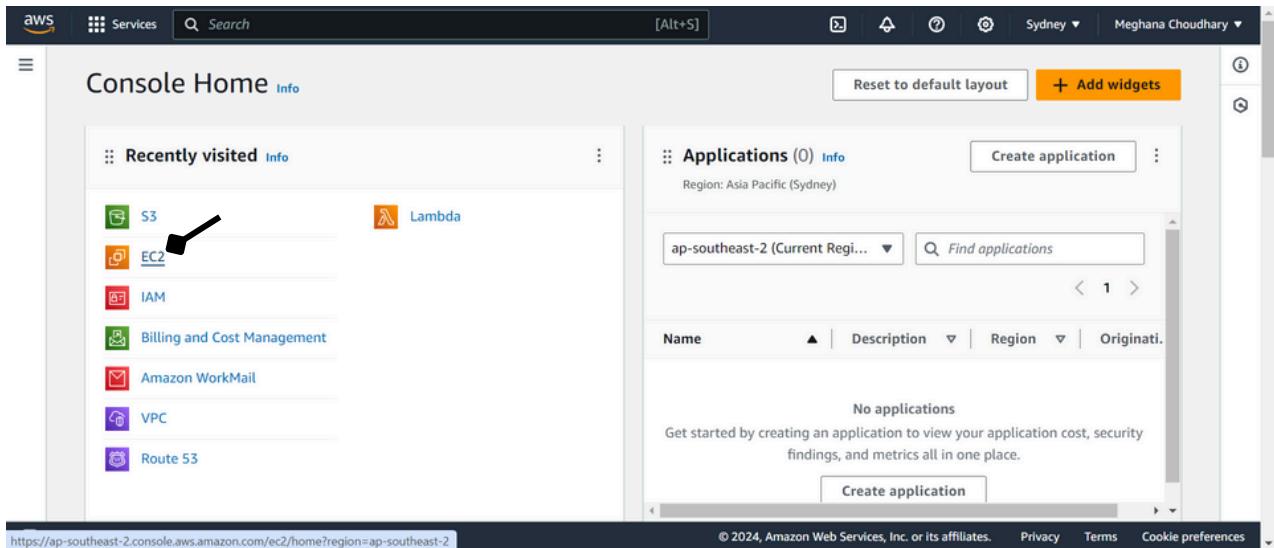


# 11. PROBLEM STATEMENT:

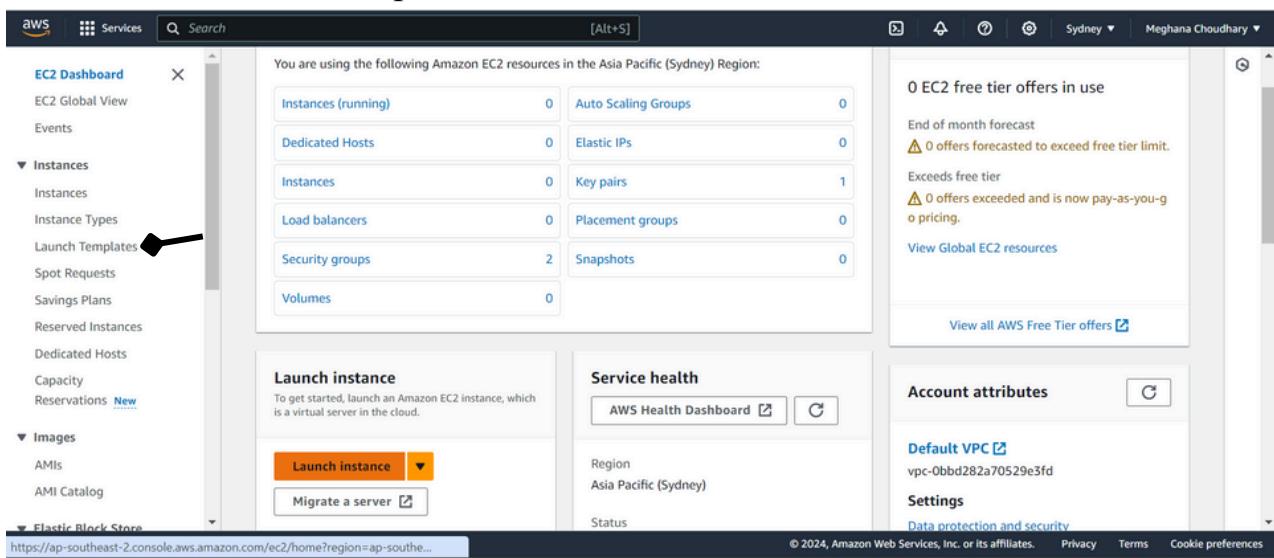
Build scaling plans in AWS that balance load on different EC2 instances.

1. Log into AWS console and click on “EC2”.



The screenshot shows the AWS Console Home page. The top navigation bar includes the AWS logo, a Services menu, a search bar, and account information for "Meghana Choudhary". Below the navigation is a "Console Home" section with a "Recently visited" list containing S3, EC2 (with a black arrow pointing to it), Lambda, IAM, Billing and Cost Management, Amazon WorkMail, VPC, and Route 53. To the right is an "Applications" section showing 0 applications, with a "Create application" button. At the bottom of the page is a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

2. Click on ‘Launch Templates’.



The screenshot shows the EC2 Dashboard. On the left, a sidebar menu lists EC2 Dashboard, EC2 Global View, Events, Instances (with "Launch Templates" highlighted and a black arrow pointing to it), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations (marked as New), Images (AMIs, AMI Catalog), and Elastic Block Store. The main content area displays EC2 resources in the Asia Pacific (Sydney) Region, including Instances (running: 0), Auto Scaling Groups (0), Dedicated Hosts (0), Elastic IPs (0), Instances (0), Key pairs (1), Load balancers (0), Placement groups (0), Security groups (2), Snapshots (0), and Volumes (0). It also features sections for Launch instance (with "Launch instance" and "Migrate a server" buttons), Service health (AWS Health Dashboard), and Account attributes (Default VPC: vpc-0bbd282a70529e3fd, Settings, Data protection and security). The footer includes a copyright notice and links for Privacy, Terms, and Cookie preferences.

### 3. Create launch templates.

The screenshot shows the AWS EC2 Launch Templates page. On the left, there's a navigation sidebar with options like EC2 Dashboard, Services, Compute, Instances, Launch Templates (which is selected), and Images. The main content area has a title 'EC2 launch templates' and a subtitle 'Streamline, simplify and standardize instance launches'. Below that is a text block about using launch templates to automate instance launches. A callout box points to the 'Create launch template' button at the bottom right of the main content area. The footer includes copyright information and links for Privacy, Terms, and Cookie preferences.

### 4. Enter necessary details.

In 'Launch templet name and description' give a name and description to your launch templet and select the check box for 'Provide guidance to help me set up a template that I can use with EC2 Auto Scaling'

The screenshot shows the 'Create launch template' wizard. The current step is 'Launch template name and description'. It has fields for 'Launch template name - required' (containing 'Temp1') and 'Template version description' (containing 'Temp1'). There's also an 'Auto Scaling guidance' section with a checked checkbox for 'Provide guidance to help me set up a template that I can use with EC2 Auto Scaling'. A callout box highlights this checkbox. To the right, there's a 'Summary' panel with sections for Software Image (AMI), Virtual server type (instance type), Firewall (security group), and Storage (volumes). At the bottom, there are 'Cancel' and 'Create launch template' buttons. A tooltip for the 'Free tier' is visible, explaining it includes 750 hours of t2.micro usage per month, 750 hours of public IPv4 address usage per month, and 30 GiB storage. The footer includes CloudShell, Feedback, and cookie preferences links.

For OS Image select ‘Ubuntu Server’

The screenshot shows the 'Launch template contents' section of the AWS Management Console. Under the 'Application and OS Images (Amazon Machine Image)' heading, the 'Ubuntu' option is selected. The 'Summary' panel on the right shows the instance type as 't2.micro', with a note about free tier benefits. Other options like 'Amazon Linux', 'macOS', and 'Windows' are also listed.

For instance type, select ‘t2.micro’ or any other free tire eligible instance type.

The screenshot shows the 'Instance type' selection screen. The 't2.micro' option is highlighted as 'Free tier eligible'. The 'Summary' panel on the right provides details for 't2.micro', including its price and usage terms.

For key pair, select an existing key pair or crate a new key pair.

The screenshot shows the 'Key pair (login)' selection screen. A key pair named 'key1' is selected. The 'Summary' panel on the right shows the same configuration as the previous screenshots, including the 't2.micro' instance type and associated resources.

For security group select an existing security group where port 4000 is open for incoming traffic.

Network settings

Subnet: Don't include in launch template

Create new subnet

Firewall (security groups): Select existing security group (sg-0bab0fa9e6d26002c, VPC: vpc-07d67f2ecedab99bd)

Compare security group rules

Summary

t2.micro

Firewall (security group): :4000 open

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 month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of

Keep Storage and Resource Tag as default.

Storage (volumes)

EBS Volumes

Volume 1 (AMI Root) (8 GiB, EBS, General purpose SSD (gp3))

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

Resource tags

No resource tags are currently included in this template. Add a resource tag to include it in the launch template.

Add new tag

You can add up to 50 more tags.

Summary

Software Image (AMI): Canonical, Ubuntu, 24.04 LTS, ami-0f58b397bc5c1f2e8

Virtual server type (instance type): t2.micro

Firewall (security group): :4000 open

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 month, 750 hours of public IPv4

Create launch template

In Advanced details scroll down to the bottom and add the text to the user data. Then click on Create launch template.

The screenshot shows the 'Create launch template' wizard in the AWS Management Console. On the left, there's a large text input area containing a shell script for provisioning a node.js application. On the right, the 'Summary' section displays configuration details: Software Image (AMI) is Canonical, Ubuntu, 24.04 LTS; Virtual server type (instance type) is t2.micro; Firewall (security group) is MySG; Storage (volumes) is 1 volume(s) - 8 GiB. A tooltip for the Free tier indicates it includes 750 hours of t2.micro or t3.micro usage per year. At the bottom right is a prominent orange 'Create launch template' button, which has a black arrow pointing to it from the text above.

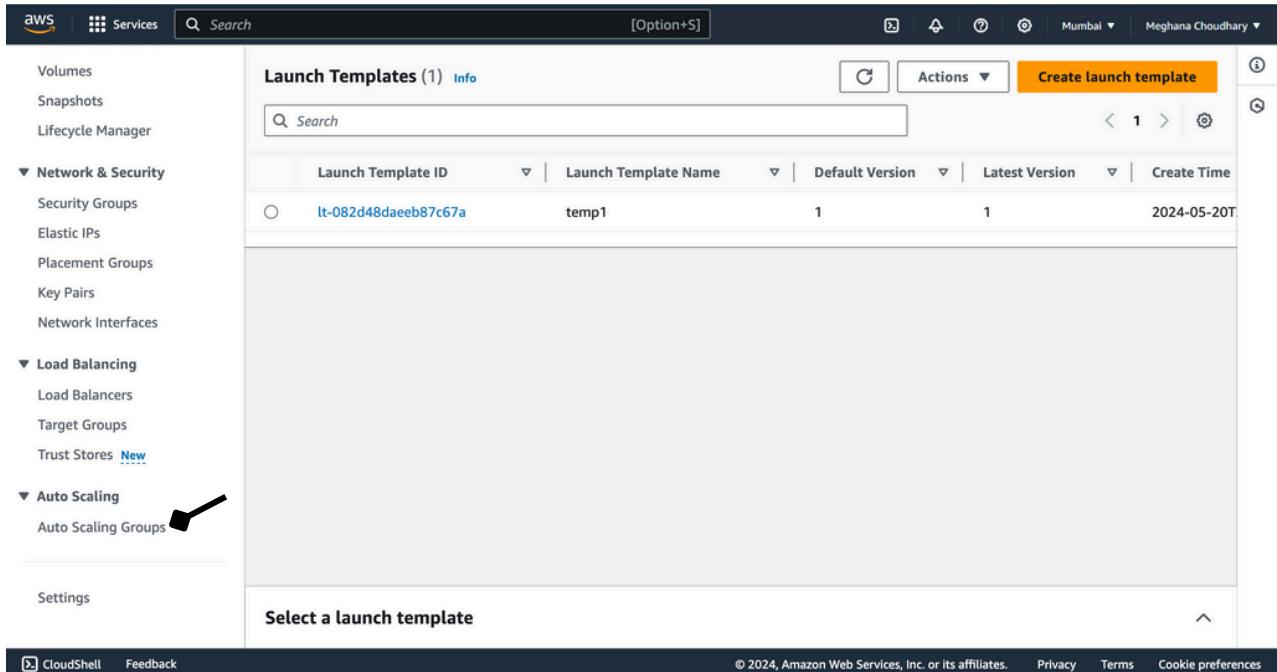
Launch Templates created successfully.

The screenshot shows the 'Launch templates' page in the AWS Management Console. It displays a green success message box stating: 'Success' and 'Successfully created temp1(lt-082d48daeeb87c67a).'. The URL in the top navigation bar is 'EC2 > Launch templates > Create launch template'.

5. Scroll down and click on View launch templates.

The screenshot shows the 'Next Steps' page in the AWS Management Console after creating a launch template. It lists several options: 'Launch an instance', 'Create an Auto Scaling group from your template', 'Create Auto Scaling group', 'Create Spot Fleet', and 'Create Spot Fleet'. At the bottom right is an orange 'View launch templates' button, which has a black arrow pointing to it from the text above.

## 6. Now go to ‘Auto Scaling Groups’

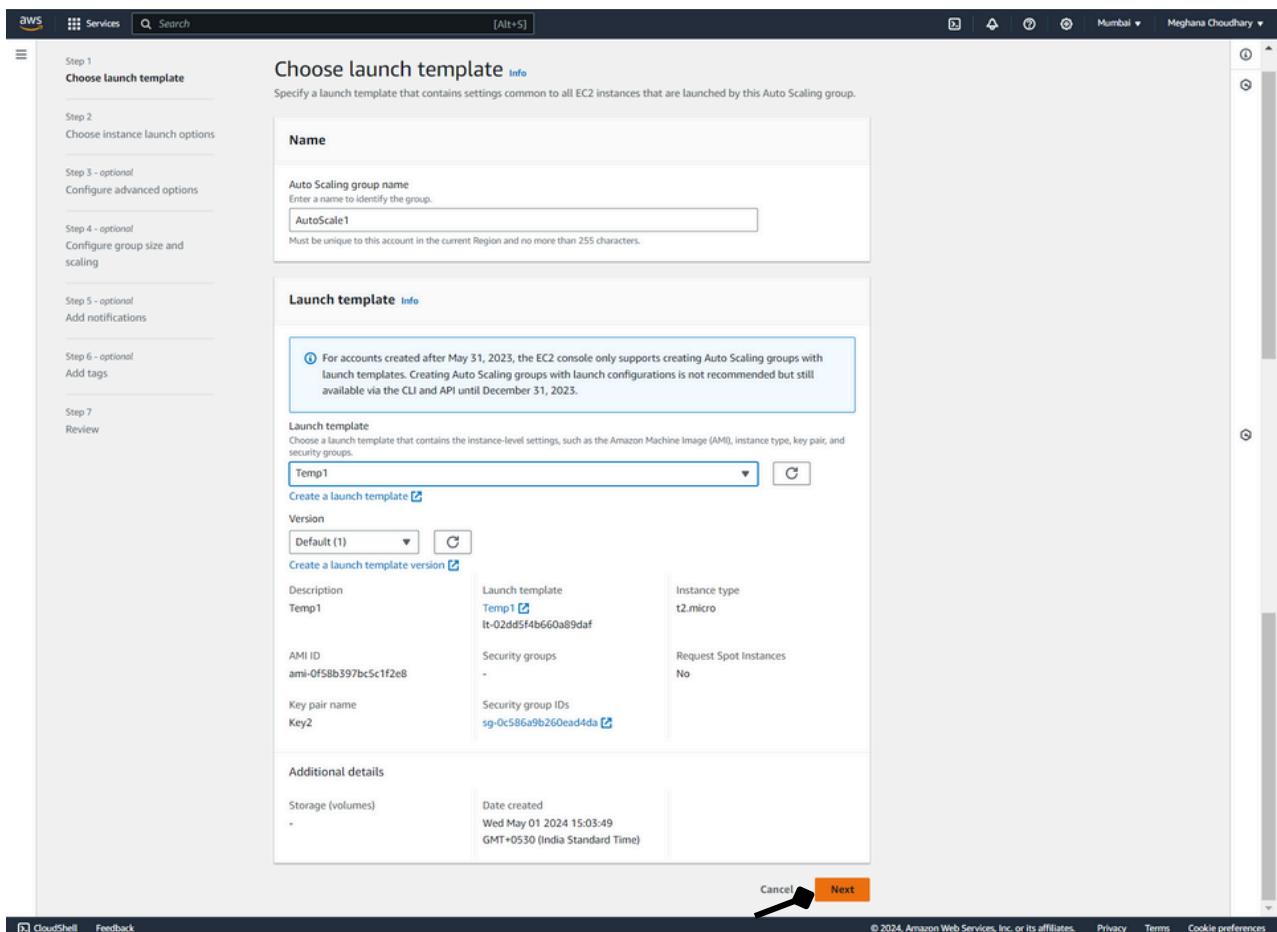


The screenshot shows the AWS Management Console with the 'Services' menu open. Under the 'Auto Scaling' section, the 'Auto Scaling Groups' link is highlighted with a black arrow. The main content area displays a table titled 'Launch Templates (1)'. The table has columns for 'Launch Template ID', 'Launch Template Name', 'Default Version', 'Latest Version', and 'Create Time'. One entry is listed: 'lt-082d48daeeb87c67a' with 'temp1' as the name, version 1, latest version 1, and a creation date of '2024-05-20T'. Below the table, a section titled 'Select a launch template' is visible.

## 7. Add Details for the Auto Scaling Group.

Step 1: Choose launch templates- Give an name to the Auto Scaling Group and choose the Launch template you just created as the launch template.

Click on Next.



The screenshot shows the 'Choose launch template' step of the Auto Scaling Group creation wizard. On the left, a sidebar lists steps from 'Step 1 - Choose launch template' to 'Step 7 - Review'. The main panel shows a 'Name' field containing 'AutoScale1' and a 'Launch template' field containing 'Temp1'. A note states: 'For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.' At the bottom right, there are 'Cancel' and 'Next' buttons, with 'Next' being highlighted by a black arrow.

Step 2: Choose instance launch options- Choose all the ‘Availability Zones’ available.

Click on Next.

The screenshot shows the AWS Auto Scaling 'Create Auto Scaling group' wizard. Step 2 is 'Choose instance launch options'. The 'Instance type requirements' section shows a launch template named 'temp1' with version 'Default' and description 'temp1'. The 'Network' section shows a selected VPC 'vpc-07d57f2ecedab99bd' and two subnets: 'ap-south-1a | subnet-062f2f3a2babf4755' and 'ap-south-1b | subnet-0c5adeb10a1552feb'. The 'Next' button is highlighted with a yellow arrow.

Step 3: Configure advanced options- For Load balancing select ‘Attach to a new load balancer’

The screenshot shows the 'Configure advanced options - optional' step. The 'Load balancing' section contains three options: 'No load balancer', 'Attach to an existing load balancer', and 'Attach to a new load balancer'. The 'Attach to a new load balancer' option is selected and highlighted with a yellow arrow. A black arrow points to the descriptive text for this option: 'Quickly create a basic load balancer to attach to your Auto Scaling group.'

For new load balancer configuration select ‘Application Load Balancer’, ‘Internet-facing’ and make HTTP port from 80 to 4000 while creating a target group.

The screenshot shows the 'Attach to a new load balancer' step in the AWS CloudFormation console. The configuration includes:

- Load balancer type:** Application Load Balancer (HTTP, HTTPS) is selected, highlighted with a black arrow.
- Load balancer name:** AutoScale1-1
- Load balancer scheme:** Internet-facing is selected, highlighted with a black arrow.
- Network mapping:** VPC: vpc-07d67f2ecedab99bd
- Availability Zones and subnets:** Subnets selected for each availability zone:
  - ap-south-1b: subnet-0c5adeb10a1552feb
  - ap-south-1a: subnet-062f2f3a2babf4755
  - ap-south-1c: subnet-0225191dd13cccd1ee
- Listeners and routing:** Protocol: HTTP, Port: 4000, Default routing (forward to): Create a target group (highlighted with a black arrow).
- Tags - optional:** Add tag button and 50 remaining tags.

At the bottom, there are links for CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Health checks tick the check box for ‘Turn on Elastic Load Balancing health checks’ and make the health check grace period 240 seconds.

Click on Next.

The screenshot shows the AWS Auto Scaling configuration wizard. The top navigation bar includes the AWS logo, Services, Search, and user Meghana Choudhary. The main content area has a header 'VPC Lattice integration options' with an 'Info' link. It explains that VPC Lattice improves networking and scalability by integrating with Auto Scaling groups. Below this, there are two radio button options: 'No VPC Lattice service' (selected) and 'Attach to VPC Lattice service'. A 'Create new VPC Lattice service' button is also present.

**Health checks** section:

- EC2 health checks:** Set to 'Always enabled'.
- Additional health check types - optional:** A checkbox labeled 'Turn on Elastic Load Balancing health checks' is checked, with a 'Recommended' link next to it. A note below states: 'EC2 Auto Scaling will start to detect and act on health checks performed by Elastic Load Balancing. To avoid unexpected terminations, first verify the settings of these health checks in the [Load Balancer console](#)'.
- Turn on VPC Lattice health checks:** This option is not selected.

**Health check grace period:** Set to 240 seconds.

**Additional settings** section:

- Monitoring:** A checkbox 'Enable group metrics collection within CloudWatch' is not selected.
- Default instance warmup:** A checkbox 'Enable default instance warmup' is not selected.

At the bottom, there are buttons for 'Cancel', 'Skip to review', 'Previous' (highlighted with a black arrow), and 'Next' (highlighted with a black arrow).

Page footer: CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences.

Step 4: Configure group size and scaling- In Group size set Desired capacity as 2. In Scaling set Min desired capacity as 2 and Max desired capacity as 3. In Automatic scaling choose target scaling policy and set the instance warmup as 240 second.

Keeping everything else as default, Click on Next.

**Configure group size and scaling - optional**

Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

**Group size**

Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

**Desired capacity type**

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

**Desired capacity**

Specify your group size.

**Scaling**

You can resize your Auto Scaling group manually or automatically to meet changes in demand.

**Scaling limits**

Set limits on how much your desired capacity can be increased or decreased.

<b>Min desired capacity</b>	<b>Max desired capacity</b>
2	3

**Automatic scaling - optional**

Choose whether to use a target tracking policy

No scaling policies  
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy  
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

**Scaling policy name**

Target Tracking Policy

**Metric type**

Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.

Average CPU utilization

**Target value**

50

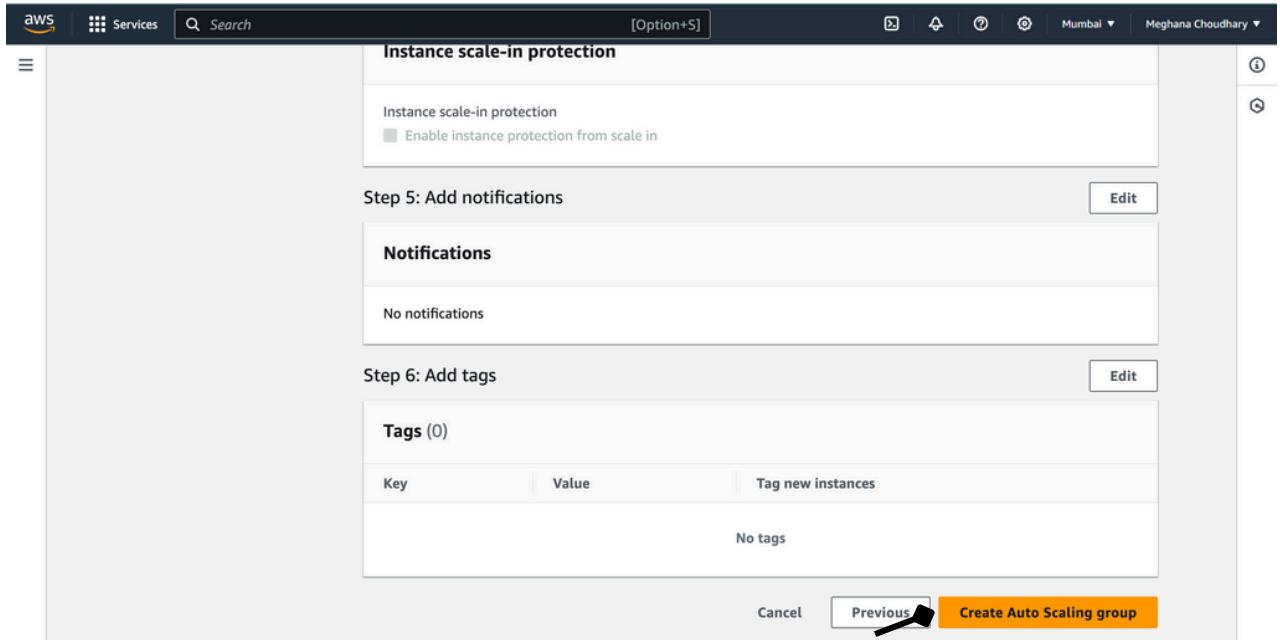
**Instance warmup**

240 seconds

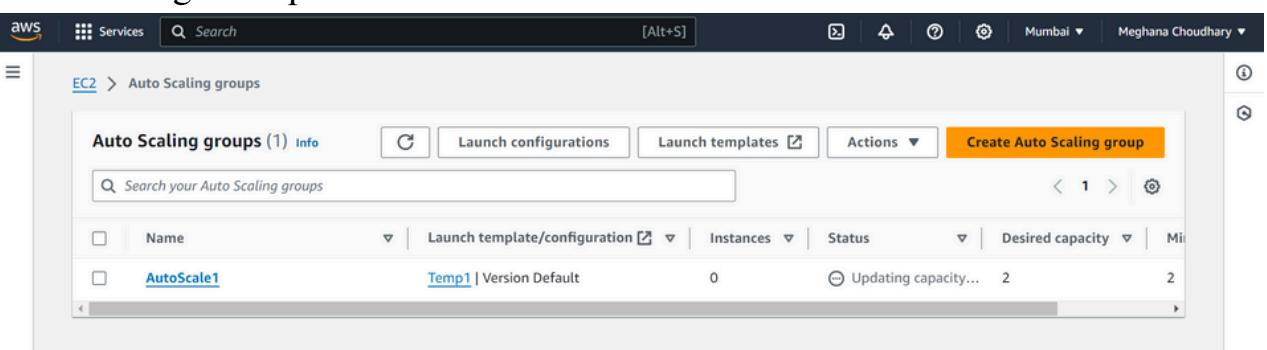
Disable scale in to create only a scale-out policy

Skip through Step 5 and Step 6 by clicking Next.

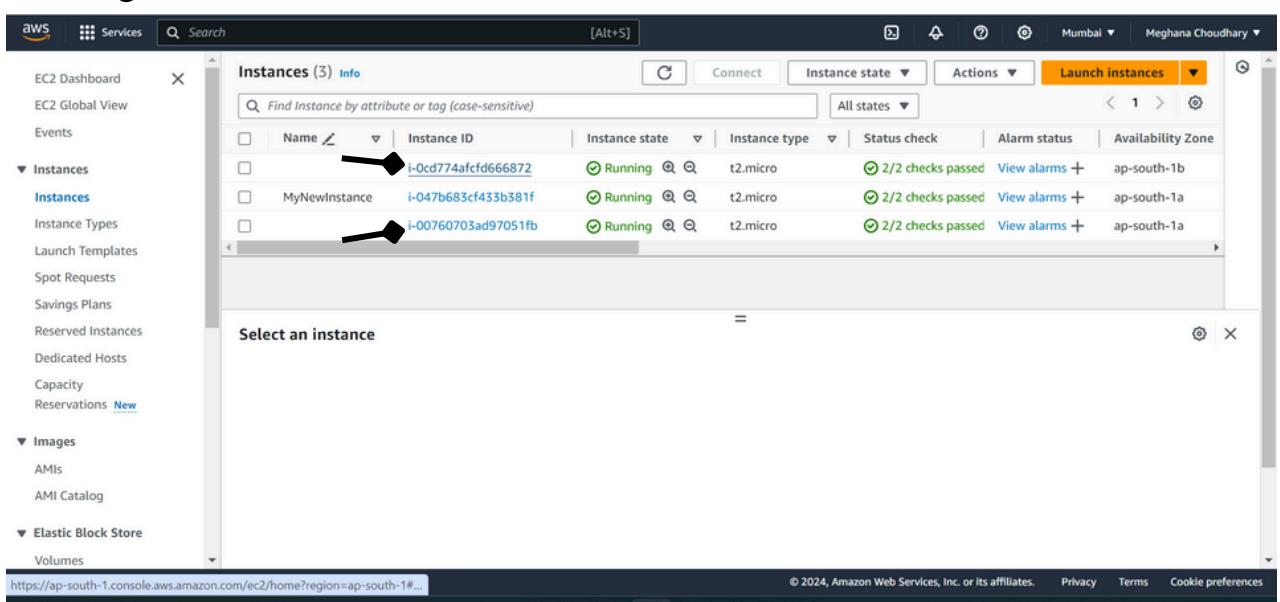
Finally in Step 7 Click on Create Auto Scaling group.



Auto Scaling Group is created.



8. Going back to instances we can see two unnamed instances are created.



8. Copy the public IPv4 address of any one of the two instance.

The screenshot shows the AWS EC2 Instances page. In the center, there's an 'Instance summary' card for an instance with ID i-0cd774afcfd666872. The Public IPv4 address 3.201.193.122 is listed and has a green checkmark indicating it was copied. To the right, there's a section for Private IPv4 addresses (172.31.4.57) and Public IPv4 DNS (ec2-13-201-193-122.ap-south-1.compute.amazonaws.com). The instance state is shown as 'Running'.

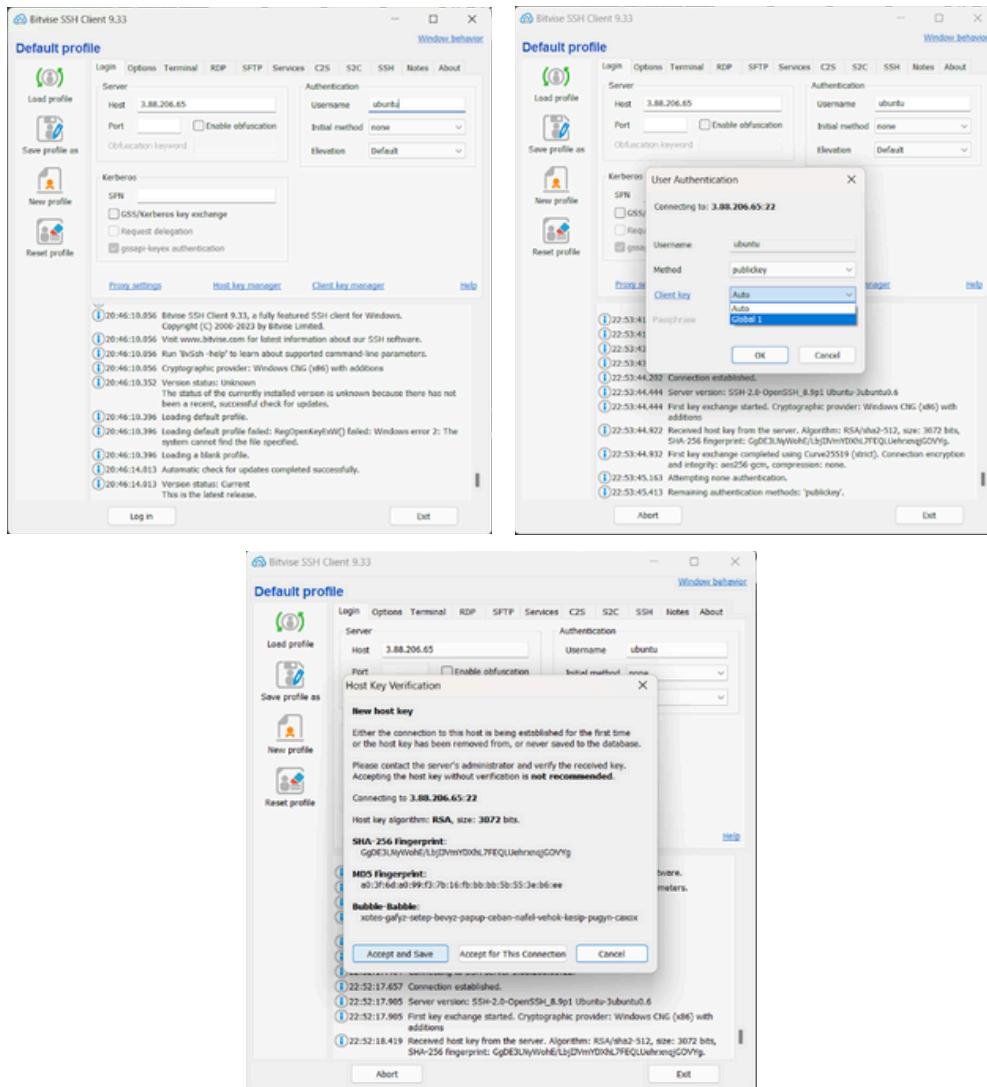
9. In Bitvise SSH Client, paste the ‘Public IPV4 address’ and click on ‘Client key Manager’.

The screenshot shows the Bitvise SSH Client 9.33 interface. The 'Default profile' tab is selected. In the 'Server' section, the host is set to 3.88.206.65. The 'Client key manager' tab is highlighted with a black arrow. The log window at the bottom shows several informational messages about the client's startup and configuration.

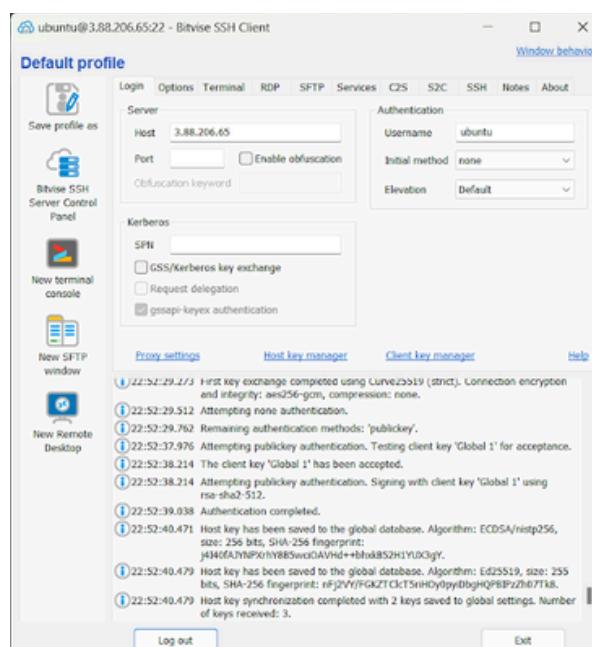
10. Under ‘Client key manager’, if there is any existing key then remove it and click on ‘Import’, select the same key that you used while creating the instance, from your file directory and then again click on ‘Import’ and then the key is successfully imported.

The screenshot shows the Bitvise Client Key Management interface. On the left, the 'Client Key Manager' table lists a single key: Global 1 (RSA, 2048 bits, SHA-256, Passphrase no). On the right, the 'Import Client Key' dialog box is open. It shows the key details: Location Global, Algorithm RSA, Size 2048. The 'Change passphrase' section indicates the key is not passphrase protected. The 'Comment' field is empty. At the bottom, there are 'Import', 'Cancel', and 'Cancel All' buttons.

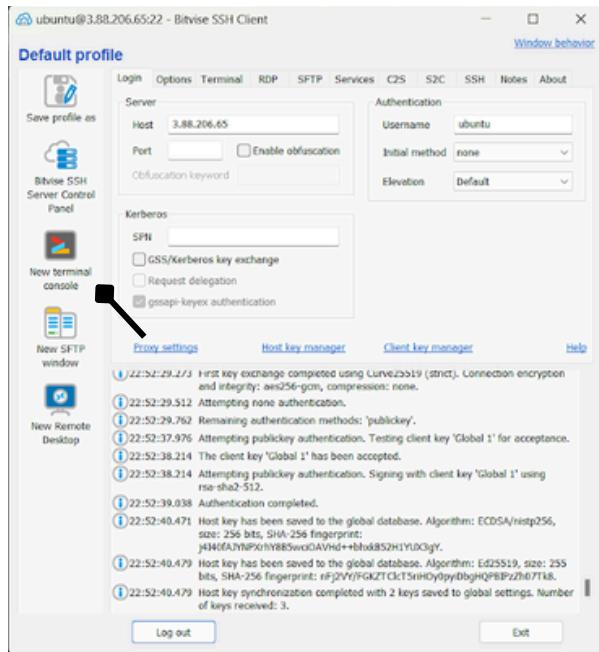
10. In ‘Bitvise SSH Client’, under ‘Default profile’ give the ‘Username’ as ‘ubuntu’ then select ‘Global1’ from ‘Client Key Manager’ then click on ‘Log in’ & ‘Accept and save’.



If the ‘Log In’ button changes to ‘Log Out’, then you are logged in successfully.



11. Click on ‘New terminal consol’, to access the terminal of your instance.



8. Once you are inside the terminal of your instance

To create a infinitely running bash script

```
vim infi.sh
```

```
ubuntu@ip-172-31-12-249:~$ vim infi.sh
```

In the bash script, add the following code.

```
#!/bin/bash
while(true)
do
```

```
    echo "Inside Loop"
```

```
done
```

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

```
while(true)
do
    echo "InsideLoop"
done
```

To make the script executable

```
sudo chmod 777 infi.sh
```

```
ubuntu@ip-172-31-5-108:~$ sudo chmod 777 infi.sh
```

Run the script

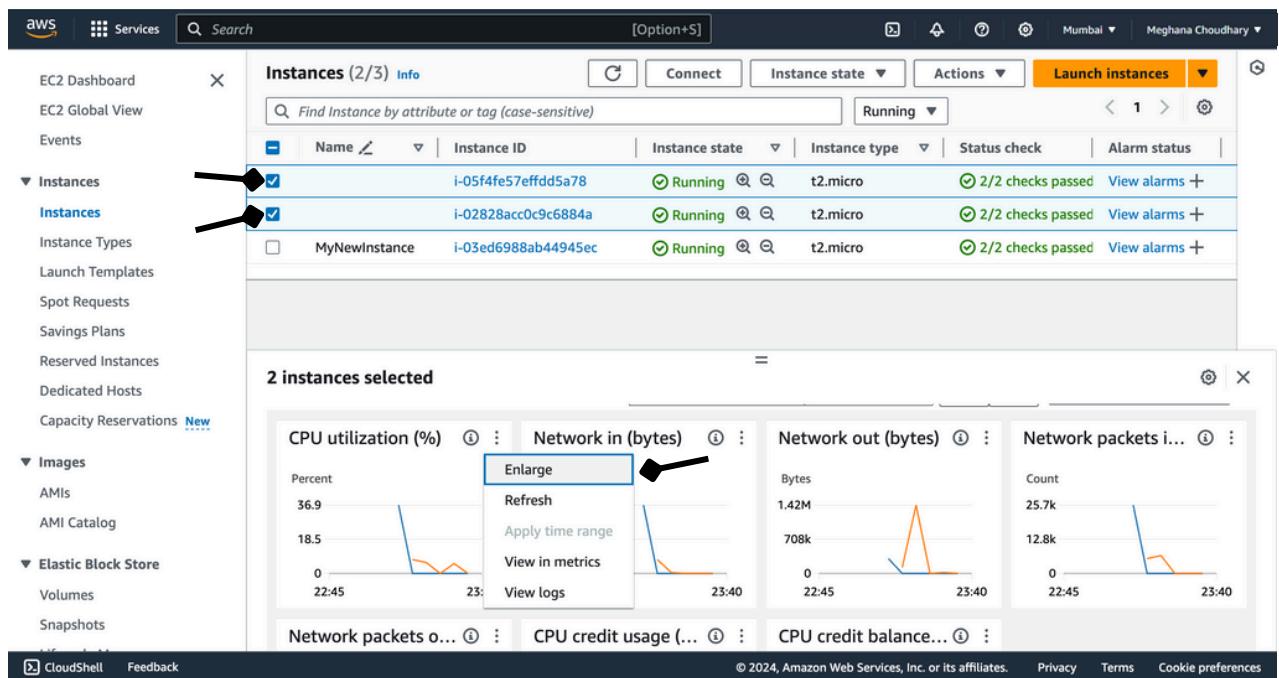
```
sh infi.sh
```

```
ubuntu@ip-172-31-5-108:~$ sh infi.sh
```

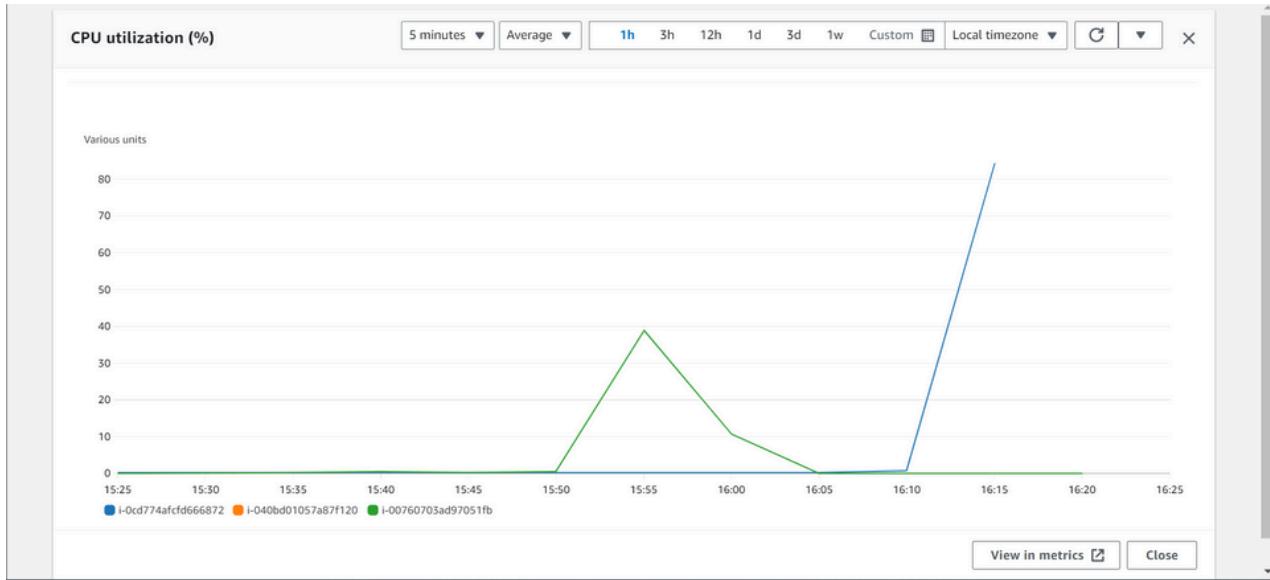
9. An infinite loop starts to run, stressing the cpu of the instance.

```
InsideLoop
```

10. Selecting the two instances we can take a look at the cpu utilisation graph.



11. After a while the instance reaches and surpasses it's 50% CPU utilisation threshold.



12. So to take that load off the instance, the auto scaler spins up another instance to balance the load.

The screenshot shows the AWS EC2 Dashboard with the 'Instances' section selected. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Instances (with a sub-link for Instances), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, and Reservations. The main pane displays a table of running instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
	i-040bd01057a87f120	Running	t2.micro	Initializing	View alarms +	ap-south-1b
	i-0cd774afcf666872	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b
MyNewInstance	i-047b683cf433b381f	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a
	i-00760703ad97051fb	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a

A black arrow points to the 'Name' column header. A modal window titled 'Select an instance' is open at the bottom, showing a list of instances: 'MyNewInstance' and 'i-00760703ad97051fb'. The bottom of the screen includes standard AWS footer links: CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.