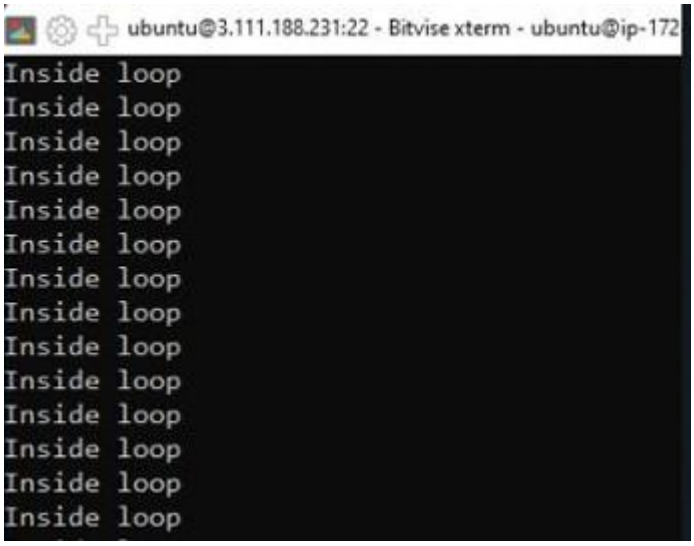
Step 9: Note the **Public IPv4** of one instance. Open the **Bitvise SSH Client**. In the main window, paste the **Public IPv4 address** into the **Host** field. Set **Username** to "ubuntu". From the **Client Key Manager** dropdown, select the imported key. Click "**Log in"** and then "**Accept & Save"** to establish the connection.



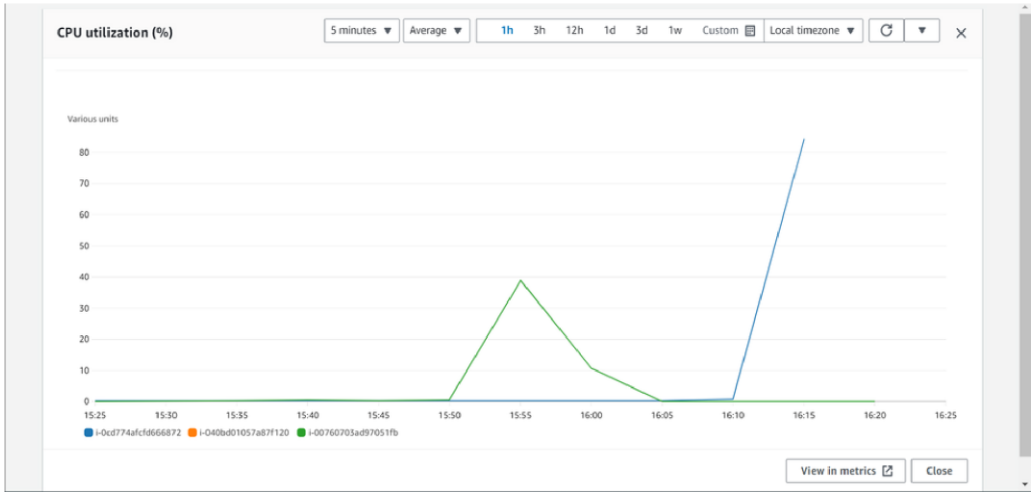
Step 8: In Bitvise SSH Client, click on "**New Terminal Console"**. Run the command "sudo nano infi.sh" to create a CPU-stress script:

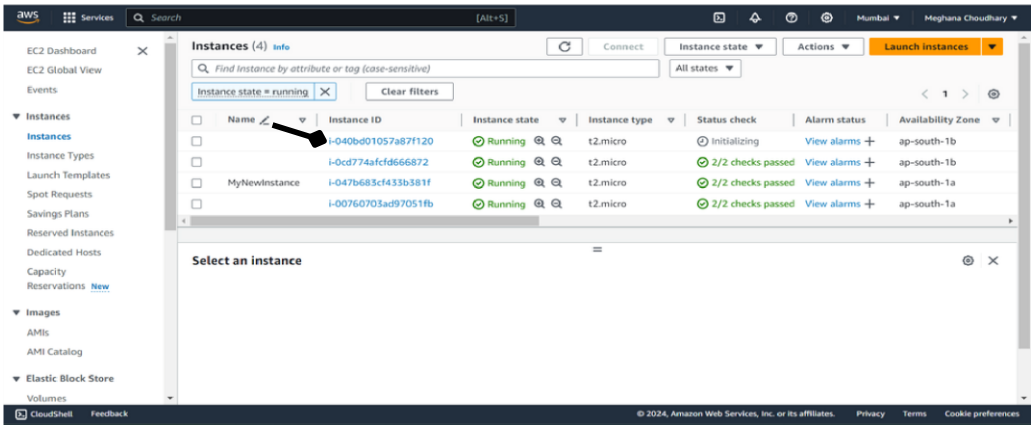
```
#!/bin/bash
while(true) do
   echo "Inside Loop"
done
```

This infinite loop will drive CPU utilization upward. Run "sudo chmod +x infi.sh" to make the script executable. Run the script using "sh infi.sh". An infinite loop starts to run, stressing the cpu of the instance.



Step 9: In the EC2 console under **Auto Scaling Groups**, monitor the **Instances** tab or view the **CPU utilization** graph. Once average CPU across the two instances exceeds **50%**, the target-tracking policy triggers the ASG to launch a third instance automatically. The new instance registers with the ALB, and traffic is rebalanced across all healthy instances.





Conclusion:

By combining an EC2 Launch Template, an Auto Scaling Group with target-tracking policies, and an Application Load Balancer, you've created a self-healing, elastic architecture that maintains performance under varying load. The infinite-loop CPU test validated that when utilization exceeds 50%, the ASG automatically adds capacity, and the ALB seamlessly distributes traffic—achieving dynamic load balancing without manual intervention.