

# Day 2 Group Activity: AWS Auto Scaling and Load Balancing

GROUP 6

WEEK 5

# OBJECTIVES

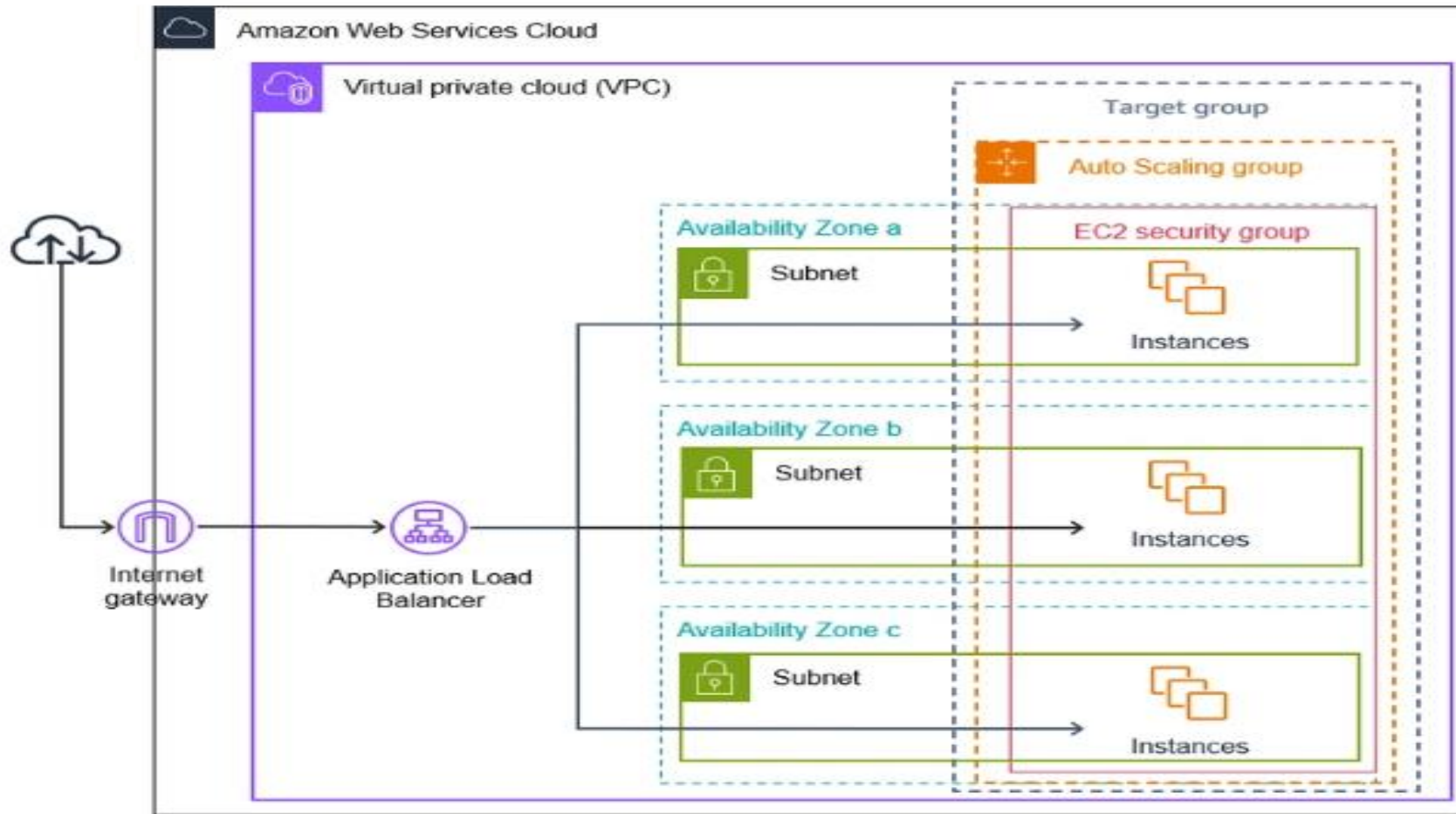
To:

- ▶ Understand how auto scaling works and why it is important for dynamic scaling
- ▶ Setup and configure an Elastic Load Balancer (ELB) and implement Auto Scaling for fault tolerance.
- ▶ Test the system under load to verify automatic scaling behavior.

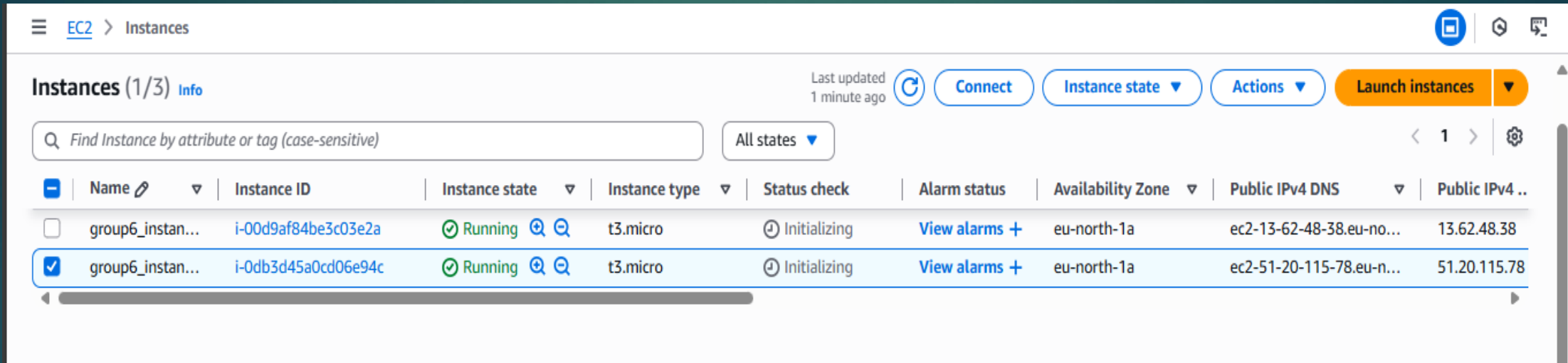
# GENERAL OVERVIEW

- ▶ Auto Scaling and Load Balancing work together to create a reliable and efficient cloud infrastructure. **Auto Scaling Groups (ASG)** automatically adjust the number of EC2 instances based on demand, optimizing performance and cost while **Load Balancers (ALB or NLB)** distribute incoming traffic evenly across instances, preventing overloads and improving reliability.
- ▶ This process includes defining scaling policies, setting up CloudWatch alarms, testing load distribution, simulating real-world traffic conditions, and monitoring system performance. By using these strategies, cloud systems can stay adaptable, cost-effective, and resilient, ensuring smooth application availability even during high traffic periods.

# Architectural Overview



# 1.LAUNCHING EC2 INSTANCES

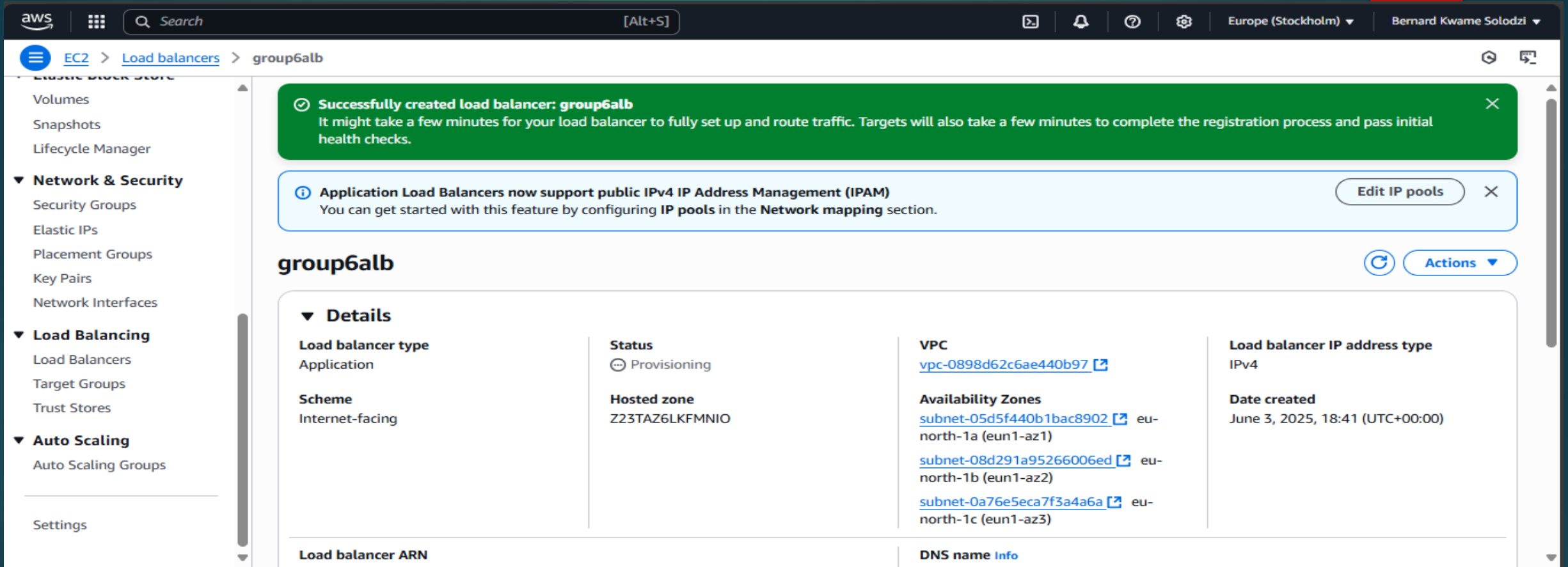


The screenshot displays the AWS Management Console 'Instances' page. At the top, there's a navigation bar with 'EC2 > Instances'. Below this, a summary bar shows 'Instances (1/3)' with an 'Info' link, a 'Last updated 1 minute ago' timestamp, and buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. A search bar prompts 'Find Instance by attribute or tag (case-sensitive)' and a filter dropdown is set to 'All states'. The main table lists two EC2 instances, both of type 't3.micro' in the 'eu-north-1a' availability zone, with a 'Running' status and 'Initializing' status check. The first instance has ID 'i-00d9af84be3c03e2a' and public IP '13.62.48.38'. The second instance has ID 'i-0db3d45a0cd06e94c' and public IP '51.20.115.78'. Both instances are associated with the same name 'group6\_instan...'.

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ..
<input type="checkbox"/>	group6_instan...	i-00d9af84be3c03e2a	Running	t3.micro	Initializing	View alarms +	eu-north-1a	ec2-13-62-48-38.eu-no...	13.62.48.38
<input checked="" type="checkbox"/>	group6_instan...	i-0db3d45a0cd06e94c	Running	t3.micro	Initializing	View alarms +	eu-north-1a	ec2-51-20-115-78.eu-n...	51.20.115.78

- We launched 2 EC2 instances using the same AMI and ensured they were both in the same availability zone i.e. *eu-north-1a*

# 2. CREATING AN ELASTIC LOAD BALANCER



The screenshot displays the AWS Management Console interface. At the top, a green banner confirms the successful creation of the load balancer 'group6alb', noting that it may take a few minutes to fully set up and route traffic. Below this, a light blue banner informs users that Application Load Balancers now support public IPv4 IP Address Management (IPAM), with a link to 'Edit IP pools'. The main content area shows the details for 'group6alb'. The left sidebar contains navigation links for 'EC2', 'Load balancers', and 'group6alb'. The 'Load Balancing' section is expanded, showing 'Load Balancers', 'Target Groups', and 'Trust Stores'. The 'Details' section for 'group6alb' includes the following information:

Details	
<b>Load balancer type</b> Application	<b>Status</b> Provisioning
<b>Scheme</b> Internet-facing	<b>Hosted zone</b> Z23TAZ6LKFMNIO
<b>VPC</b> <a href="#">vpc-0898d62c6ae440b97</a>	<b>Load balancer IP address type</b> IPv4
<b>Availability Zones</b>	
<a href="#">subnet-05d5f440b1bac8902</a> eu-north-1a (eun1-az1)	
<a href="#">subnet-08d291a95266006ed</a> eu-north-1b (eun1-az2)	
<a href="#">subnet-0a76e5eca7f3a4a6a</a> eu-north-1c (eun1-az3)	
<b>Date created</b> June 3, 2025, 18:41 (UTC+00:00)	
<b>Load balancer ARN</b>	<b>DNS name</b> <a href="#">Info</a>

- We created an application load balancer and attached it to the VPC with three Availability zones.
- Configured it to accept HTTP requests using port 80.
- Selected the launched instances to register with the ELB.
- This was done to improve fault tolerance, high availability and latency.



# 3.CONFIGURING AN AUTO SCALING GROUP

✔ Auto Scaling group updated successfully

Auto Scaling groups (1) Info

Last updated less than a minute ago

Launch configurations

Launch templates

Actions

Create Auto Scaling group

Search your Auto Scaling groups

< 1 >

<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max
<input type="checkbox"/>	<a href="#">group6-asg</a>	<a href="#">group6_ltemplate</a>   Version Default	2	-	2	1	4

- Created an auto-scaling group named *group6-asg*
- This was achieved by first creating a launch template named *group6\_ltemplate* to serve as a template for creating instances.
- Desired capacity was set to 2, minimum=1 & maximum=4.
- The load balancer was then attached to the auto scaling group.

# 4.SETTING UP SCALING POLICIES

**Alarms (2)**

☐ Hide Auto Scaling alarms

Clear selection

Create composite alarm

Actions ▾

Create alarm

Alarm state: Any ▾

Alarm type: Any ▾

Actions status: Any ▾

< 1 > ⚙

**CPUabove70**☐

**Policy type**  
Step scaling

**Enabled or disabled**  
Enabled

**Execute policy when**  
**CPUaBOVE70**  
breaches the alarm threshold: CPUUtilization >= 70 for 1 consecutive periods of 60 seconds for the metric dimensions:  
AutoScalingGroupName = group6-asg

**Take the action**  
Add 1 capacity units when 70 <= CPUUtilization < +infinity

**Instances need**  
60 seconds to warm up after each step

**CPUBelow30**☐

**Policy type**  
Step scaling

**Enabled or disabled**  
Enabled

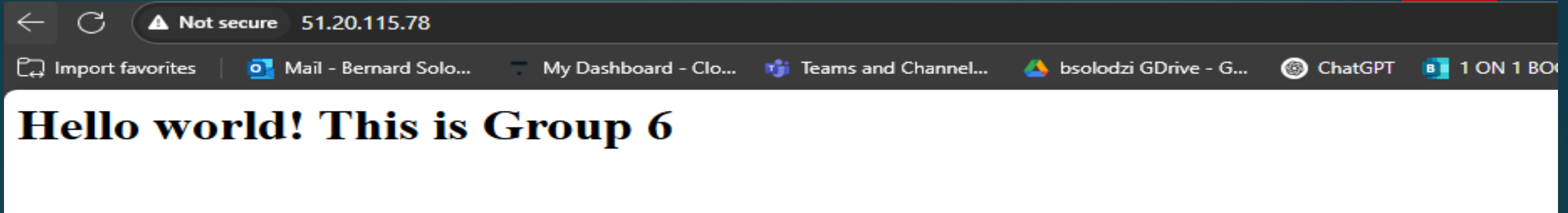
**Execute policy when**  
**CPUBelow30**  
breaches the alarm threshold: CPUUtilization <= 30 for 1 consecutive periods of 60 seconds for the metric dimensions:  
AutoScalingGroupName = group6-asg

**Take the action**  
Remove 1 capacity units when 30 >= CPUUtilization > -infinity

- We used CloudWatch to create alarms based on CPU usage and attached to scaling policies that will trigger instance creation or termination based on CPU utilization.
- CPUabove70 handled CPU usage above 70% which would trigger an addition of 1 instance to our resources.
- CPUbelow30 handled CPU usage below 30% which would trigger a termination of 1 instance from our resources.
- These policies were then linked to the auto-scaling group.



# 5. TESTING THE LOAD BALANCER



- We accessed the application load balancer's URL by copying its DNS Name: *group6alb-1737430742.eu-north-1.elb.amazonaws.com* and pasted in web tab.
- This displayed the content of the html file we saw earlier but this time with a different address and not the public IP address of the EC2 instance indicating an active load balancer.

# 6.SIMULATING AUTO-SCALING

```
Total download size: 34 k
Installed size: 68 k
Downloading Packages:
stress-1.0.7-2.amzn2023.0.1.x86_64.rpm          584 kB/s | 34 kB    00:00
-----
Total                                          408 kB/s | 34 kB    00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                                1/1
  Installing     : stress-1.0.7-2.amzn2023.0.1.x86_64 1/1
  Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64 1/1
  Verifying      : stress-1.0.7-2.amzn2023.0.1.x86_64 1/1

Installed:
  stress-1.0.7-2.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-10-0-0-188 ~]$ stress -c 1 -i 1 -m 1 --vm-bytes 128M -t 100s
stress: info: [29821] dispatching hogs: 1 cpu, 1 io, 1 vm, 0 hdd
stress: info: [29821] successful run completed in 100s
[ec2-user@ip-10-0-0-188 ~]$
```

i-0db3d45a0cd06e94c (group6\_instance2)

PublicIPs: 51.20.115.78 PrivateIPs: 10.0.0.188

- Firstly, we connected to our running instances via the AWS CLI.
- To simulate traffic, we installed a tool called stress using the '*sudo yum install -y stress*' command
- Then, we run a command - *stress -c 1 -i 1 -m 1 --vm-bytes 128M -t 100s* to simulate high load.
- **stress-** Runs the stress tool to load system resources
- **-c 1:** Uses 1 CPU core for stress testing
- **-i 1:** Starts 1 I/O stressor (light disk I/O load)
- **-m 1:** Starts 1 memory stressor (uses RAM)
- **--vm-bytes 128 :** Allocates 128 MB of memory per memory worker
- **-t 100s:** Runs the stress test for 100 seconds

# SIMULATING AUTO-SCALING cont'd

CPU Utilization: Average

5 minutes ▼

Average ▼

1h

3h

12h

1d

3d

1w

Custom 

UTC timezone ▼

 ▼

✕

Various units



Instances (5) [Info](#)

Last updated  
less than a minute ago



Connect

Instance state ▼















Actions ▼

Launch instances ▼

 Find Instance by attribute or tag (case-sensitive)

All states ▼

< 1 > 

<input type="checkbox"/>	Name  ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability Zone ▼	Public
<input type="checkbox"/>		<a href="#">i-0217e330ffdb86228</a>	Terminated  	t3.micro	–	<a href="#">View alarms +</a>	eu-north-1c	–
<input type="checkbox"/>	ec2tut	<a href="#">i-0e911191d9732408a</a>	Stopped  	t3.micro	–	<a href="#">View alarms +</a>	eu-north-1a	–
<input type="checkbox"/>	group6_instan...	<a href="#">i-00d9af84be3c03e2a</a>	Running  	t3.micro	 3/3 checks passec	<a href="#">View alarms +</a>	eu-north-1a	ec2-13-
<input type="checkbox"/>	group6_instan...	<a href="#">i-0db3d45a0cd06e94c</a>	Running  	t3.micro	 3/3 checks passec	<a href="#">View alarms +</a>	eu-north-1a	ec2-51-
<input type="checkbox"/>		<a href="#">i-09f906853a74a273c</a>	Running  	t3.micro	 3/3 checks passec	<a href="#">View alarms +</a>	eu-north-1b	ec2-51-

# SIMULATING AUTO-SCALING cont'd

CPU Utilization: Average

5 minutes

Average

1h

3h

12h

1d

3d

1w

Custom

UTC timezone

Refresh Close

Various units



Instances (1/6) Info

Last updated  
2 minutes ago



Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
<input type="checkbox"/>	<a href="#">ec2tut</a>	i-0217e330ffdb86228	Terminated	t3.micro	–	<a href="#">View alarms +</a>	eu-north-1c	–
<input type="checkbox"/>	ec2tut	i-0e911191d9732408a	Stopped	t3.micro	–	<a href="#">View alarms +</a>	eu-north-1a	–
<input type="checkbox"/>	group6_instan...	i-00d9af84be3c03e2a	Running	t3.micro	3/3 checks passec	<a href="#">View alarms +</a>	eu-north-1a	ec2-13-
<input type="checkbox"/>	group6_instan...	i-0db3d45a0cd06e94c	Running	t3.micro	3/3 checks passec	<a href="#">View alarms +</a>	eu-north-1a	ec2-51-
<input type="checkbox"/>		i-09f906853a74a273c	Terminated	t3.micro	–	<a href="#">View alarms +</a>	eu-north-1b	–
<input checked="" type="checkbox"/>		i-03259e0719af33505	Running	t3.micro	3/3 checks passec	<a href="#">View alarms +</a>	eu-north-1b	ec2-16-

# 7. Monitoring Auto scaling activity

## Activity history (7)

Filter activity history

<1>

Status	Description	Cause	Start time	End
Successful	Terminating EC2 instance: i-04f4117d733570069	At 2025-06-03T21:13:42Z a monitor alarm CPUbelow30 in state ALARM triggered policy CPUbelow30 changing the desired capacity from 2 to 1. At 2025-06-03T21:13:48Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-06-03T21:13:48Z instance i-04f4117d733570069 was selected for termination.	2025 June 03, 09:13:48 PM +00:00	2025 June 03, 09:13:48 PM +00:00
Successful	Launching a new EC2 instance: i-04f4117d733570069	At 2025-06-03T21:08:13Z a user request update of AutoScalingGroup constraints to min: 1, max: 4, desired: 2 changing the desired capacity from 1 to 2. At 2025-06-03T21:08:21Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2025 June 03, 09:08:23 PM +00:00	2025 June 03, 09:08:23 PM +00:00
Successful	Launching a new EC2 instance: i-03259e0719af33505	At 2025-06-03T20:11:11Z an instance was launched in response to an unhealthy instance needing to be replaced.	2025 June 03, 08:11:13 PM +00:00	2025 June 03, 08:11:13 PM +00:00
Successful	Terminating EC2 instance: i-09f906853a74a273c	At 2025-06-03T20:11:11Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2025 June 03, 08:11:11 PM +00:00	2025 June 03, 08:11:11 PM +00:00
Successful	Terminating EC2 instance: i-0217e330ffdb86228	At 2025-06-03T19:26:42Z a monitor alarm CPUbelow30 in state ALARM triggered policy CPUbelow30 changing the desired capacity from 2 to 1. At 2025-06-03T19:26:56Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-06-03T19:26:56Z instance i-0217e330ffdb86228 was selected for termination.	2025 June 03, 07:26:56 PM +00:00	2025 June 03, 07:26:56 PM +00:00

© 2025, Amazon Web Services, Inc. or its affiliates.

[Privacy](#)

[Terms](#)

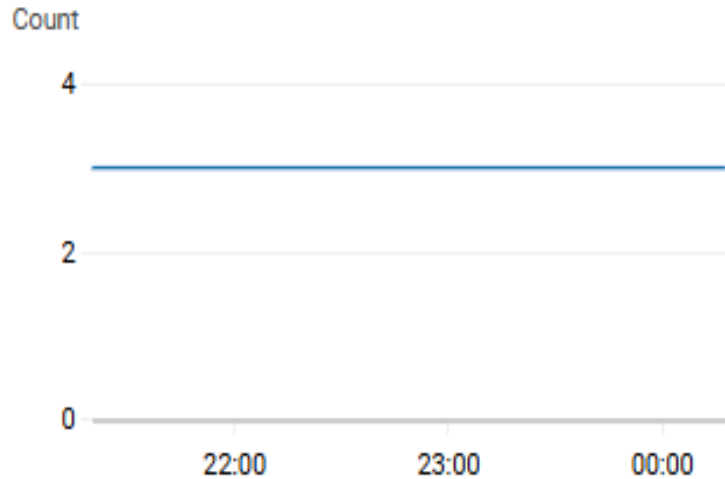
[Cookie preferences](#)

## Target group: group6target

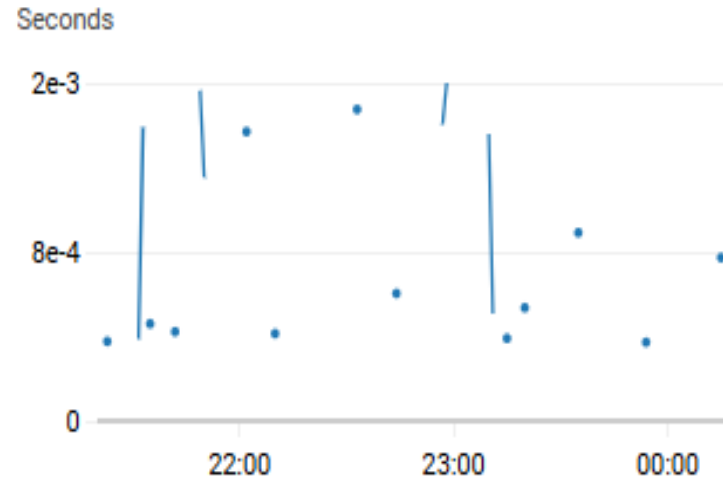
Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 80	HTTP1	<a href="#">vpc-0898d62c6ae440b97</a>
IP address type	Load balancer		
IPv4	<a href="#">group6alb</a>		
3 Total targets	3 Healthy 0 Anomalous	0 Unhealthy	0 Unused
		0 Initial	0 Draining

# Monitoring Auto scaling activity cont.

Healthy Hosts (Average)



Target Response Time



Requests



- Using CloudWatch, we checked the number of healthy instances running, auto scaling activity (scale up and down actions) and then the health status of ALB and EC2. Inspecting the activity history, we also noted that the ASG adjusts the number of instances based on traffic.



# CONCLUSION

- ▶ Load balancing and autoscaling help keep systems running smoothly by distributing traffic and adjusting resources as needed.
- ▶ Load balancers ensure no single server gets overwhelmed, while autoscaling automatically adds or removes computing power based on demand.
- ▶ Testing the system under heavy load showed how these technologies work in real time. Watching resources scale up when traffic increased and scale down when it dropped helped the group understand how autoscaling keeps things efficient while load balancing keeps everything stable.
- ▶ This hands-on experience made it clear why these strategies are essential for maintaining reliable, fault tolerant and responsive cloud systems.

# Group Members

Bernard Solodzi

Clement Owusu Bempah

Rachel Atia

Esther Acheampong

Benedicta Opoku-Amankwaah

Samuel Kofi Asare Dwumah

Marie-Pearl Otoo

Grace Wiredu

Muniratu Iddris

Riverson Atta

Dorothy Assan

Benedicta Maayuumle Djangmah



Thank you