

Task-6.

Cost Optimization

Use free-tier eligible instances

Minimal resources

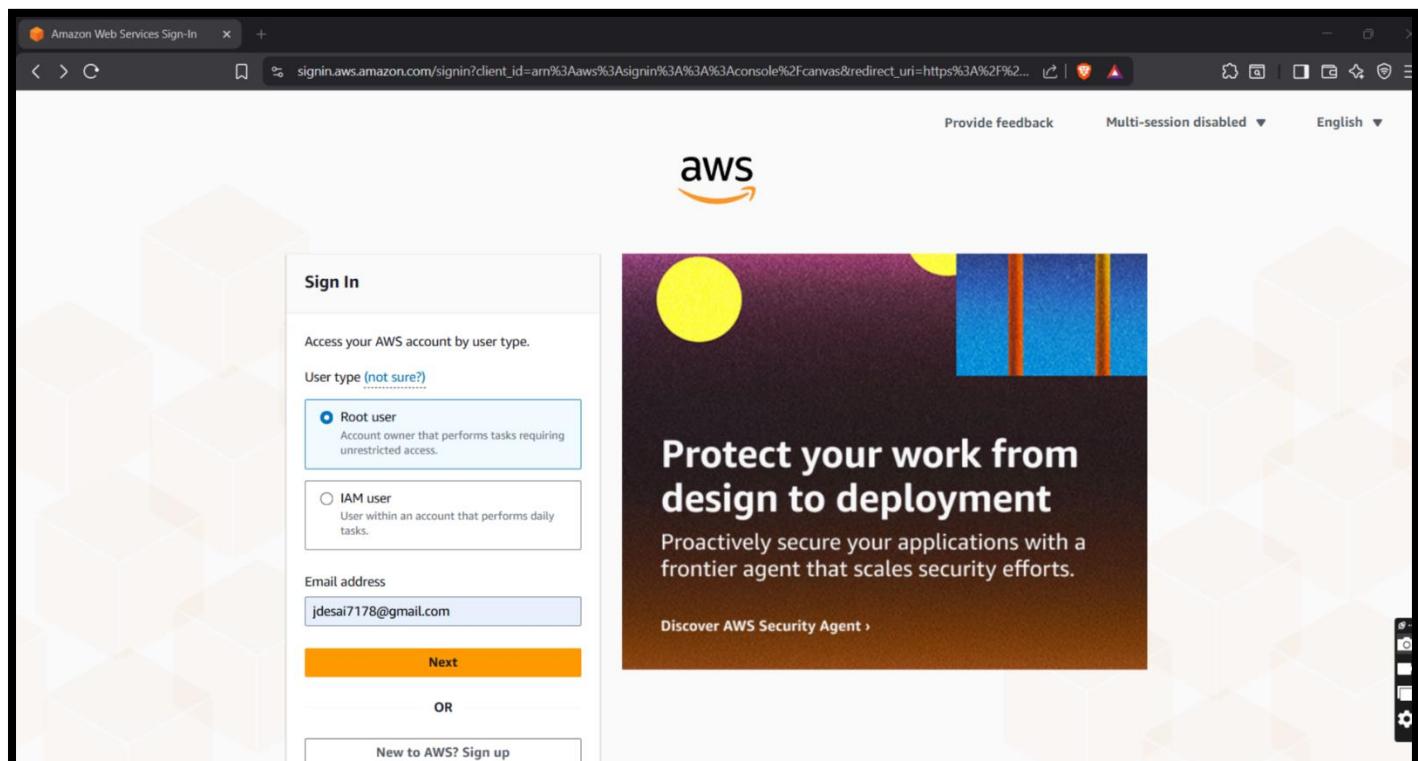
Auto Scaling to avoid over-provisioning

Steps:

Step 1: Cost Optimization – Implementation Steps

Using Free-Tier Eligible Instances

1. Login to AWS Console



Step 2: Going to EC2 → Launch Instance

The screenshot shows the AWS EC2 Instances page. The left sidebar has sections for EC2 (Dashboard, AWS Global View, Events, Instances, Images, Elastic Block Store), S3, IAM, Simple Notification Service, VPC, Aurora and RDS. The main area has tabs for Instances (selected) and Info. It includes a search bar, filters (Instance state = running, Clear filters), and columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. A message says "No matching instances found". Below is a section titled "Select an instance" with a "Launch instances" button.

Step 3: Choose Amazon Linux or Ubuntu (because it is free-tier eligible)

and select free tier eligible ami. These AMIs are marked as “Free tier eligible”, AWS provides these OS images at no additional cost, Optimized for AWS environment

The screenshot shows the AWS Launch an instance page. The left sidebar has sections for EC2 (Instances selected), S3, IAM, Simple Notification Service, VPC, Aurora and RDS. The main area has tabs for Name and tags (Info selected) and Application and OS Images (Amazon Machine Image). The Name and tags section shows "task-6" and "Add additional tags". The Application and OS Images section shows a search bar and a grid of recent AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. To the right is a summary panel with "Number of instances" set to 1, "Software Image (AMI)" (Amazon Linux 2023 AMI 2023.10...), "Virtual server type (instance type)" (t2.micro), "Firewall (security group)" (New security group), "Storage (volumes)" (1 volume(s) - 8 GiB), and a note about "Free tier". At the bottom are "Launch instance" and "Preview code" buttons.

Step 4: Select instance type t2.micro / t3.micro [free tier eligible]

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Instance type' section, the 't2.micro' option is selected, which is marked as 'Free tier eligible'. Below this, there's a note about additional costs for AMIs with pre-installed software. The 'Key pair (login)' section shows a dropdown for 'Key pair name - required' containing 'geeta', with a 'Create new key pair' button. The 'Network settings' section shows 'Network' set to 'vpc-052859de1479c19a' and 'Subnet' set to 'No preference (Default subnet in any availability zone)'. On the right side, the 'Summary' panel shows 1 instance being launched with the 'Amazon Linux 2023 AMI 2023.10...' selected. It also lists 'Virtual server type (instance type)' as 't2.micro', 'Firewall (security group)' as 'New security group', and 'Storage (volumes)' as '1 volume(s) - 8 GiB'. A 'Free tier' callout indicates that in the first year, 750 hours per month are covered. At the bottom right are 'Launch instance' and 'Preview code' buttons.

1. In Instance Type, choose:

t2.micro OR

t3.micro

2. Instance shows Free tier eligible

- 1 vCPU
- 1 GB RAM
- Suitable for small applications and testing
- Covered under 750 hours/month

3. I Choose Existing Key-Pair

Choose existing key pair OR create new

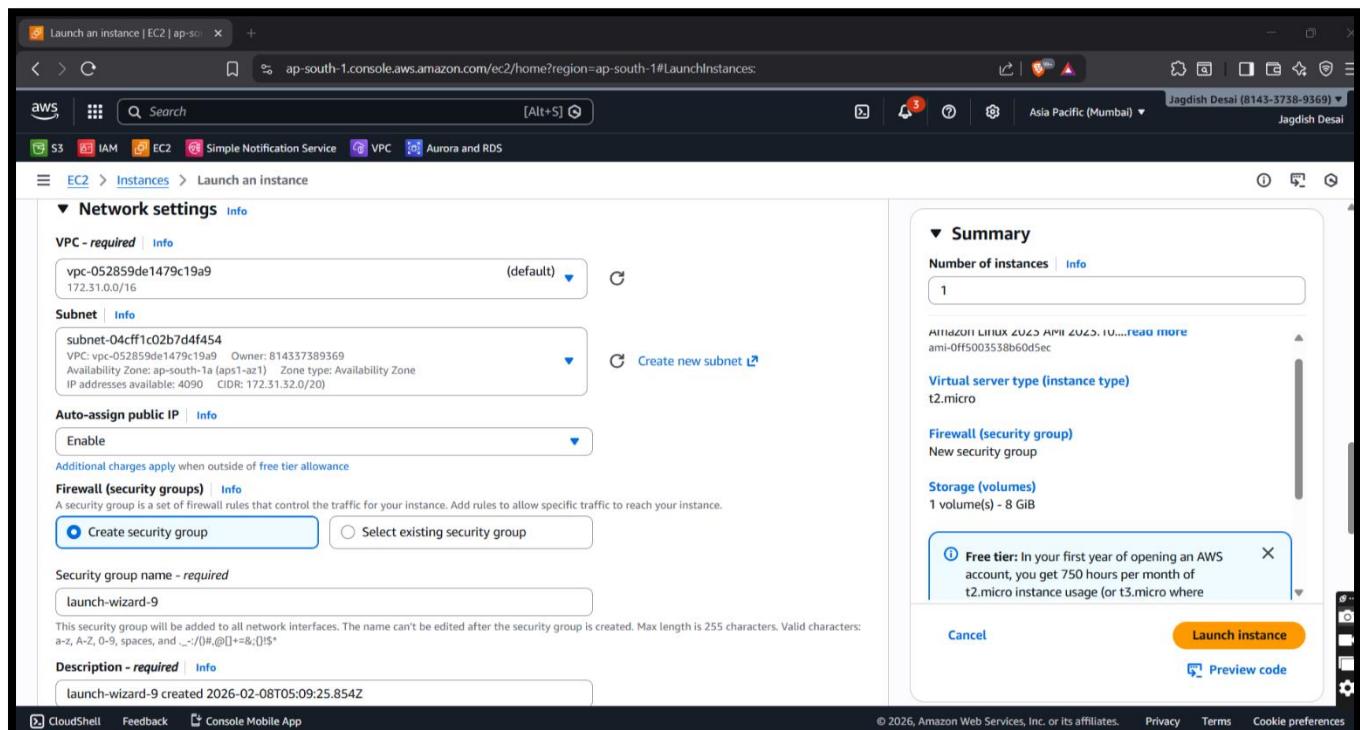
Required for SSH login

No key pair → no server access

Step 5: Configuring Instance Settings

1. Leave default settings:
 - o Network: Default VPC
 - o Subnet: Auto-assign

2. Enable **Auto-assign Public IP** because
 - No extra networking cost
 - Easy access using public IP



Step 6: Configuring Storage

1. Root volume size: **8 GB (default)**
2. Volume type: **gp2 / gp3**

Because:

- Free tier allows **30 GB** total
- Keeping minimal storage avoids extra charges

The screenshot shows the 'Advanced network configuration' step of the 'Launch an instance' wizard. Under 'Configure storage', a single volume is selected: 8 GiB gp3, labeled as 'Root volume, 3000 IOPS, Not encrypted'. A note indicates that free-tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. An 'Add new volume' button is available. Below this, a note says to click refresh to view backup information. Under 'Advanced details', there is a note about file systems. On the right, the 'Summary' section shows 1 instance, the AMI (Amazon Linux 2023 AMI 2023.10...), the instance type (t2.micro), and a note about the free tier. The 'Launch instance' button is prominently displayed.

Step 7: Configure Security Group

1. Create a new security group
2. Allow:
 - o SSH → Port 22 (My IP)
 - o HTTP → Port 80 (Anywhere)

Because We give

- Secure access
- Only required ports open (cost & security optimized)

The screenshot shows the 'Inbound Security Group Rules' step of the 'Launch an instance' wizard. It displays two rules: 'Security group rule 1 (TCP, 22, 0.0.0.0/0)' allowing SSH (TCP port 22) from anywhere, and 'Security group rule 2 (TCP, 80, 0.0.0.0/0)' allowing HTTP (TCP port 80) from anywhere. A warning message at the bottom states: 'Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' On the right, the 'Summary' section shows 1 instance, the AMI (Amazon Linux 2023 AMI 2023.10...), the instance type (t2.micro), and a note about the free tier. The 'Launch instance' button is present.

Step 8: Launch Instance

The screenshot shows the AWS EC2 Instances page. In the left sidebar, under the 'Instances' section, 'Instances' is selected. The main table displays one instance: 'task-6' (i-041888754f68c68ee), which is currently 'Running'. The instance type is 't2.micro'. Other columns include 'Instance ID', 'Status check', 'Alarm status', 'Availability Zone', and 'Public IP'. Below the table, the instance details for 'i-041888754f68c68ee (task-6)' are shown, including its public and private IP addresses and DNS name.

Our Task-6 Instance Successfully Launched with minimal Resources.

Benefits of Using Minimal Resources

Lower cost

we pay only for what we use, so AWS bills stay low.

Free-tier safe

Helps you stay within free-tier limits and avoid charges.

Less waste

No unused EC2, storage, or IPs running without purpose.

Easy management

Fewer resources = easier monitoring and troubleshooting.

Better scaling

Auto Scaling works efficiently when starting small.

Improved performance control

we can clearly see when to scale up or down.

Auto Scaling to avoid over-provisioning

Step 1: Configure Auto Scaling (Avoid Over-Provisioning)

Create Launch Template

1. Go to EC2 → Launch Templates
2. Create template using:
 - Free-tier instance (t2.micro)
 - AMI with your app installed
3. Save template

The screenshot shows the 'Create launch template' page in the AWS Management Console. The 'Launch template contents' section is open, displaying the 'Application and OS Images (Amazon Machine Image)' section. It lists various AMIs including Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debi. A search bar at the top of this section allows users to find specific AMIs. Below the list, there's a detailed view of the 'Amazon Machine Image (AMI)' for Amazon Linux 2023 kernel-6.1 AMI, showing its base AMI ID, virtualization type (hvm), and root device type (ebs). The 'Free tier eligible' status is indicated. To the right, a 'Summary' panel provides a quick overview of the selected AMI and instance type.

This screenshot continues from the previous one, showing the 'Create launch template' page. The 'Instance type' section is now active, showing the t2.micro instance type selected. It provides details about the instance, including family, CPU, memory, and current generation. The 'Advanced' tab is visible above the instance type section. Below it, the 'Key pair (login)' section is shown, where a key pair named 'geeta' is selected. The 'Network settings' section is also partially visible at the bottom. The 'Summary' panel on the right remains the same, providing a summary of the selected configuration.

The screenshot shows the 'Create launch template' wizard in the AWS EC2 console. The 'Software Image (AMI)' is set to 'Amazon Linux 2023 AMI 2023.10...' with a note about the free tier. The 'Virtual server type (instance type)' is 't2.micro'. The 'Firewall (security group)' is 'launch-wizard-8'. Under 'Storage (volumes)', there is 1 volume(s) of 8 GiB. The 'User data' section contains a script to update yum, install httpd, enable it, and create an index.html file. A note indicates that user data has already been base64 encoded. The 'Summary' section provides a summary of the configuration.

Mytmp Template Successfully Launched:

The screenshot shows the 'Launch Templates' page in the AWS EC2 console. It displays a single launch template named 'mytmp' with Launch Template ID 'lt-0929995699f90e3c1'. The table includes columns for Launch Template ID, Launch Template Name, Default Version, Latest Version, Create Time, and Created By. The 'Actions' dropdown and 'Create launch template' button are visible at the top right.

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
lt-0929995699f90e3c1	mytmp	1	1	2026-02-08T06:03:27.000Z	arn:aws:iam::8143

Step 3: Create Target Groups

The screenshot shows the 'Create target group' wizard in the AWS Management Console. The title bar says 'Step 1 Create target group | EC2'. The left sidebar shows 'Step 1 Create target group' selected. The main content area is titled 'Create target group' and contains a note: 'A target group can be made up of one or more targets. Your load balancer routes requests to the targets in a target group and performs health checks on the targets.' Below this is a section titled 'Settings - immutable' with a note: 'Choose a target type and the load balancer and listener will route traffic to your target. These settings can't be modified after target group creation.' The 'Target type' section lists four options:

- Instances**: Supports load balancing to instances in a VPC. Integrate with Auto Scaling Groups or ECS services for automatic management. Suitable for: ALB, NLB, GWLB.
- IP addresses**: Supports load balancing to VPC and on-premises resources. Facilitates routing to IP addresses and network interfaces on the same instance. Supports IPv6 targets. Suitable for: ALB, NLB, GWLB.
- Lambda function**: Supports load balancing to a single Lambda function. ALB required as traffic source. Suitable for: ALB.
- Application Load Balancer**: Allows use of static IP addresses and PrivateLink with an Application Load Balancer. NLB required as traffic source. Suitable for: NLB.

The 'Target group name' field is present, with a note: 'Name must be unique per Region per AWS account.' A character limit note: 'Accepts: a-z, A-Z, 0-9, and hyphen (-). Can't begin or end with hyphen. 1-32 total characters; Count: 0/32'.

Review Create Target group:

The screenshot shows the 'Review and create' step of the wizard. The title bar says 'Step 3 Create target group | EC2'. The left sidebar shows 'Step 1 Create target group', 'Step 2 - recommended Register targets', and 'Step 3 Review and create' selected. The main content area is titled 'Review and create' with a note: 'Review your target group configuration before creating'. It includes a 'Step 1: Target group details' section and a 'Health check details' section.

Step 1: Target group details

Name	Target type	Protocol : Port	Protocol version
mytg	Instance	HTTP: 80	HTTP1
VPC	IP address type		
vpc-052859de1479c19a9	IPv4		

Health check details

Health check protocol	Health check path	Health check port	Interval
HTTP	/	traffic-port	30 seconds
Timeout	Healthy threshold	Unhealthy threshold	Success codes
5 seconds	5	2	200

Step 4: Create Load Balancer

The screenshot shows the AWS CloudFormation Create Stack Wizard Step 4: Set Stack Configuration. The page title is "Create Application Load Balancer". The "Name" field contains "my-alb". The "Scheme" section has "Internet-facing" selected. The "Load balancer IP address type" section has "IPv4" selected. The "Security groups" section is empty.

Select atleast 2 availability zones

The screenshot shows the AWS CloudFormation Create Stack Wizard Step 4: Set Stack Configuration. The "Availability Zones and subnets" section lists three availability zones: "ap-south-1a (aps1-az1)", "ap-south-1b (aps1-az3)", and "ap-south-1c (aps1-az2)". Each zone has a checked checkbox and a dropdown menu showing its subnets. The "Security groups" section is empty.

Attach Target Group:

The screenshot shows the 'Default action' and 'Routing action' sections. Under 'Default action', there is a note about choosing a default action for traffic. Under 'Routing action', three options are listed: 'Forward to target groups' (selected), 'Redirect to URL', and 'Return fixed response'. In the 'Forward to target group' section, a target group named 'mytg' is selected. The 'Target type' is set to 'Instance, IPv4' and 'Target stickiness' is set to 'Off'. The 'HTTP' scheme is chosen. A weight of '1' is assigned with a 'Percent' of '100%'. Below this, there is a link to 'Add target group' and a note that up to 4 more target groups can be added. In the 'Target group stickiness' section, there is a note about enabling load balancer stickiness and a checkbox for 'Turn on target group stickiness'.

Successfully created load balancer

The screenshot shows the AWS Management Console with the URL 'ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LoadBalancers'. The left sidebar shows navigation for EC2, Load balancers, Auto Scaling, and other services. The main area displays a table of 'Load balancers (1/1)'. The table includes columns for Name (my-alb), State (Active), Type (application), Scheme (Internet-facing), IP address type (IPv4), VPC ID (vpc-052859de1479c19a9), and Availability Zones (3 Availability Zones). Below the table, the details for 'Load balancer: my-alb' are shown, including tabs for Details, Listeners and rules, Network mapping, Resource map, Security, Monitoring, Integrations, Attributes, and Capacity. The 'Details' tab is selected, displaying information such as Load balancer type (Application), Status (Active), VPC (vpc-052859de1479c19a9), Scheme (Internet-facing), Hosted zone (7P07RAFLXTN2K), Availability Zones (subnet-04cff1c02b7d4f454), and Date created (February 8, 2026, 12:00 (UTC+05:30)).

Step 5: Creating Auto Scaling Group

Select launch template

The screenshot shows the 'Create Auto Scaling group' wizard at Step 1: 'Choose launch template'. The left sidebar lists steps 1 through 7. Step 1 is selected. The main area is titled 'Choose launch template' with a note: 'Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.' It includes fields for 'Name' (set to 'myasg') and 'Launch template' (set to 'mytmp'). 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.' The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and Console Mobile App.

Choose VPC and subnets

The screenshot shows the 'Create Auto Scaling group' wizard at Step 2: 'Choose VPC and subnets'. The left sidebar lists steps 1 through 7. Step 2 is selected. The main area includes sections for 'VPC' (selected VPC: 'vpc-052859de1479c19a9'), 'Availability Zones and subnets' (three subnets listed: 'aps1-az1 (ap-south-1a) | subnet-04cff1c02b7d4f454', 'aps1-az2 (ap-south-1c) | subnet-0b794ec7dbb17441d', and 'aps1-az3 (ap-south-1b) | subnet-0870f37a4647fd624'), and 'Availability Zone distribution - new' (radio button selected: 'Balanced best effort'). The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and Console Mobile App.

Integrate with other services –

The screenshot shows the AWS Auto Scaling group creation wizard at Step 3: Integrate with other services. The left sidebar lists steps from 3 to 7. The main panel is titled 'Load balancing' and contains three options: 'No load balancer' (disabled), 'Attach to an existing load balancer' (selected), and 'Attach to a new load balancer' (disabled). Below this is a section for attaching to an existing load balancer, with a dropdown menu showing 'mytg | HTTP' selected. At the bottom is a section for VPC Lattice integration options.

Configuring group size:

The screenshot shows the AWS Auto Scaling group creation wizard at Step 4: Configure group size and scaling. The left sidebar lists steps from 3 to 7. The main panel is titled 'Group size' and includes fields for 'Desired capacity type' (Units: number of instances) set to 1, and 'Desired capacity' (Specify your group size) also set to 1. Below this is a 'Scaling' section with 'Min desired capacity' (1) and 'Max desired capacity' (3). Further down are sections for 'Automatic scaling - optional' (Choose whether to use a target tracking policy) and 'Target tracking scaling policy' (No scaling policies selected).

Choose whether to use a target tracking policy

The screenshot shows the AWS CloudWatch Metrics console interface. At the top, there are tabs for 'Create Auto Scaling group | EC2', 'Load balancers | EC2 | ap-south-1', and 'Target groups | EC2 | ap-south-1'. The main content area is titled 'Automatic scaling - optional' and contains a section for 'Choose whether to use a target tracking policy'. There are two options: 'No scaling policies' (selected) and 'Target tracking scaling policy' (highlighted with a blue border). The 'Target tracking scaling policy' section includes fields for 'Scaling policy name' (set to 'Target Tracking Policy'), 'Metric type' (set to 'Average CPU utilization'), 'Target value' (set to '50'), and 'Instance warmup' (set to '200 seconds'). A checkbox for 'Disable scale in to create only a scale-out policy' is also present. The bottom of the screen shows standard AWS navigation links like CloudShell, Feedback, and Console Mobile App.

Successfully created Auto Scaling group: myasg

The screenshot shows the AWS Auto Scaling Groups console. The top navigation bar includes tabs for 'Auto Scaling groups | EC2 | ap-south-1', 'Load balancers | EC2 | ap-south-1', and 'Target groups | EC2 | ap-south-1'. The main content area displays a table titled 'Auto Scaling groups (1/1)'. The table has columns for Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and Availability Zones. One row is selected, showing 'myasg' under 'Name' and 'mytmp | Version Default' under 'Launch template/configuration'. The 'Actions' dropdown menu at the top right contains options like 'Edit', 'Delete', and 'Scale'. Below the table, a detailed view for 'Auto Scaling group: myasg' is shown with tabs for 'Details', 'Integrations', 'Automatic scaling', 'Instance management', 'Instance refresh', 'Activity', 'Monitoring', and 'Tags - moved'. The 'Automatic scaling' tab is active, showing the 'myasg Capacity overview' with sections for 'Desired capacity' (1), 'Scaling limits' (1 - 3), 'Desired capacity type' (Units (number of instances)), and 'Status' (-). The bottom of the screen shows standard AWS navigation links like CloudShell, Feedback, and Console Mobile App.

- **Users** → Send requests to **ALB**
- **ALB** → Distributes requests to **Target Group**
- **Target Group** → Monitors **EC2 instances** for health
- **ASG** → Automatically scales EC2 based on load (CPU, traffic)

First Instance will be created by auto scaling groups-asg 1

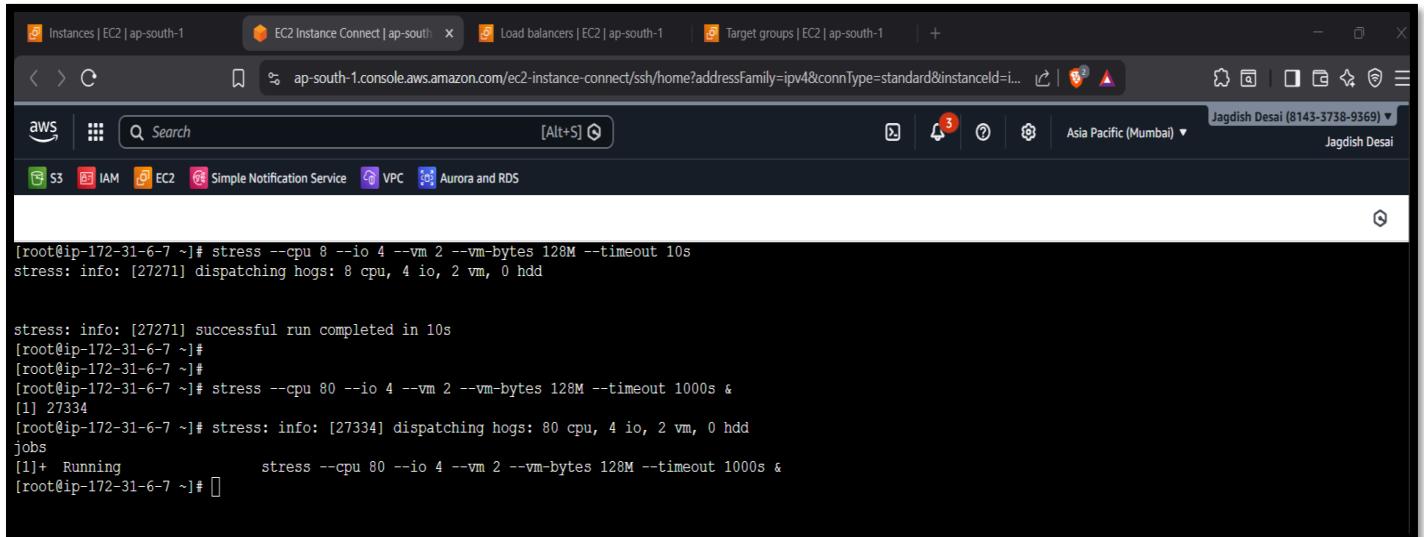
The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with 'EC2' selected, followed by 'Instances' and 'Images'. The main area displays a green success message: 'Successfully initiated stopping of i-041888754f68c68ee'. Below this, the 'Instances (1/1)' section shows one instance: 'asg1' (Instance ID: i-0c28c4710e88c2729), which is 'Running'. The instance details page for 'i-0c28c4710e88c2729' is shown, with tabs for 'Details', 'Status and alarms', 'Monitoring', 'Security', 'Networking', 'Storage', and 'Tags'. Under 'Details', it shows the 'Instance summary' with fields like 'Instance ID' (i-0c28c4710e88c2729), 'Public IPv4 address' (13.204.62.154), 'Private IPv4 addresses' (172.31.6.7), and 'Public DNS'.

Increasing Load On that instance for for check working using stress command

```
[root@ip-172-31-6-7 ~]# yum install stress
```

Increasing stress using that command

```
stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s&
```

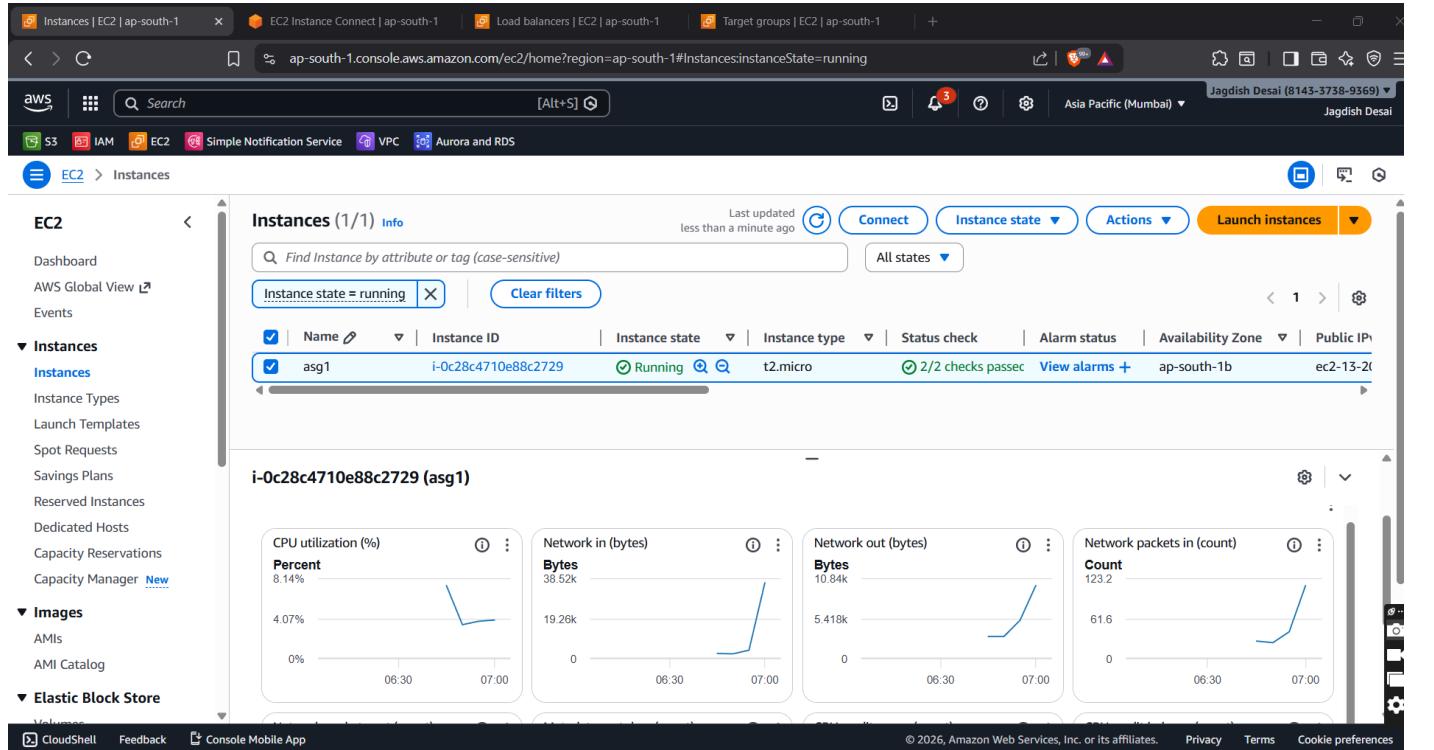


The screenshot shows a terminal window within the AWS CloudShell interface. The user has run the command `stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s &`. The output indicates that the stress test was successful and completed in 10 seconds. The terminal also shows the user's session history, including previous runs of the stress command.

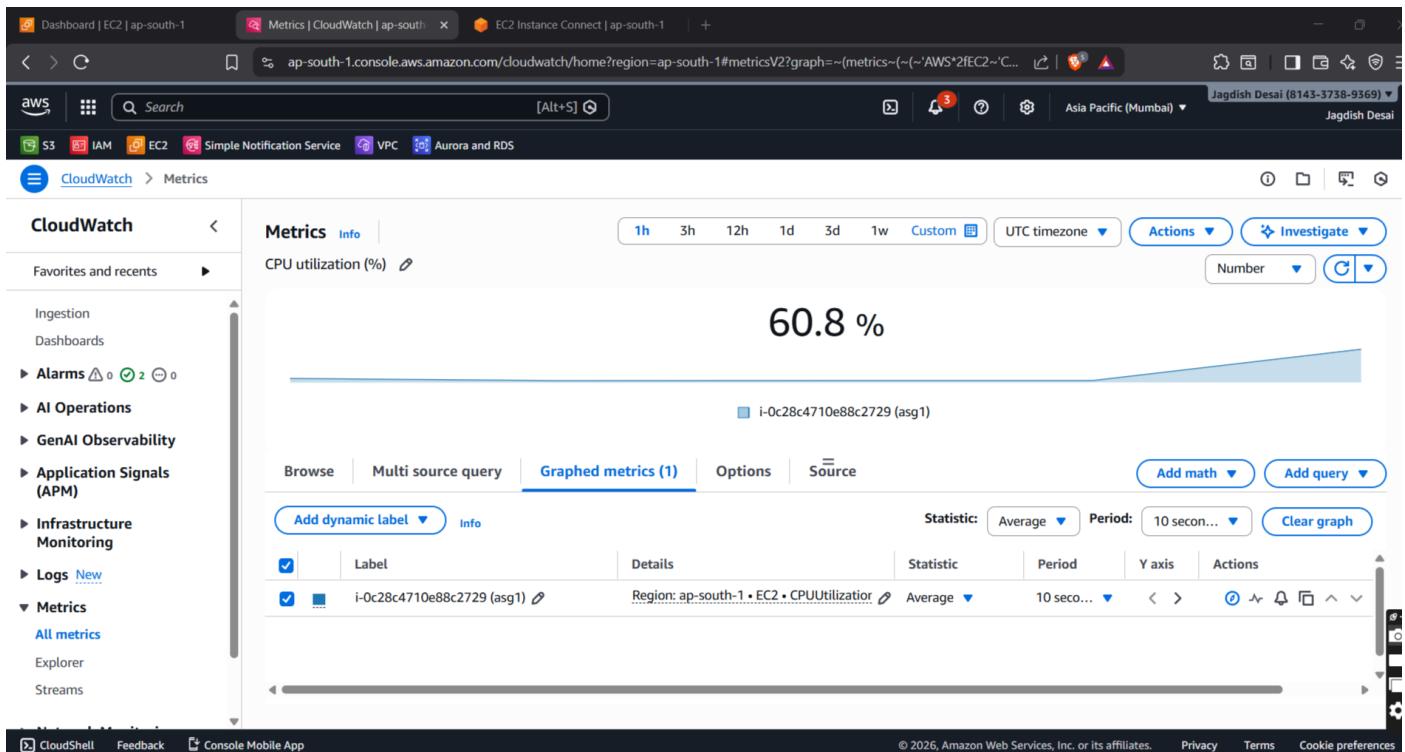
```
[root@ip-172-31-6-7 ~]# stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M --timeout 10s
stress: info: [27271] dispatching hogs: 8 cpu, 4 io, 2 vm, 0 hdd

stress: info: [27271] successful run completed in 10s
[root@ip-172-31-6-7 ~]#
[root@ip-172-31-6-7 ~]# stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s &
[1] 27334
[root@ip-172-31-6-7 ~]# stress: info: [27334] dispatching hogs: 80 cpu, 4 io, 2 vm, 0 hdd
jobs
[1]+  Running                  stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s &
[root@ip-172-31-6-7 ~]# 
```

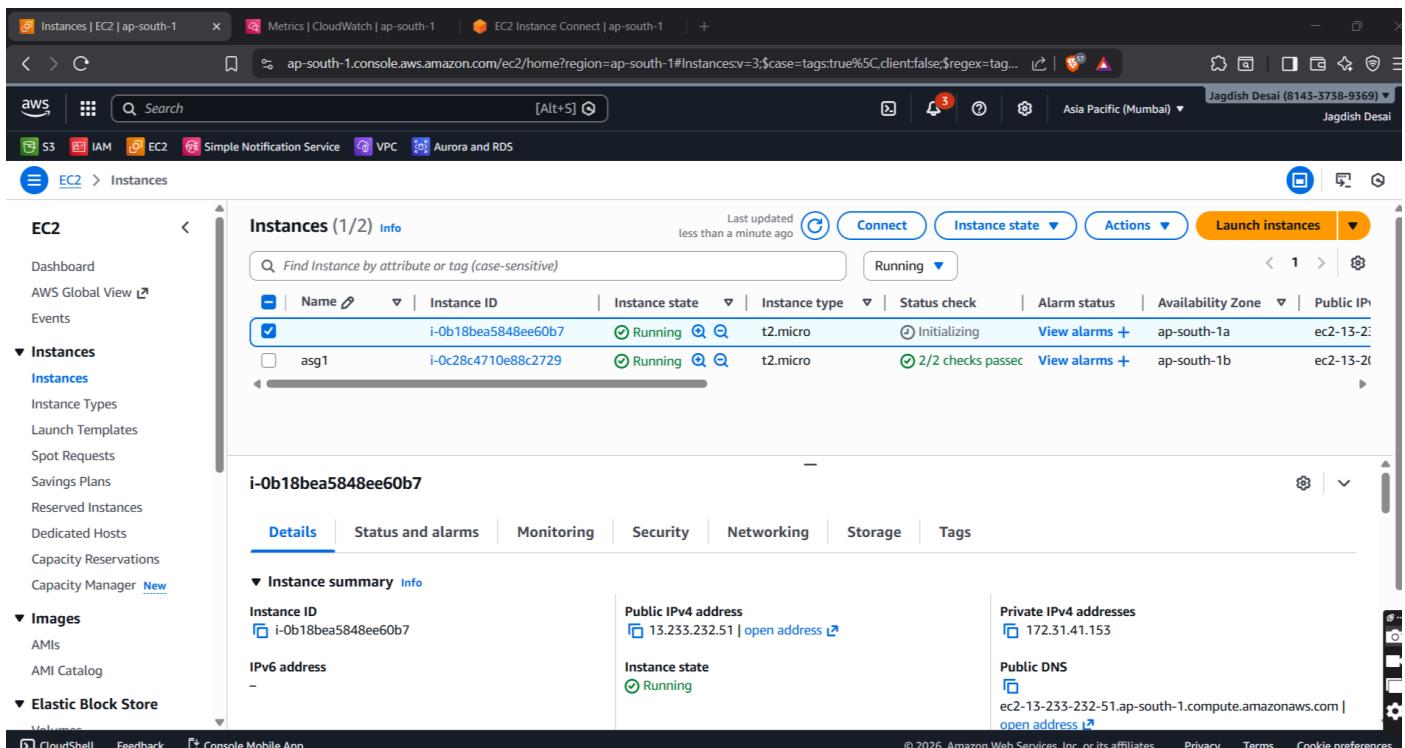
Then monitor cpu load.



Load Goes Above 50% so check instances because new instance will be creates



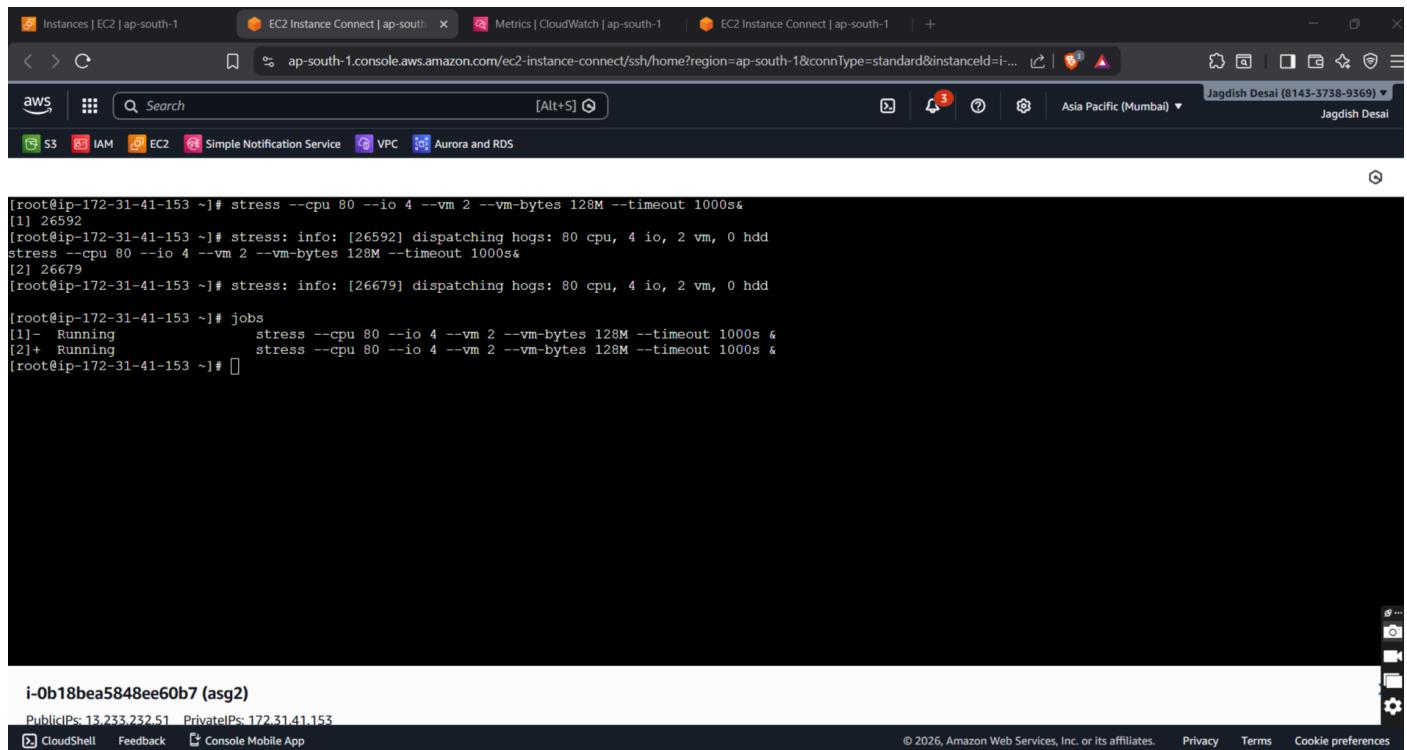
Checking instances



New Instance will Be automatically created after load will increases.

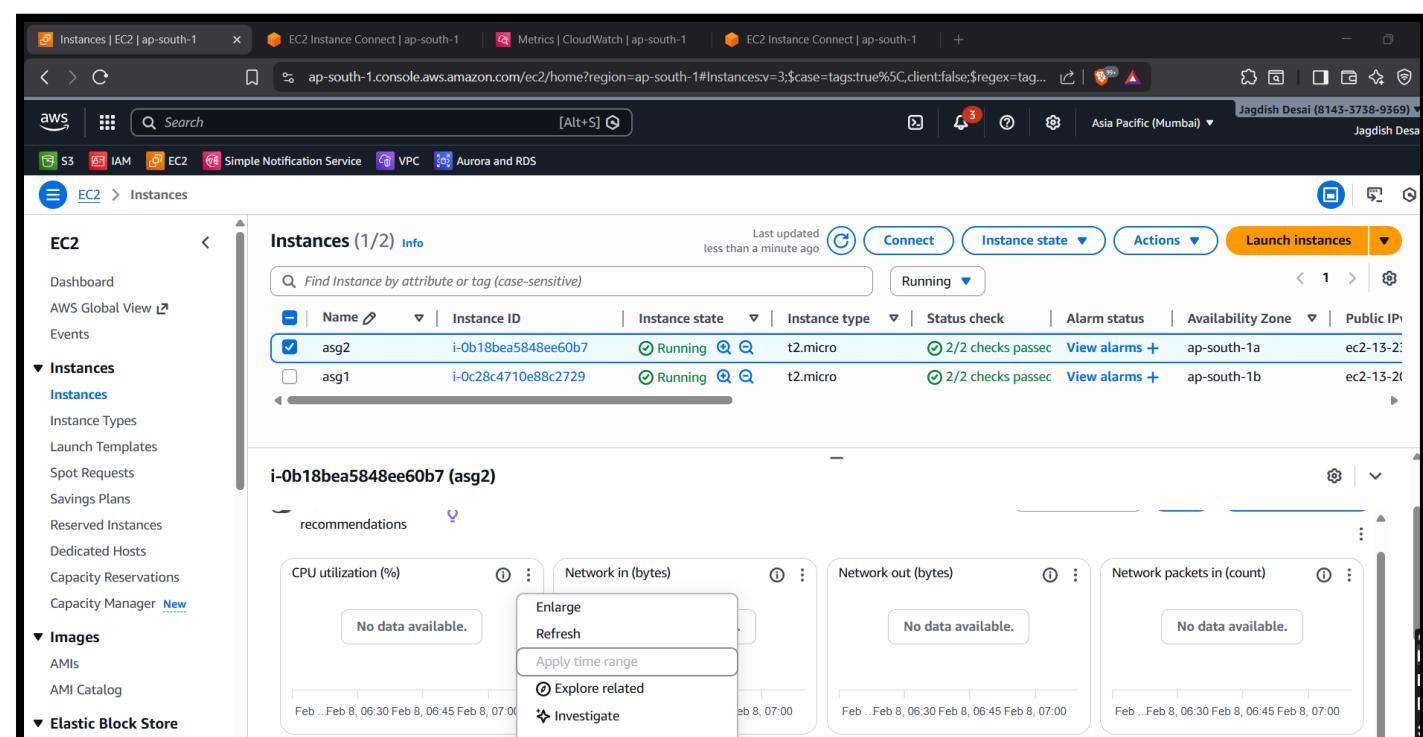
Connect to new instance and increasing load

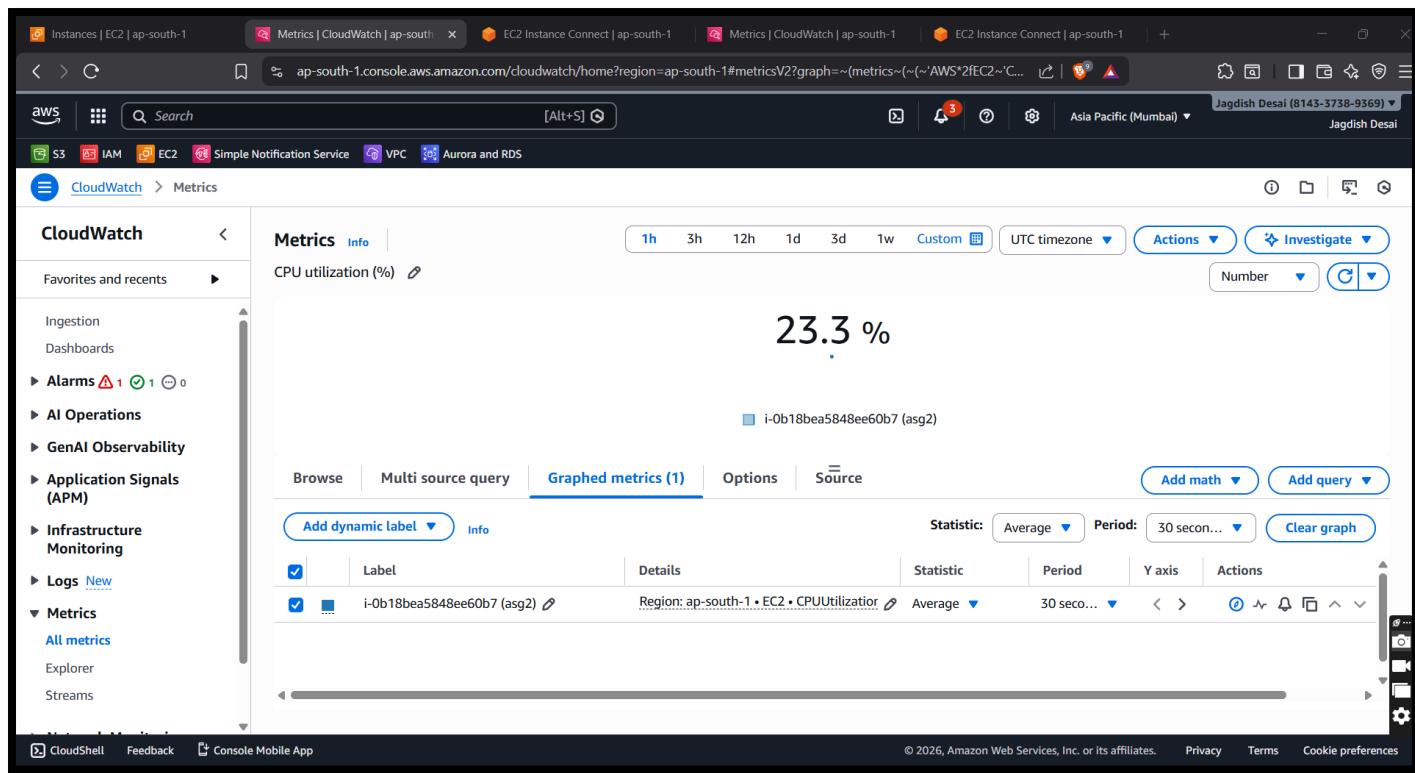
stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s&



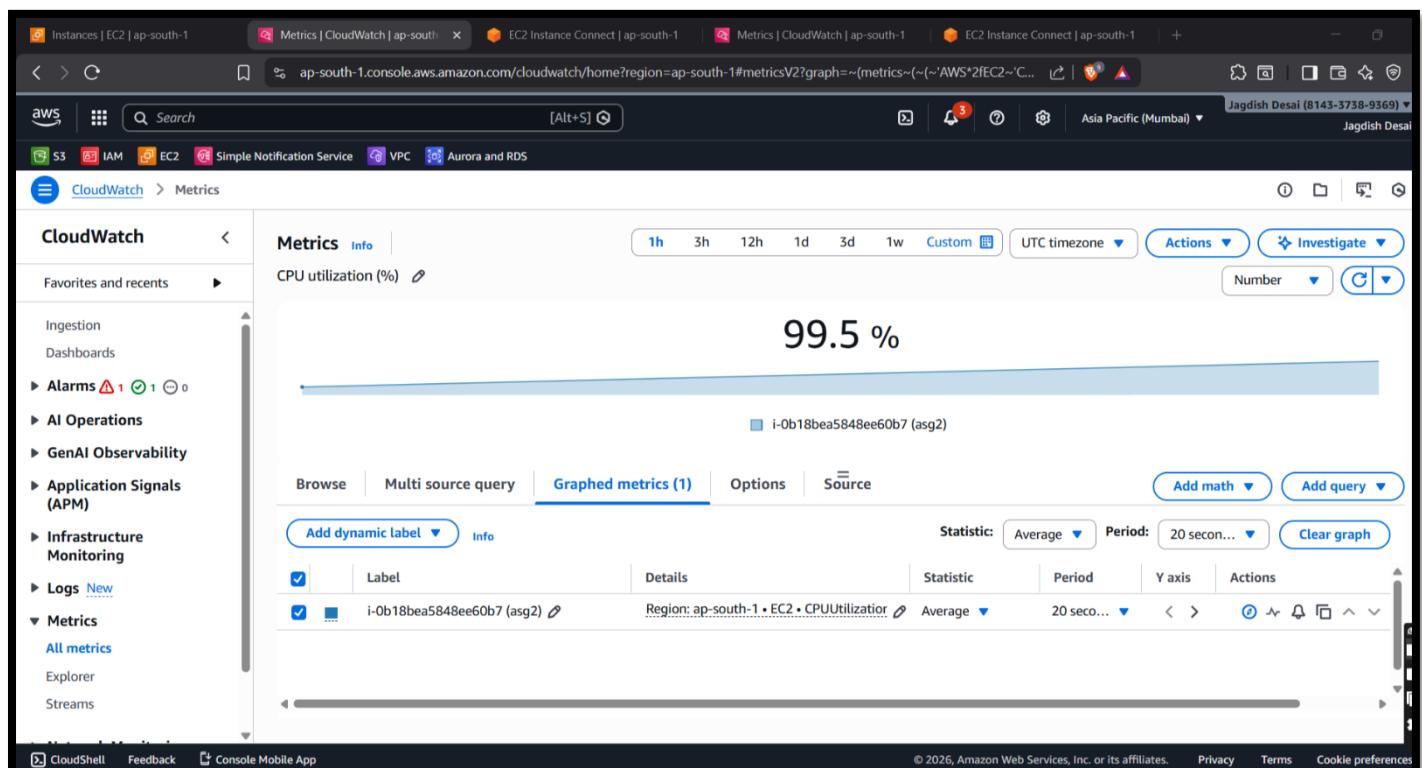
```
[root@ip-172-31-41-153 ~]# stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s&
[1] 26592
[root@ip-172-31-41-153 ~]# stress: info: [26592] dispatching hogs: 80 cpu, 4 io, 2 vm, 0 hdd
stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s&
[2] 26679
[root@ip-172-31-41-153 ~]# stress: info: [26679] dispatching hogs: 80 cpu, 4 io, 2 vm, 0 hdd
[root@ip-172-31-41-153 ~]# jobs
[1]-  Running                  stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s &
[2]+  Running                  stress --cpu 80 --io 4 --vm 2 --vm-bytes 128M --timeout 1000s &
[root@ip-172-31-41-153 ~]# 
```

Monitor Cpu Utilization





After Giving More Stress :



Go To Instances and check

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed, and the main area displays a table of instances. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. Three instances are listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
asg2	i-0b18bea5848ee60b7	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-2-
asg1	i-0c28c4710e88c2729	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-13-2-
i-01efef1c540ef0e48	i-01efef1c540ef0e48	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-15-2-

Below the table, the details for instance **i-01efef1c540ef0e48** are shown. The **Details** tab is selected, displaying the following information:

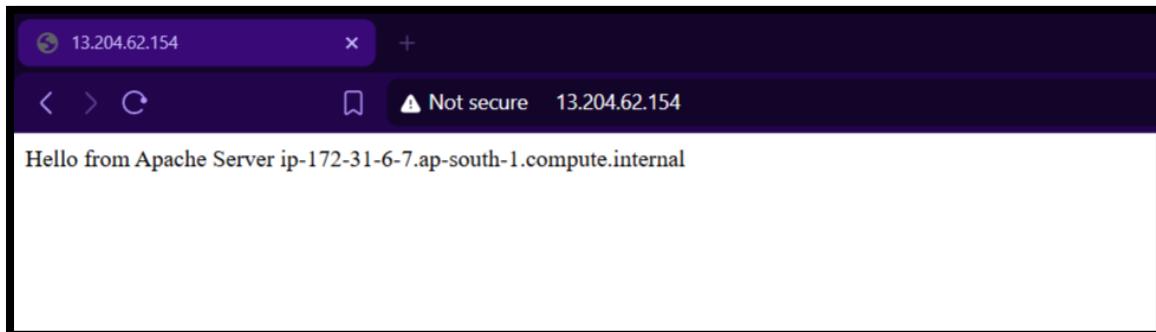
- Instance summary**:
 - Instance ID: i-01efef1c540ef0e48
 - IPv6 address: -
 - Public IPv4 address: 15.207.249.193 [open address]
 - Private IPv4 addresses: 172.31.8.111
 - Instance state: Running
 - Public DNS: ec2-15-207-249-193.ap-south-1.compute.amazonaws.com [open address]

New Instance will Be automatically created after load will increases.

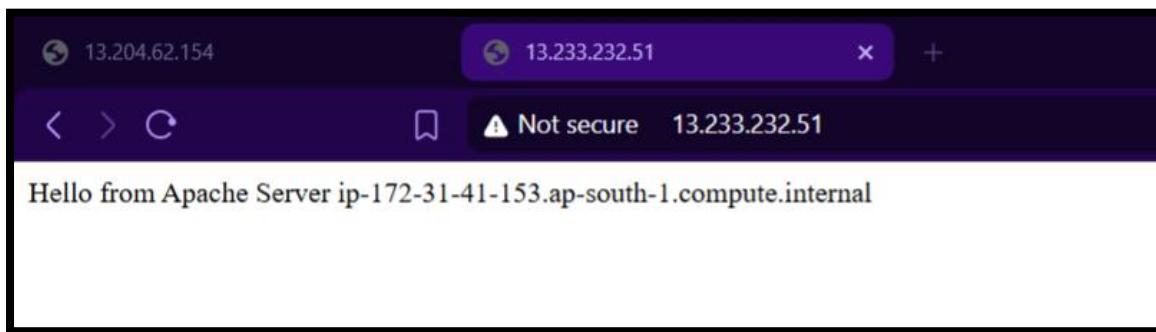
Asg1 ip - 13.204.62.154	private ip- 172.31.6.7
Asg2 ip - 13.233.232.51	private ip- 172.31.41.153
Asg3 ip - 15.207.249.193	private ip- 172.31.8.111

Checking Public Ip of each instance on browser:

Asg1 Instance:



Asg2 Instance:



Asg3 Instance:



It Shows and runs our script i.e hostname means Name of the machine (AWS uses private Ip).

ALB + ASG Setup workflow:

1. Launched EC2 / Prepare Launch Template

- Free-tier AMI (Amazon Linux 2 / Ubuntu)
- Instance type: t2.micro
- User Data: install Apache / app

2. Launched Template

3. Created Target Group

- Type: Instance
- Protocol: HTTP, Port 80
- Health check path: /

4. Created Application Load Balancer (ALB)

- Internet-facing, 2 public subnets
- Listener: HTTP → 80
- Attach Target Group

5. Created Auto Scaling Group (ASG)

- Select Launch Template
- VPC + 2 subnets
- Attach ALB + Target Group
- Capacity: Min 1, Desired 1, Max 2
- Scaling policy: CPU 50%

6. Verified Setup

- ALB DNS → App accessible
- Target Group → Healthy
- ASG → EC2 scales automatically

Over-Provisioning Means?

Over-provisioning = Allocating more resources (EC2 instances, CPU, memory) than actually needed.

In my project:

- If we manually launch 3 EC2 instances but our app only needs 1 → extra 2 instances are **unused** → you **pay unnecessarily**.
- This wastes money and is inefficient.

How Auto Scaling Helps

1. **Auto Scaling Group (ASG)** monitors CPU or traffic.
2. It **launches instances only when needed** (high traffic).
3. It **terminates instances when traffic is low**.

Result:

- No extra EC2 running → **avoids over-provisioning**
- Cost is optimized → you only pay for what is used

Example in our Project

Scenario	EC2 Instances Needed	Manually Launched	Auto Scaling
Low traffic	1	3 (over-provisioned)	1 (optimal)
High traffic	2	3	2–3 (scaled automatically)