Cycle 08 Homework Notes

Name: A Chetan Varma

Date: 22-07-2025

Auto Scaling Group (ASG)

Part 1: Hands-On ASG Lab

1. Launch an EC2 Instance

- AMI: Ubuntu (latest LTS)
- Instance type: t2.micro
- Security Group: Allow HTTP (port 80) and SSH (port 22)
- Commands to install Apache and serve a test page:

```
bash
CopyEdit
sudo apt update
sudo apt install apache2 -y
echo "<h1>Apache Test Page</h1>" | sudo tee /var/www/html/index.html
sudo systemctl enable apache2
sudo systemctl start apache2
```

- Validate by visiting `http://<EC2-Public-IP>`

---

2. Create a Custom AMI

- Stop the EC2 instance
- Go to EC2 → Actions → Image → Create Image
- Name: `My_Apache_AMI`

- Wait for image status to become available

---

3. Create a Launch Template

- Use the newly created AMI
- Instance type: t2.micro
- Security Group: Same as before (HTTP + SSH)
- Keep default settings for other options
- Save and create the launch template

---

4. Set Up an Auto Scaling Group (ASG)

- Use the launch template
- Choose at least 2 subnets in different Availability Zones
- ASG configuration:
  - Minimum capacity: 1
  - Desired capacity: 1
  - Maximum capacity: 3
- Health check type: EC2
- Health check grace period: 60 seconds

---

5. Configure Scaling Policy

- Add Target Tracking Policy:
  - Target value: 60 percent CPU utilization
  - Cooldown period: 120 seconds
- Optionally, install CloudWatch Agent for more metrics

---

6. Test Scaling Behavior

a. Spike CPU Load

- SSH into the running instance:

```
bash
CopyEdit
sudo apt install stress -y
stress -c 2
```

- Monitor EC2 console → ASG should launch a new instance if CPU > 60% for 2 minutes

b. Terminate Instance Manually

- Terminate an instance from ASG manually
- ASG should automatically launch a replacement to maintain desired count

---

Part 2: Advanced ASG Experiment

1. Scale Based on HTTP Requests (ALB + ASG)

- Create an Application Load Balancer (ALB)
  - At least 2 subnets
  - Target group: Register ASG
- Attach ALB to the ASG
- Create scaling policy:
  - Metric: ALB RequestCountPerTarget
  - Target value: Example 50 requests per minute
- Send test traffic using tools like Apache Benchmark:

```
bash
CopyEdit
```

```
ab -n 1000 -c 100 http://<ALB-DNS-Name>/
```

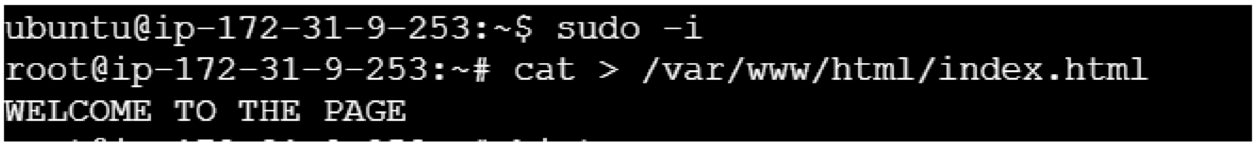
2. Multi-AMI Testing

- Create a second EC2 instance with Nginx:

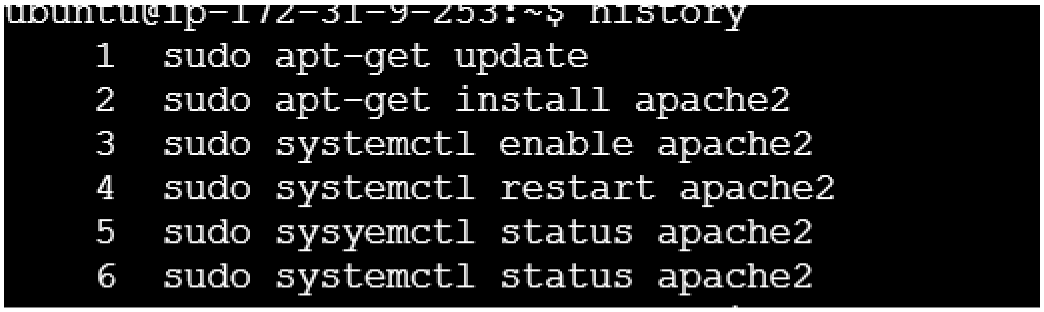
```
bash
CopyEdit
sudo apt update
sudo apt install nginx -y
echo "Nginx Test Page" | sudo tee /var/www/html/index.html
```

- Stop instance and create AMI: `My_Nginx_AMI`
- Create another Launch Template using this new AMI
- Create a separate ASG to compare Apache vs Nginx scaling behavior

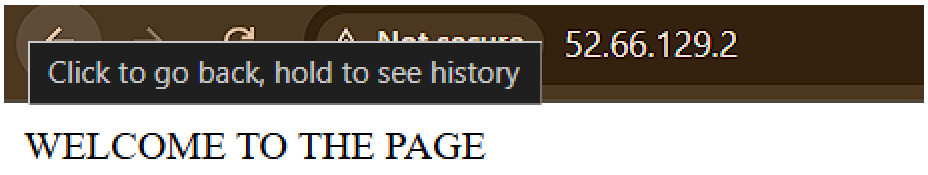
SCREENSHOTS:



```
ubuntu@ip-172-31-9-253:~$ sudo -i
root@ip-172-31-9-253:~# cat > /var/www/html/index.html
WELCOME TO THE PAGE
```

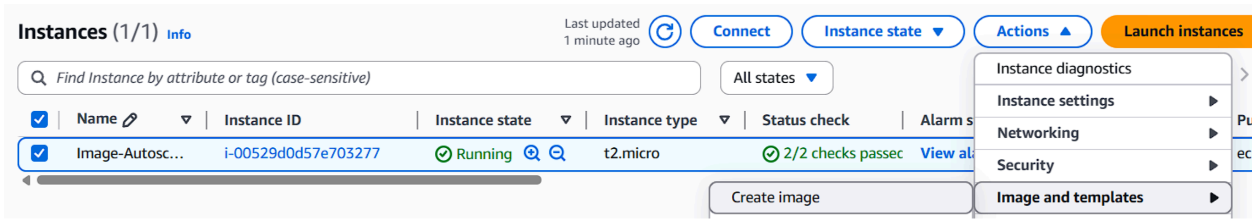


```
ubuntu@ip-172-31-9-253:~$ history
    1  sudo apt-get update
    2  sudo apt-get install apache2
    3  sudo systemctl enable apache2
    4  sudo systemctl restart apache2
    5  sudo sysyemctl status apache2
    6  sudo systemctl status apache2
```


Not secure 52.66.129.2

Click to go back, hold to see history

WELCOME TO THE PAGE

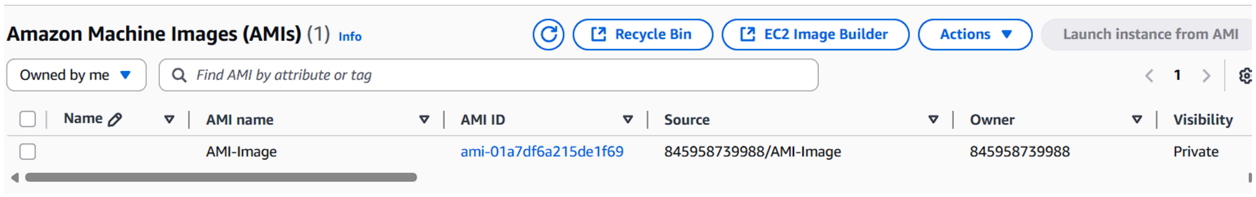

Instances (1/1) Info

Last updated 1 minute ago

Connect | Instance state ▼ | Actions ▲ | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states ▼

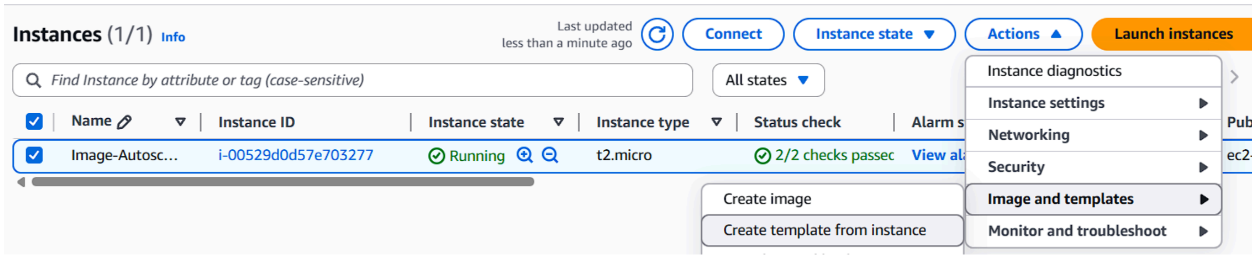
Name	Instance ID	Instance state	Instance type	Status check	Alarm s
Image-Autosc…	i-00529d0d57e703277	Running	t2.micro	2/2 checks passec	View al


Amazon Machine Images (AMIs) (1) Info

Recycle Bin | EC2 Image Builder | Actions ▼ | Launch instance from AMI

Owned by me ▼ | Find AMI by attribute or tag

Name	AMI name	AMI ID	Source	Owner	Visibility
	AMI-Image	ami-01a7df6a215de1f69	845958739988/AMI-Image	845958739988	Private

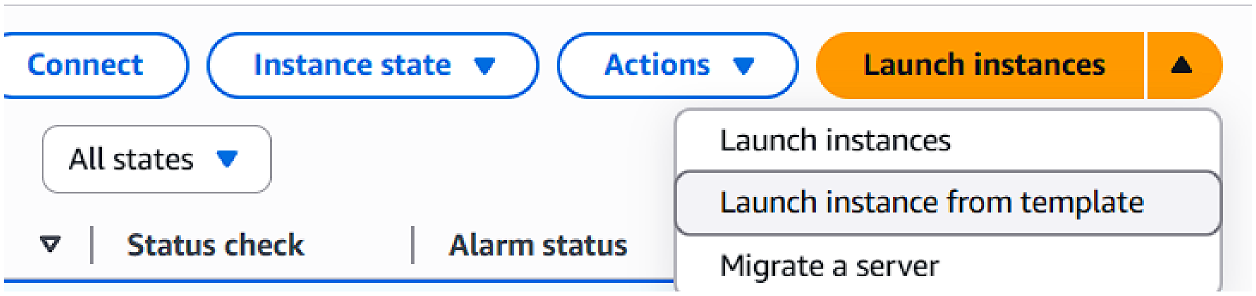

Instances (1/1) Info

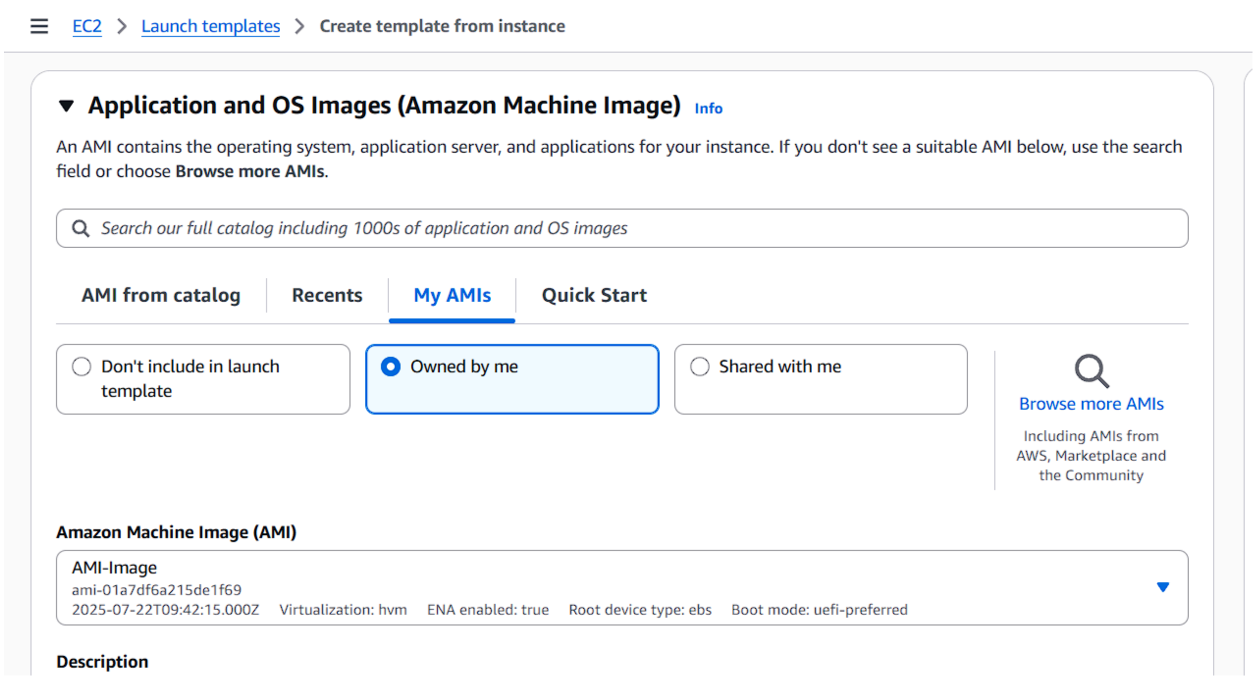
Last updated less than a minute ago

Connect | Instance state ▼ | Actions ▲ | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states ▼

Name	Instance ID	Instance state	Instance type	Status check	Alarm s
Image-Autosc…	i-00529d0d57e703277	Running	t2.micro	2/2 checks passec	View al




EC2 > Launch templates > Create template from instance

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Search our full catalog including 1000s of application and OS images

AMI from catalog | Recents | My AMIs | Quick Start

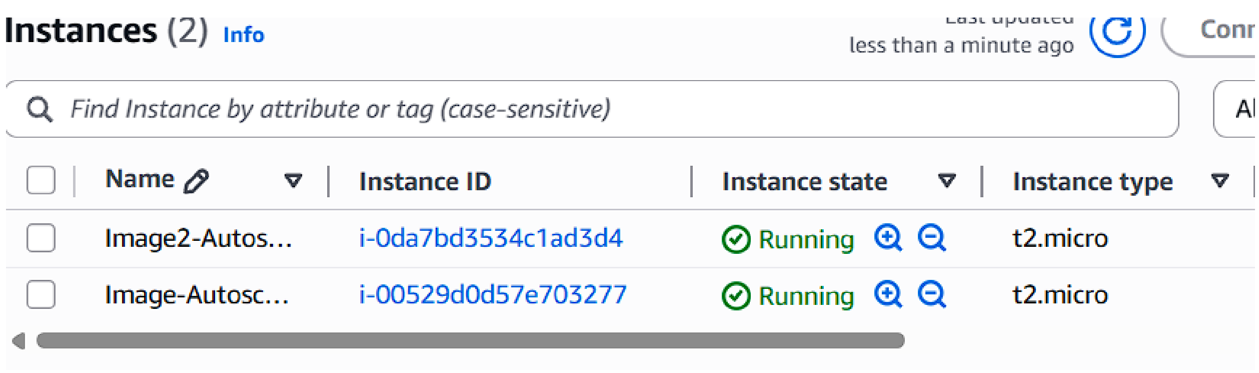
( ) Don't include in launch template [✓ Owned by me] ( ) Shared with me

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

AMI-Image
ami-01a7df6a215de1f69
2025-07-22T09:42:15.000Z Virtualization: hvm ENA enabled: true Root device type: ebs Boot mode: uefi-preferred

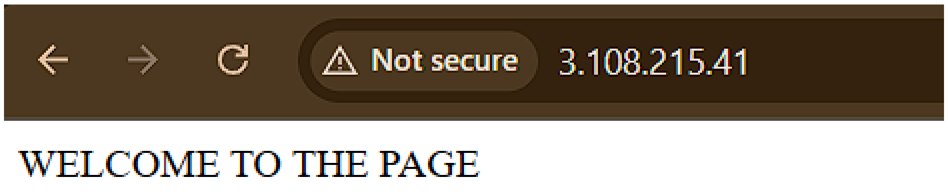
Description


Instances (2) Info

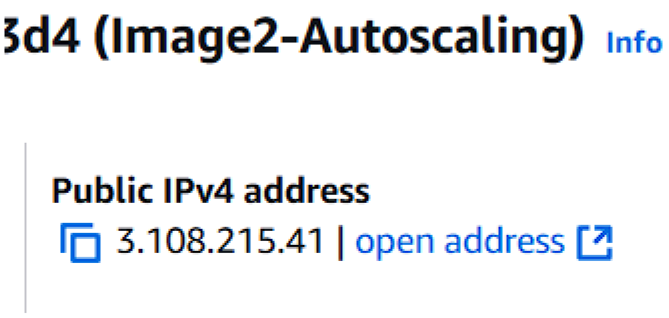
Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type
Image2-Autos…	i-0da7bd3534c1ad3d4	Running	t2.micro
Image-Autosc…	i-00529d0d57e703277	Running	t2.micro

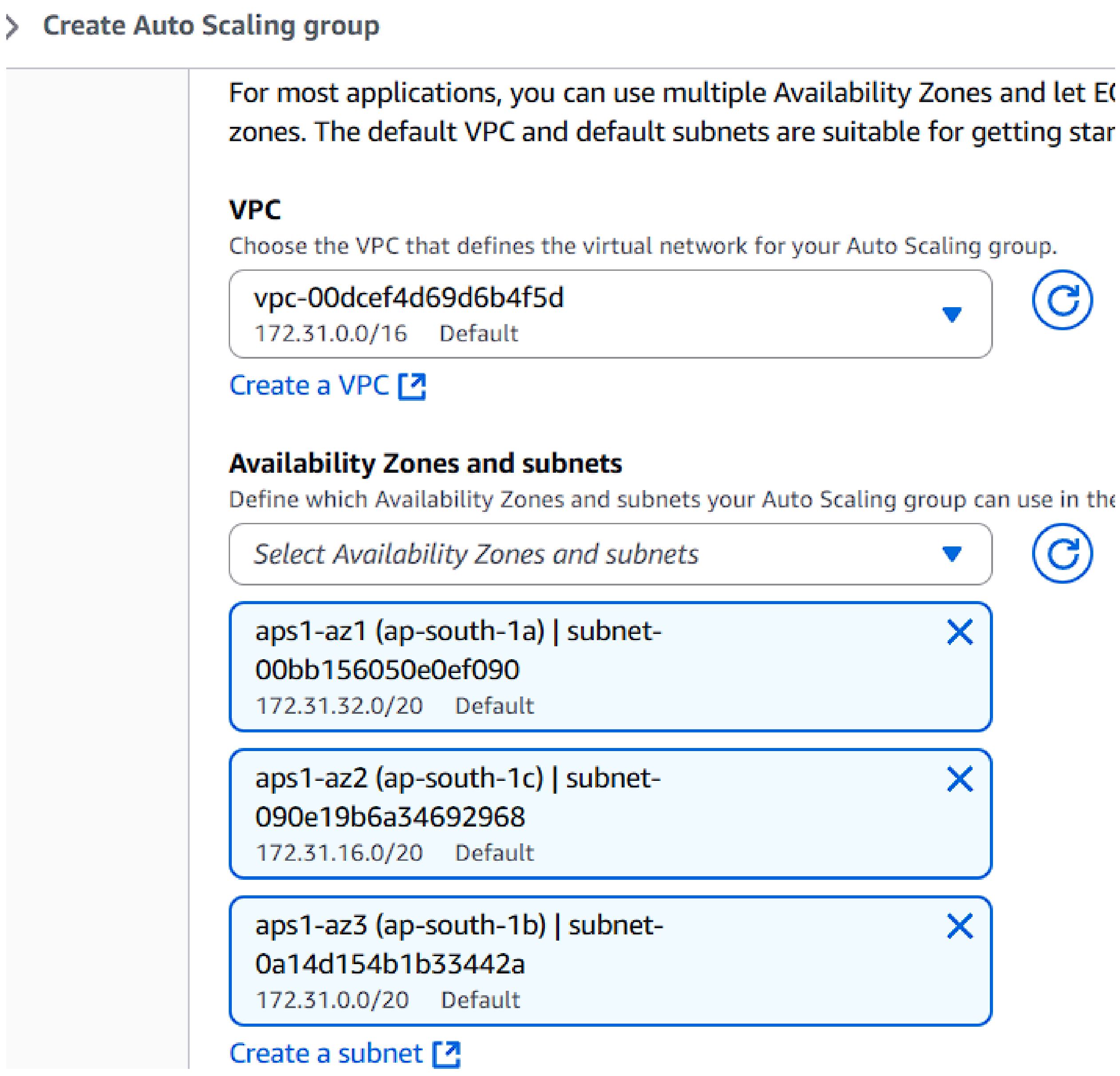

Not secure 3.108.215.41

WELCOME TO THE PAGE


3d4 (Image2-Autoscaling) Info

Public IPv4 address

3.108.215.41 | open address


Create Auto Scaling group

For most applications, you can use multiple Availability Zones and let E
zones. The default VPC and default subnets are suitable for getting star

VPC

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-00dcef4d69d6b4f5d
172.31.0.0/16 Default

Create a VPC

Availability Zones and subnets

Define which Availability Zones and subnets your Auto Scaling group can use in the

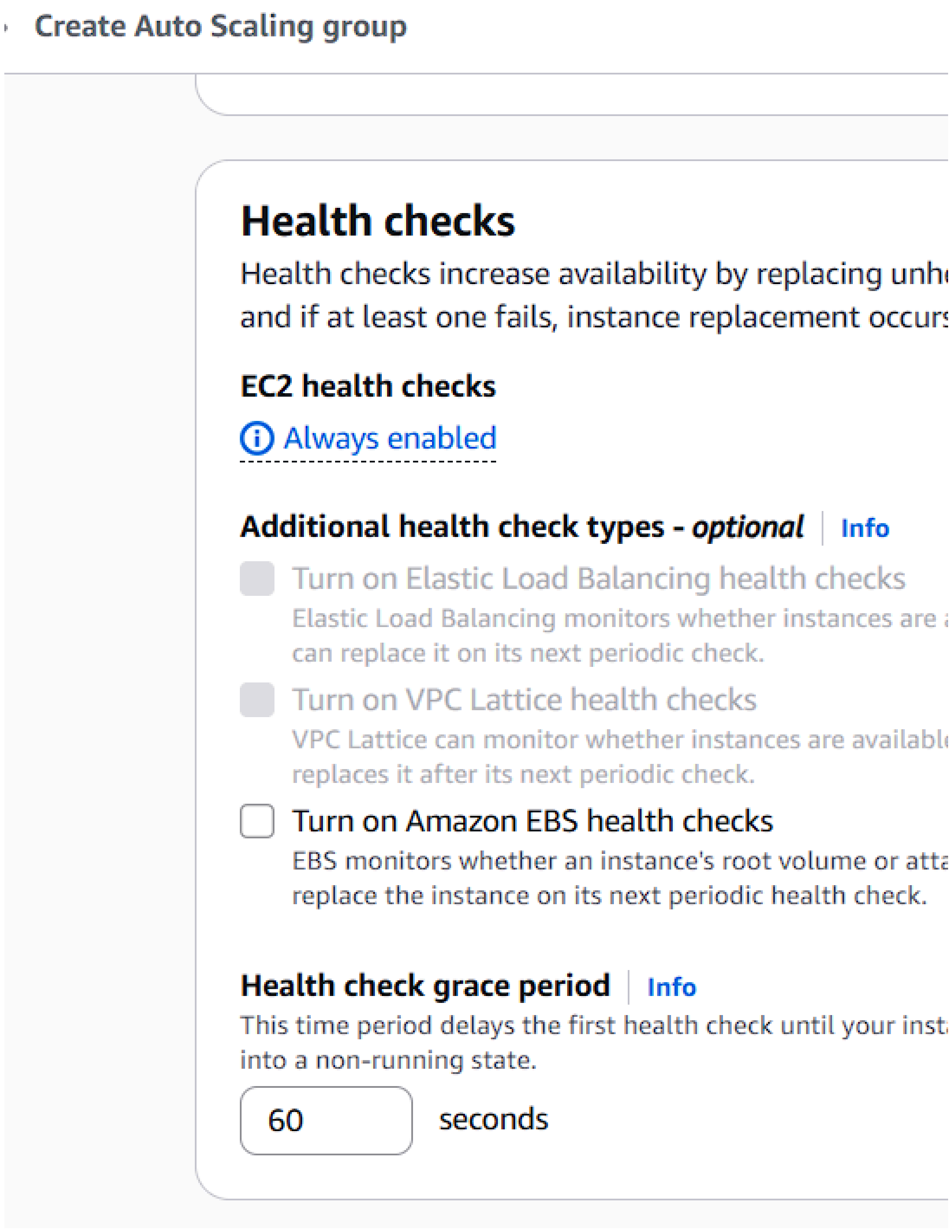
Select Availability Zones and subnets

aps1-az1 (ap-south-1a) | subnet-00bb156050e0ef090
172.31.32.0/20 Default

aps1-az2 (ap-south-1c) | subnet-090e19b6a34692968
172.31.16.0/20 Default

aps1-az3 (ap-south-1b) | subnet-0a14d154b1b33442a
172.31.0.0/20 Default

Create a subnet


Create Auto Scaling group

Health checks

Health checks increase availability by replacing unh
and if at least one fails, instance replacement occur

EC2 health checks

Always enabled

Additional health check types - *optional* Info

☐ Turn on Elastic Load Balancing health checks
Elastic Load Balancing monitors whether instances are
can replace it on its next periodic check.

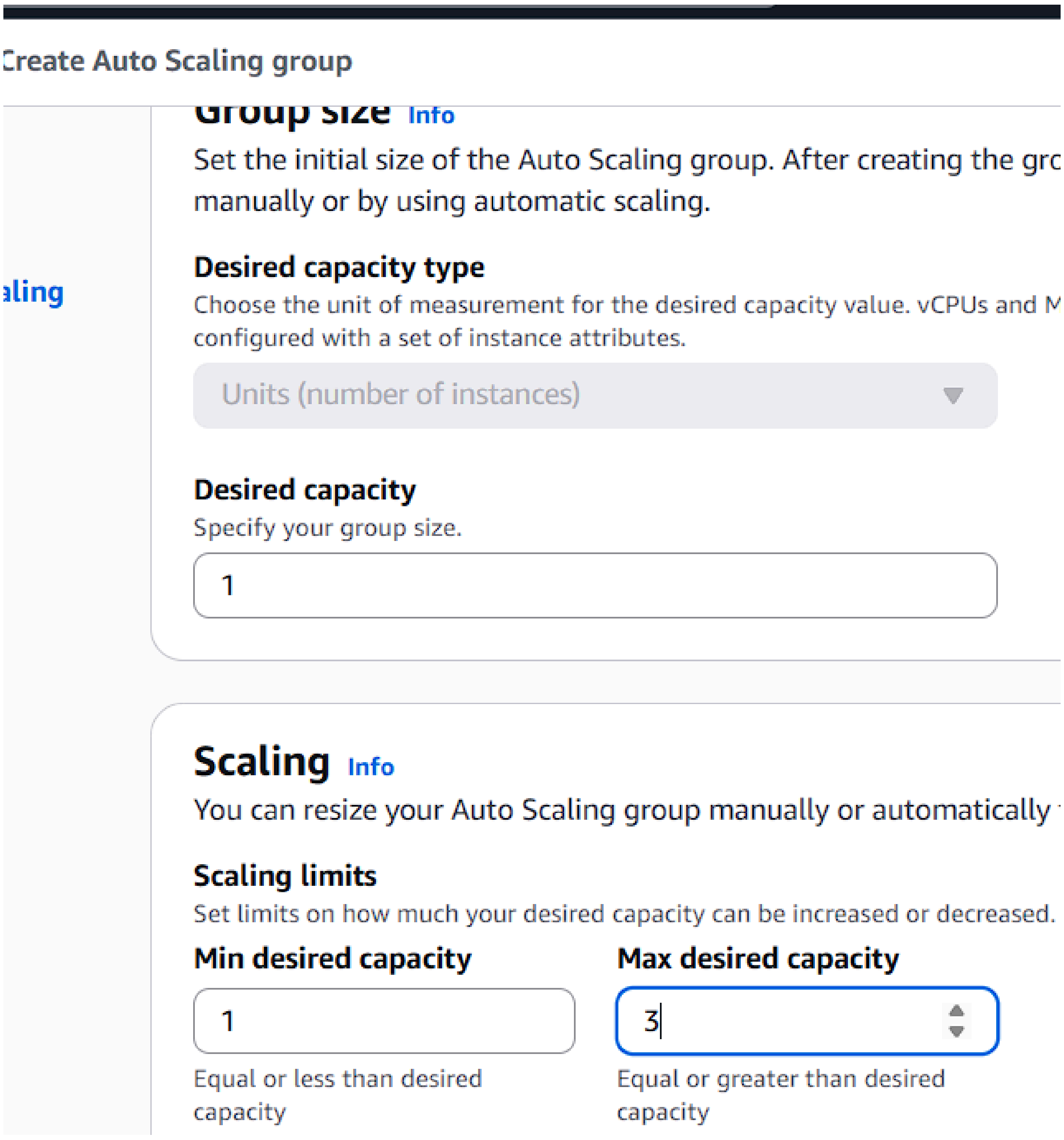
☐ Turn on VPC Lattice health checks
VPC Lattice can monitor whether instances are availabl
replaces it after its next periodic check.

☐ Turn on Amazon EBS health checks
EBS monitors whether an instance's root volume or att
replace the instance on its next periodic health check.

Health check grace period Info

This time period delays the first health check until your inst
into a non-running state.

60 seconds


Create Auto Scaling group

aling

Group size Info

Set the initial size of the Auto Scaling group. After creating the grc manually or by using automatic scaling.

Desired capacity type

Choose the unit of measurement for the desired capacity value. vCPUs and M configured with a set of instance attributes.

Units (number of instances)

Desired capacity

Specify your group size.

1

Scaling Info

You can resize your Auto Scaling group manually or automatically

Scaling limits

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

Min desired capacity

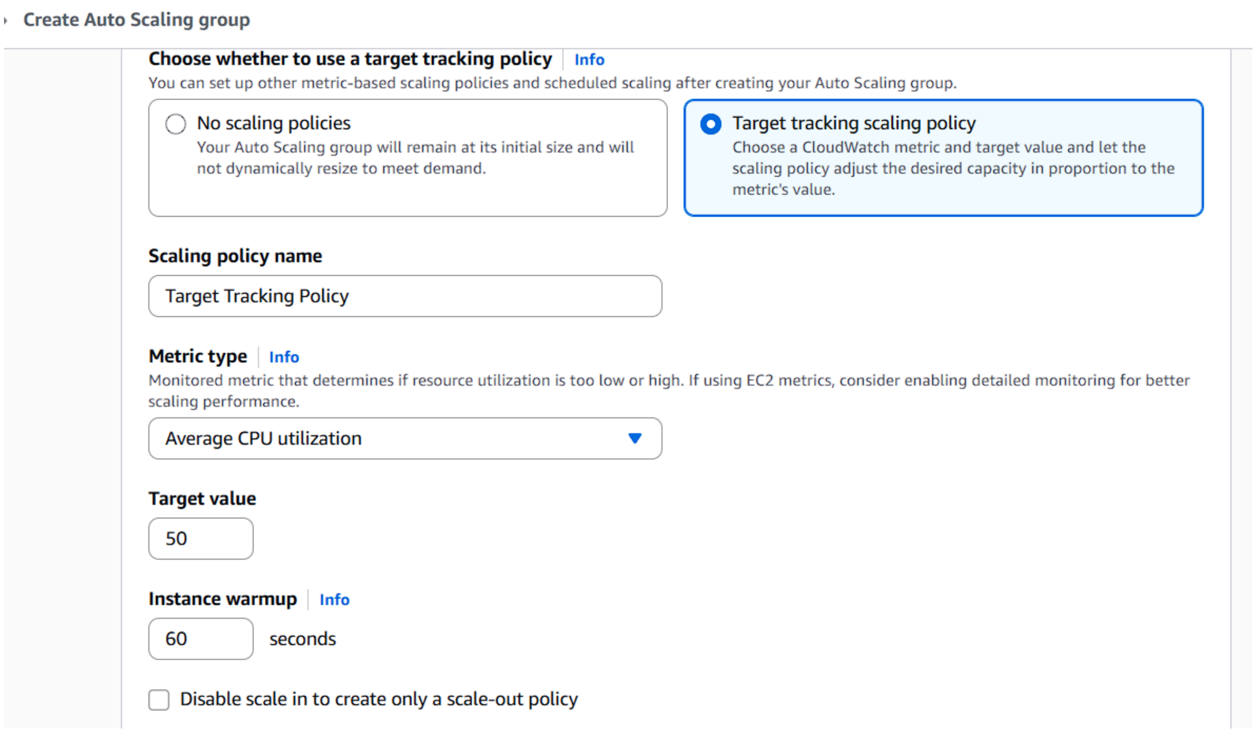
1

Equal or less than desired capacity

Max desired capacity

3

Equal or greater than desired capacity



Create Auto Scaling group

Choose whether to use a target tracking policy Info

You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

- 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 Info

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

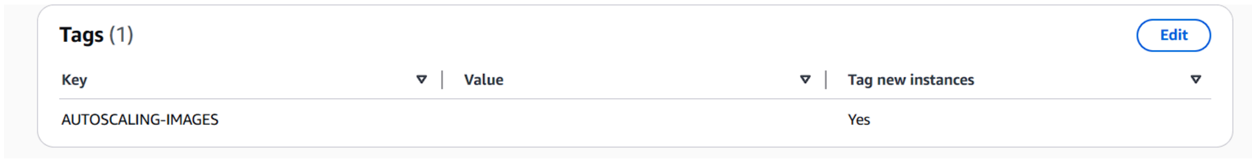
Target value

50

Instance warmup Info

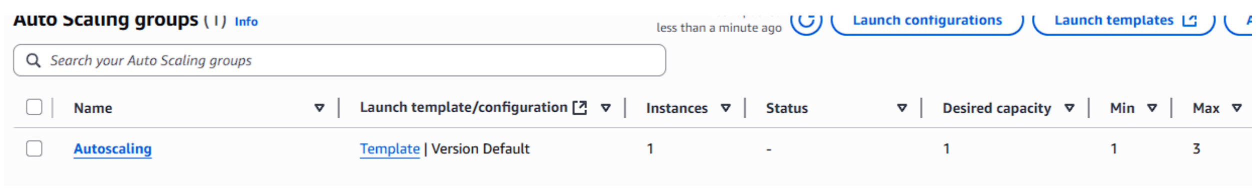
60 seconds

☐ Disable scale in to create only a scale-out policy



Tags (1)

Key	Value	Tag new instances
AUTOSCALING-IMAGES		Yes

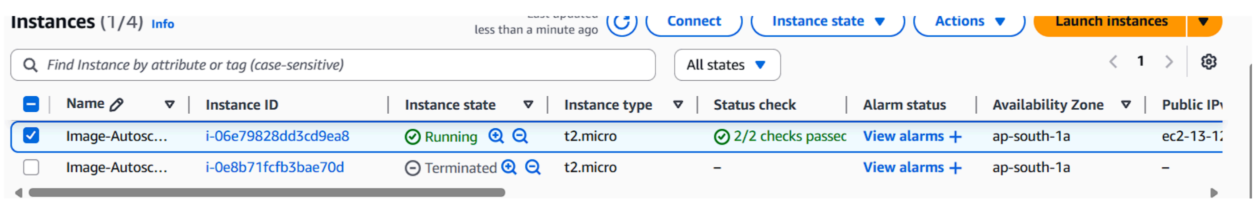


Auto Scaling groups (1) Info

less than a minute ago | Launch configurations | Launch templates

Search your Auto Scaling groups

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max
Autoscaling	Template | Version Default	1	-	1	1	3

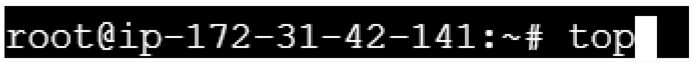


Instances (1/4) Info

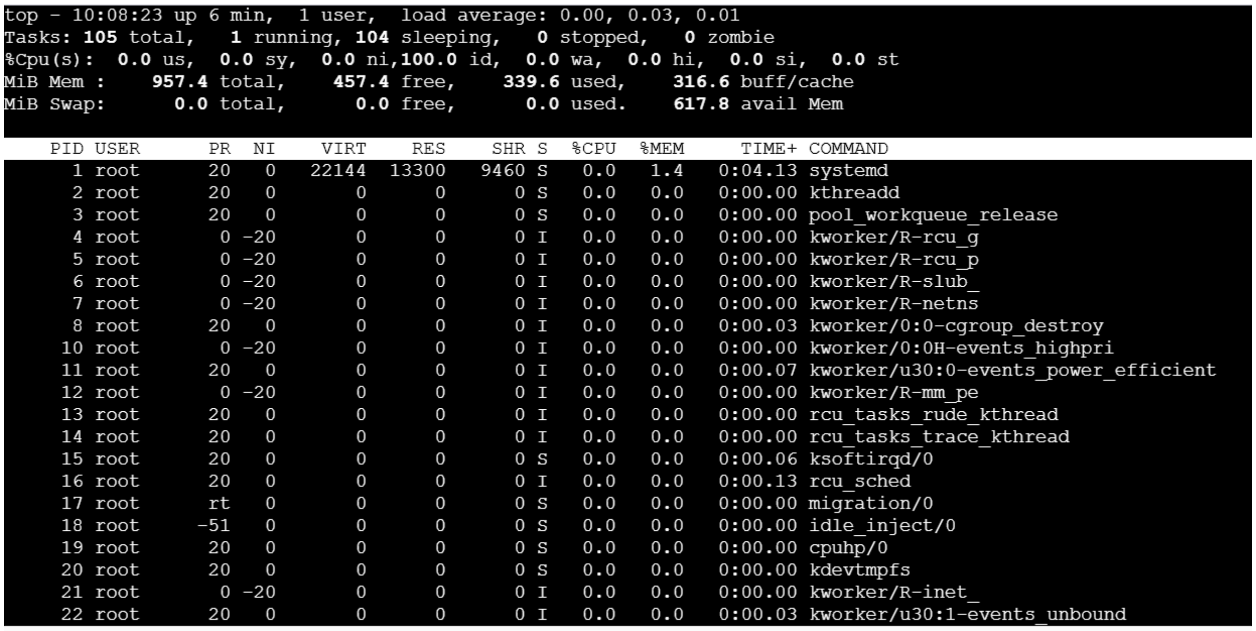
less than a minute ago | Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states

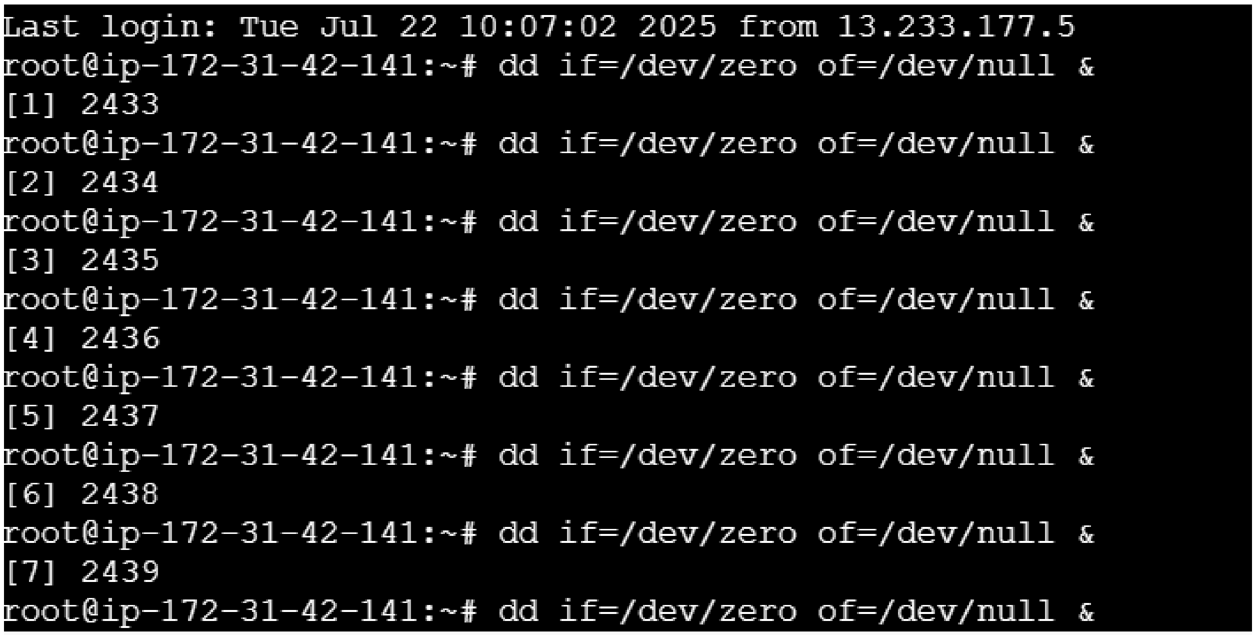
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
Image-Autosc...	i-06e79828dd3cd9ea8	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1a	ec2-13-1
Image-Autosc...	i-0e8b71fcfb3bae70d	Terminated	t2.micro	–	View alarms +	ap-south-1a	–



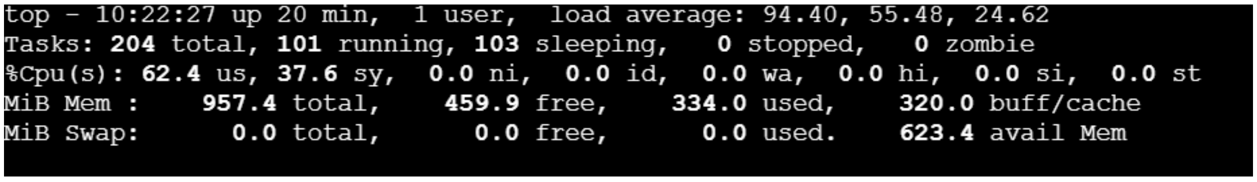
```
root@ip-172-31-42-141:~# top
```



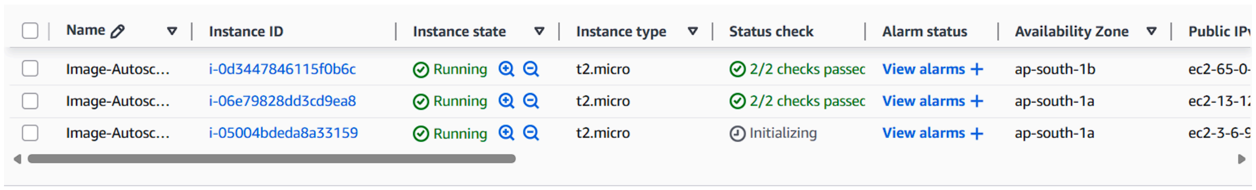
```
top - 10:08:23 up 6 min,  1 user,  load average: 0.00, 0.03, 0.01
Tasks: 105 total,   1 running, 104 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    957.4 total,    457.4 free,    339.6 used,    316.6 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    617.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0   22144  13300   9460 S   0.0   1.4   0:04.13 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_release
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_p
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-slub_
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-netns
      8 root      20   0       0      0      0 I   0.0   0.0   0:00.03 kworker/0:0-cgroup_destroy
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.07 kworker/u30:0-events_power_efficient
     12 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-mm_pe
     13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
     14 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.06 ksoftirqd/0
     16 root      20   0       0      0      0 I   0.0   0.0   0:00.13 rcu_sched
     17 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/0
     18 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     20 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
     21 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-inet_
     22 root      20   0       0      0      0 I   0.0   0.0   0:00.03 kworker/u30:1-events_unbound
```



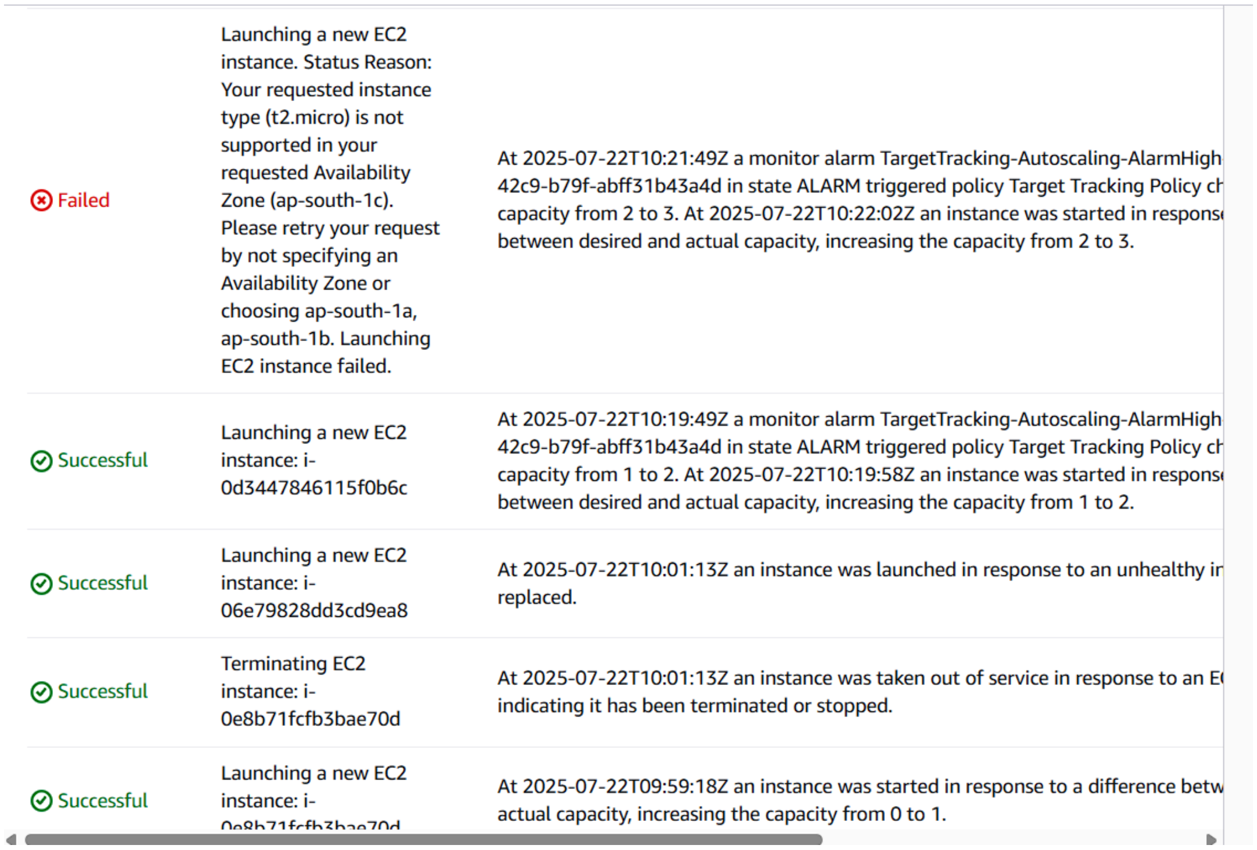
```
Last login: Tue Jul 22 10:07:02 2025 from 13.233.177.5
root@ip-172-31-42-141:~# dd if=/dev/zero of=/dev/null &
[1] 2433
root@ip-172-31-42-141:~# dd if=/dev/zero of=/dev/null &
[2] 2434
root@ip-172-31-42-141:~# dd if=/dev/zero of=/dev/null &
[3] 2435
root@ip-172-31-42-141:~# dd if=/dev/zero of=/dev/null &
[4] 2436
root@ip-172-31-42-141:~# dd if=/dev/zero of=/dev/null &
[5] 2437
root@ip-172-31-42-141:~# dd if=/dev/zero of=/dev/null &
[6] 2438
root@ip-172-31-42-141:~# dd if=/dev/zero of=/dev/null &
[7] 2439
root@ip-172-31-42-141:~# dd if=/dev/zero of=/dev/null &
```



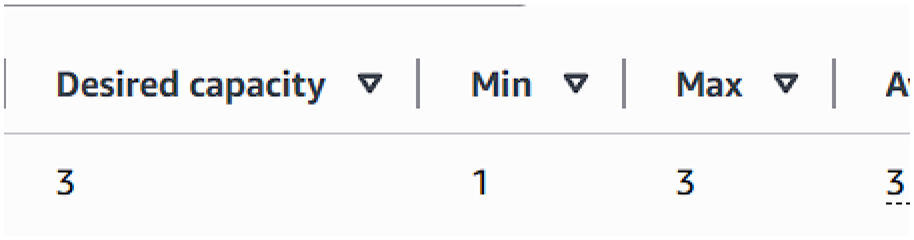
```
top - 10:22:27 up 20 min,  1 user,  load average: 94.40, 55.48, 24.62
Tasks: 204 total, 101 running, 103 sleeping,   0 stopped,   0 zombie
%Cpu(s): 62.4 us, 37.6 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    957.4 total,    459.9 free,    334.0 used,    320.0 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    623.4 avail Mem
```



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
☐	Image-Autosc…	i-0d3447846115f0b6c	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	ap-south-1b	ec2-65-0-
☐	Image-Autosc…	i-06e79828dd3cd9ea8	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	ap-south-1a	ec2-13-1
☐	Image-Autosc…	i-05004bdeda8a33159	✓ Running	t2.micro	Initializing	View alarms +	ap-south-1a	ec2-3-6-



⊗ Failed	Launching a new EC2 instance. Status Reason: Your requested instance type (t2.micro) is not supported in your requested Availability Zone (ap-south-1c). Please retry your request by not specifying an Availability Zone or choosing ap-south-1a, ap-south-1b. Launching EC2 instance failed.	At 2025-07-22T10:21:49Z a monitor alarm TargetTracking-Autoscaling-AlarmHigh 42c9-b79f-abff31b43a4d in state ALARM triggered policy Target Tracking Policy ch capacity from 2 to 3. At 2025-07-22T10:22:02Z an instance was started in respons between desired and actual capacity, increasing the capacity from 2 to 3.
✓ Successful	Launching a new EC2 instance: i-0d3447846115f0b6c	At 2025-07-22T10:19:49Z a monitor alarm TargetTracking-Autoscaling-AlarmHigh 42c9-b79f-abff31b43a4d in state ALARM triggered policy Target Tracking Policy ch capacity from 1 to 2. At 2025-07-22T10:19:58Z an instance was started in respons between desired and actual capacity, increasing the capacity from 1 to 2.
✓ Successful	Launching a new EC2 instance: i-06e79828dd3cd9ea8	At 2025-07-22T10:01:13Z an instance was launched in response to an unhealthy in replaced.
✓ Successful	Terminating EC2 instance: i-0e8b71fcfb3bae70d	At 2025-07-22T10:01:13Z an instance was taken out of service in response to an E indicating it has been terminated or stopped.
✓ Successful	Launching a new EC2 instance: i-0e8b71fcfb3bae70d	At 2025-07-22T09:59:18Z an instance was started in response to a difference betw actual capacity, increasing the capacity from 0 to 1.



Desired capacity	Min	Max	A
3	1	3	3

Part 3: Troubleshooting Tasks

1. Simulate Failed Health Check

- SSH into instance and stop Apache:

```
bash
CopyEdit
sudo systemctl stop apache2
```

- Within a minute, ASG should detect health check failure and terminate the instance
- A new healthy instance should be launched automatically

---

2. Analyze CloudWatch Metrics

- Go to CloudWatch → Metrics → EC2 → Per-Instance Metrics
- Look for:
    - CPUUtilization
    - GroupDesiredCapacity
    - GroupInServiceInstances
    - GroupTerminatingInstances
- Verify spikes during stress test and scaling events

EOD Report Suggestions

- Document screenshots:
    - Apache test page
    - AMI creation
    - Launch template
    - ASG settings
    - Scaling events in EC2 and CloudWatch
- Learning Outcomes:

- Understood autoscaling fundamentals: health checks, templates, and policies
- Practiced scaling based on CPU and HTTP requests
- Learned troubleshooting using service stop and CloudWatch metrics