# Assignment ELB

**Name:** vikram

---

## Problem Statement

You work for XYZ Corporation that uses on-premise solutions and a limited number of systems. With the increase in requests in their application, the load also increases. So, to handle the load, the corporation has to buy more systems almost on a regular basis. Realizing the need to cut down the expenses on systems, they decided to move their infrastructure to AWS.

## Tasks To Be Performed:

1. Manage the scaling requirements of the company by:

   a. Deploying multiple compute resources on the cloud as soon as the load increases and the CPU utilization exceeds 80%

   b. Removing the resources when the CPU utilization goes under 60%

2. Create a load balancer to distribute the load between compute resources.
3. Route the traffic to the company's domain.

---

## Step-by-Step Setup

## STEP 1: Create a Launch Template

Go to EC2 → Launch Templates → Create launch template

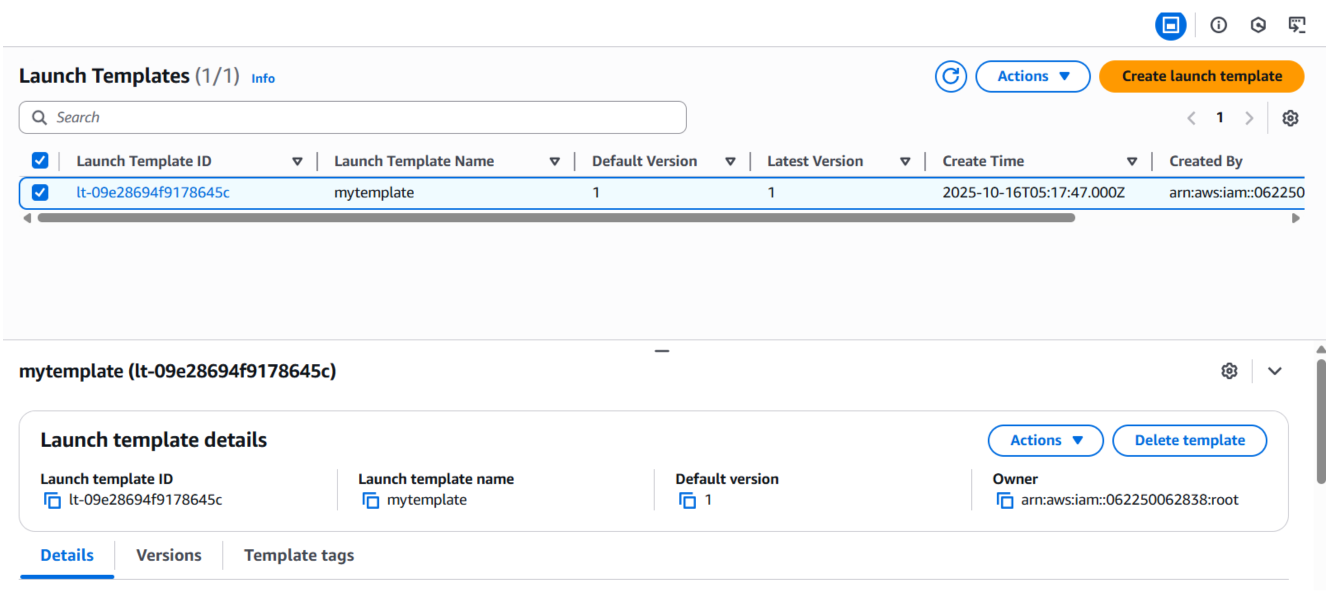
- Name: mytemplate
- AMI: Amazon Linux 2 (x86_64)
- Instance type: t3
- Key pair: vik-87
- Security group: create a new one:
  - Allow HTTP (80) from 0.0.0.0/0
  - Allow SSH (22) from  0.0.0.0/0

- User data: paste this script:

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl enable httpd
systemctl start httpd
echo "<h1>Welcome to my App - $(hostname)</h1>" > /var/www/html/index.html
```

Click Create launch template

---

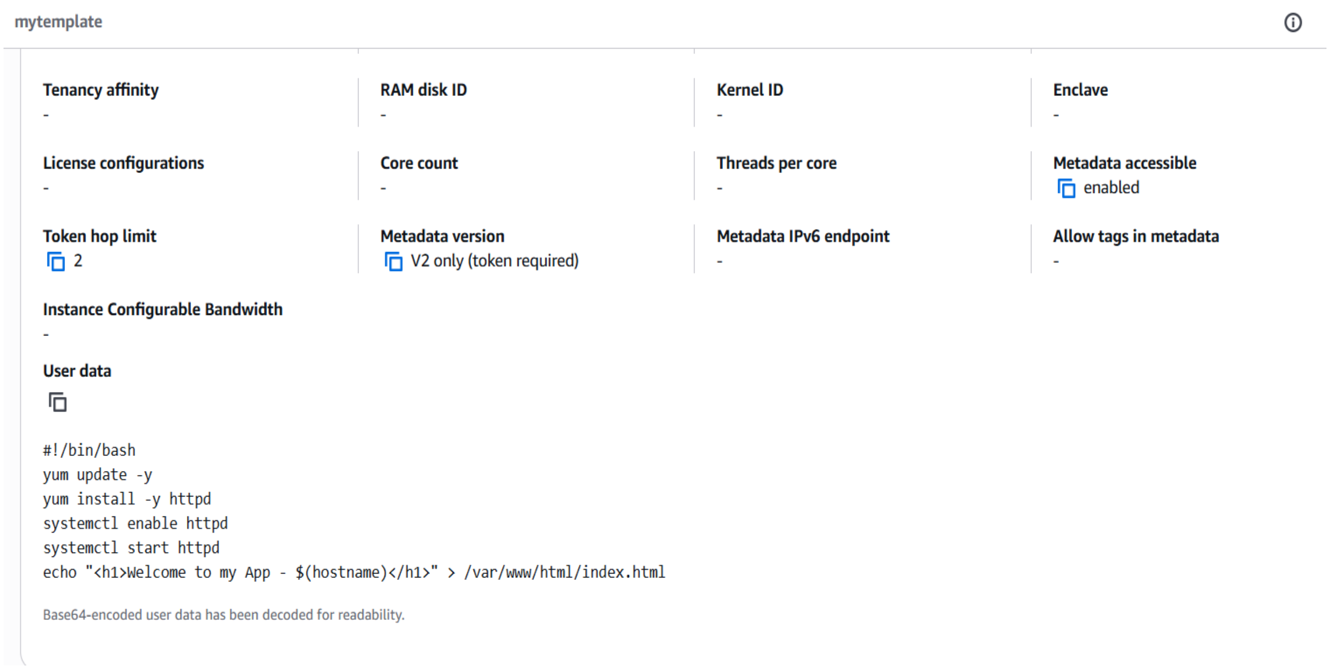


| Launch Template ID | Launch Template Name | Default Version | Latest Version | Create Time | Created By |
|---|---|---|---|---|---|
| lt-09e28694f9178645c | mytemplate | 1 | 1 | 2025-10-16T05:17:47.000Z | arn:aws:iam::062250 |

mytemplate (lt-09e28694f9178645c)

Launch template details

Launch template ID: lt-09e28694f9178645c
Launch template name: mytemplate
Default version: 1
Owner: arn:aws:iam::062250062838:root



| | | | |
|---|---|---|---|
| Tenancy affinity<br>- | RAM disk ID<br>- | Kernel ID<br>- | Enclave<br>- |
| License configurations<br>- | Core count<br>- | Threads per core<br>- | Metadata accessible<br>enabled |
| Token hop limit<br>2 | Metadata version<br>V2 only (token required) | Metadata IPv6 endpoint<br>- | Allow tags in metadata<br>- |
| Instance Configurable Bandwidth<br>- | | | |

User data

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl enable httpd
systemctl start httpd
echo "<h1>Welcome to my App - $(hostname)</h1>" > /var/www/html/index.html
```

Base64-encoded user data has been decoded for readability.

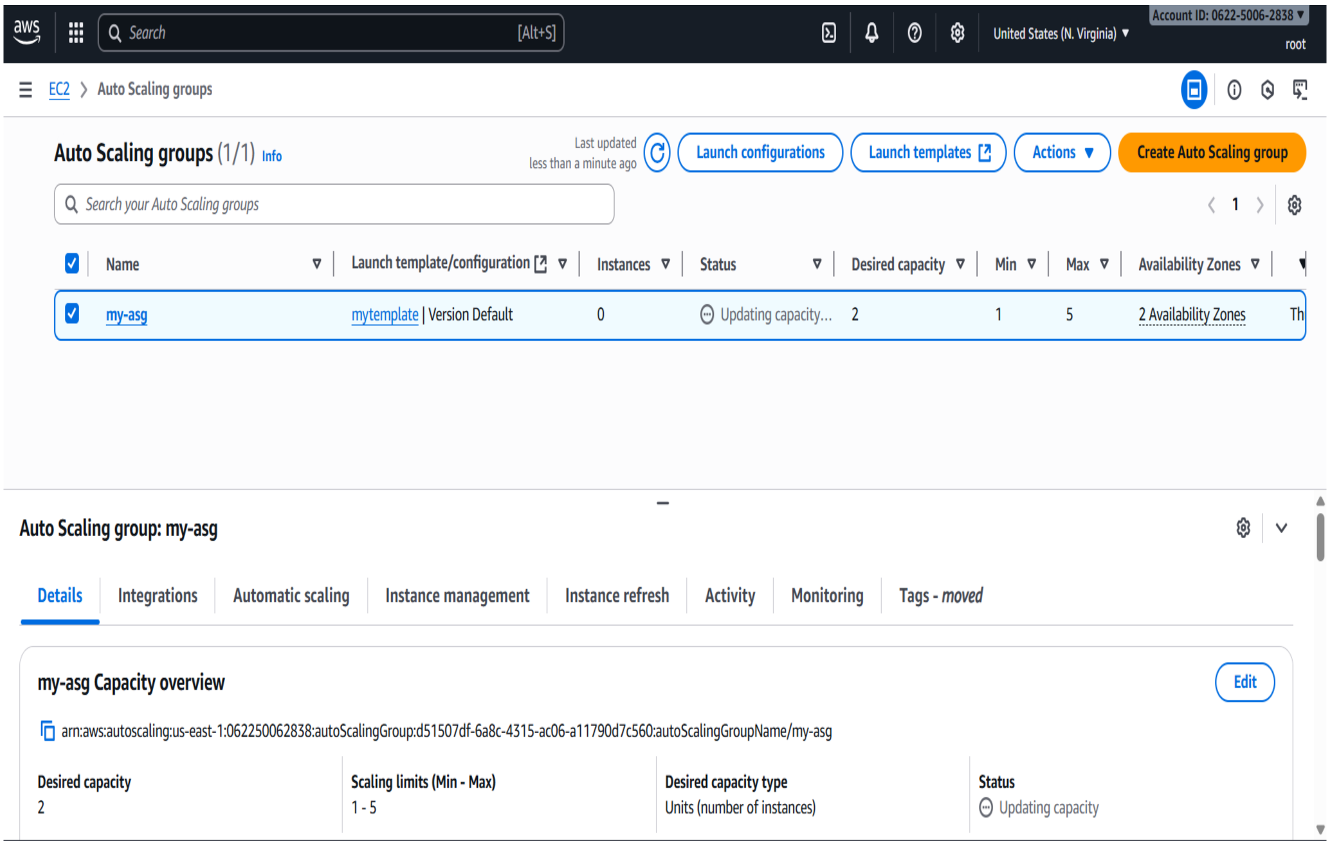
## STEP 2: Create an Auto Scaling Group

Go to EC2 → Auto Scaling Groups → Create Auto Scaling Group

- Choose launch template: mytemplate
- Name: my asg
- Choose your VPC and Subnets: (select at least 2 in different AZs)
- Group size:
  - Desired capacity → 2
  - Minimum capacity → 1
  - Maximum capacity → 5
- Advanced options: leave defaults

Click Next → Next → Create Auto Scaling group

---



Search [Alt+S]
United States (N. Virginia)
Account ID: 0622-5006-2838
root
EC2 > Auto Scaling groups
Last updated less than a minute ago
Launch configurations
Launch templates
Actions
Create Auto Scaling group
Search your Auto Scaling groups
Details
Integrations
Automatic scaling
Instance management
Instance refresh
Activity
Monitoring
Tags - moved
Edit

Auto Scaling groups (1/1) Info

| Name | Launch template/configuration | Instances | Status | Desired capacity | Min | Max | Availability Zones |
|---|---|---|---|---|---|---|---|
| my-asg | mytemplate \| Version Default | 0 | Updating capacity... | 2 | 1 | 5 | 2 Availability Zones |

Auto Scaling group: my-asg

my-asg Capacity overview

arn:aws:autoscaling:us-east-1:062250062838:autoScalingGroup:d51507df-6a8c-4315-ac06-a11790d7c560:autoScalingGroupName/my-asg

| Desired capacity | Scaling limits (Min - Max) | Desired capacity type | Status |
|---|---|---|---|
| 2 | 1 - 5 | Units (number of instances) | Updating capacity |

## STEP 3: Create an Application Load Balancer (ALB)

Go to EC2 → Load Balancers → Create Load Balancer

- Choose: Application Load Balancer
- Name: my alb
- Scheme: Internet-facing
- IP type: IPv4
- Choose the same VPC and subnets: (2 AZs)
- Security group: create one:
    - Allow HTTP (80) from 0.0.0.0/0
- Listener: HTTP:80

Create a new target group:

- Target type: Instances
- Name: my-targets
- Health check path: /

Click Create Load Balancer

After creation, go to the Target Group → Targets tab and click Edit → Add instances

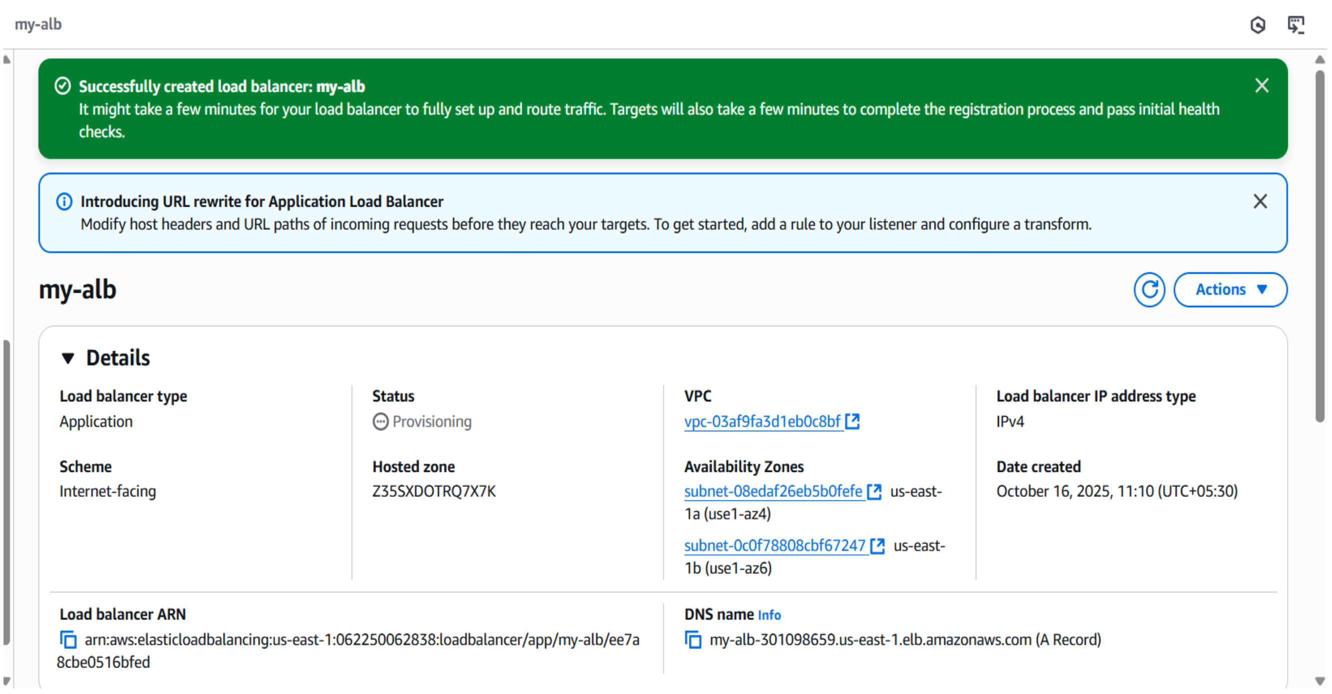
Select the instances from your Auto Scaling Group → Include as pending → Save

☑ Now attach the load balancer to your Auto Scaling Group

Go to EC2 → Auto Scaling Groups → my asg → Edit → Load balancing → Attach to an existing load balancer target group and choose my-targets.

Save the configuration.

---



**Successfully created load balancer: my-alb**
It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

**Introducing URL rewrite for Application Load Balancer**
Modify host headers and URL paths of incoming requests before they reach your targets. To get started, add a rule to your listener and configure a transform.

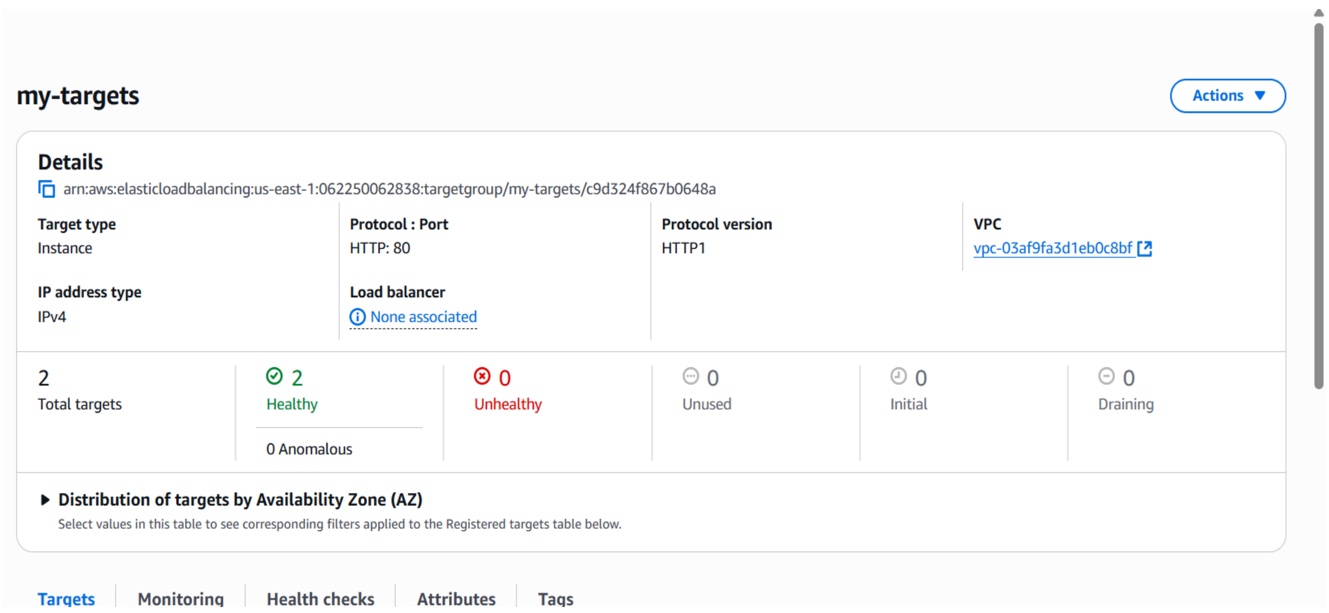
**my-alb**

**Details**

| | | | |
|---|---|---|---|
| **Load balancer type**<br>Application | **Status**<br>Provisioning | **VPC**<br>vpc-03af9fa3d1eb0c8bf | **Load balancer IP address type**<br>IPv4 |
| **Scheme**<br>Internet-facing | **Hosted zone**<br>Z35SXDOTRQ7X7K | **Availability Zones**<br>subnet-08edaf26eb5b0fefe us-east-1a (use1-az4)<br>subnet-0c0f78808cbf67247 us-east-1b (use1-az6) | **Date created**<br>October 16, 2025, 11:10 (UTC+05:30) |

**Load balancer ARN**
arn:aws:elasticloadbalancing:us-east-1:062250062838:loadbalancer/app/my-alb/ee7a8cbe0516bfed

**DNS name** Info
my-alb-301098659.us-east-1.elb.amazonaws.com (A Record)


my-targets

Actions ▼

**Details**

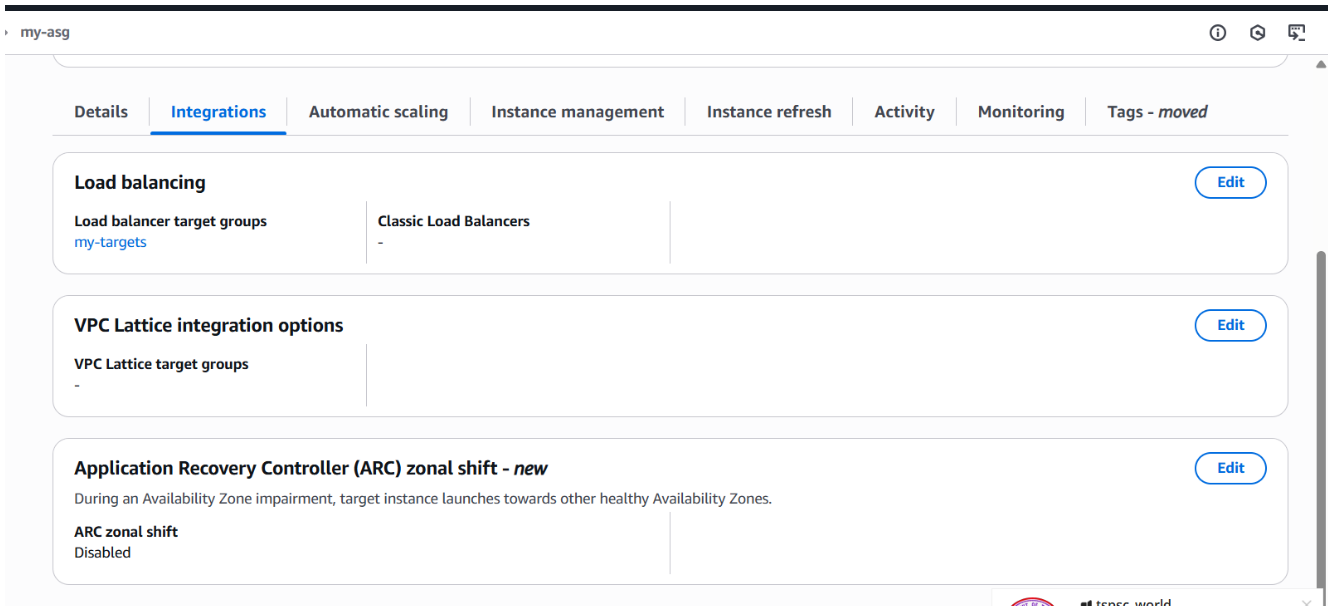
arn:aws:elasticloadbalancing:us-east-1:062250062838:targetgroup/my-targets/c9d324f867b0648a

| Target type | Protocol : Port | Protocol version | VPC |
|---|---|---|---|
| Instance | HTTP: 80 | HTTP1 | vpc-03af9fa3d1eb0c8bf |
| **IP address type** | **Load balancer** | | |
| IPv4 | None associated | | |

| 2 | ✓ 2 | ⊗ 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|
| Total targets | Healthy | Unhealthy | Unused | Initial | Draining |
| | 0 Anomalous | | | | |

▶ Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.



**Load balancing** — Edit

| Load balancer target groups | Classic Load Balancers |
|---|---|
| my-targets | - |

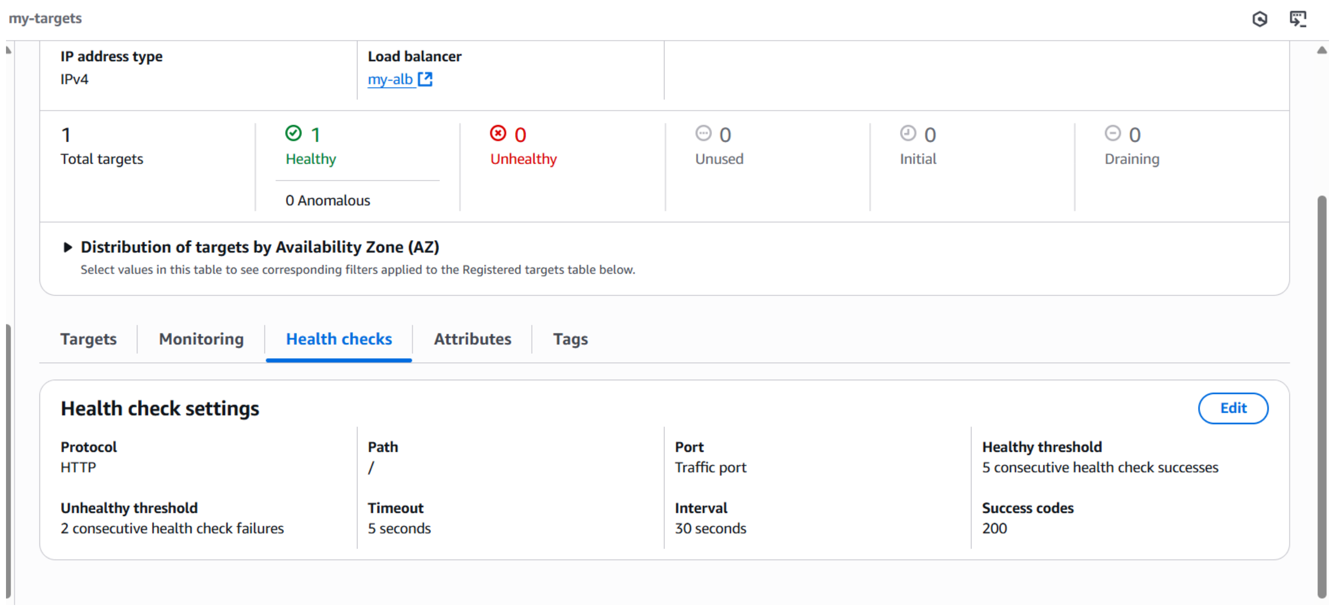
**VPC Lattice integration options** — Edit

VPC Lattice target groups: -

**Application Recovery Controller (ARC) zonal shift - *new*** — Edit

During an Availability Zone impairment, target instance launches towards other healthy Availability Zones.

ARC zonal shift: Disabled



| IP address type | Load balancer |
|---|---|
| IPv4 | my-alb |

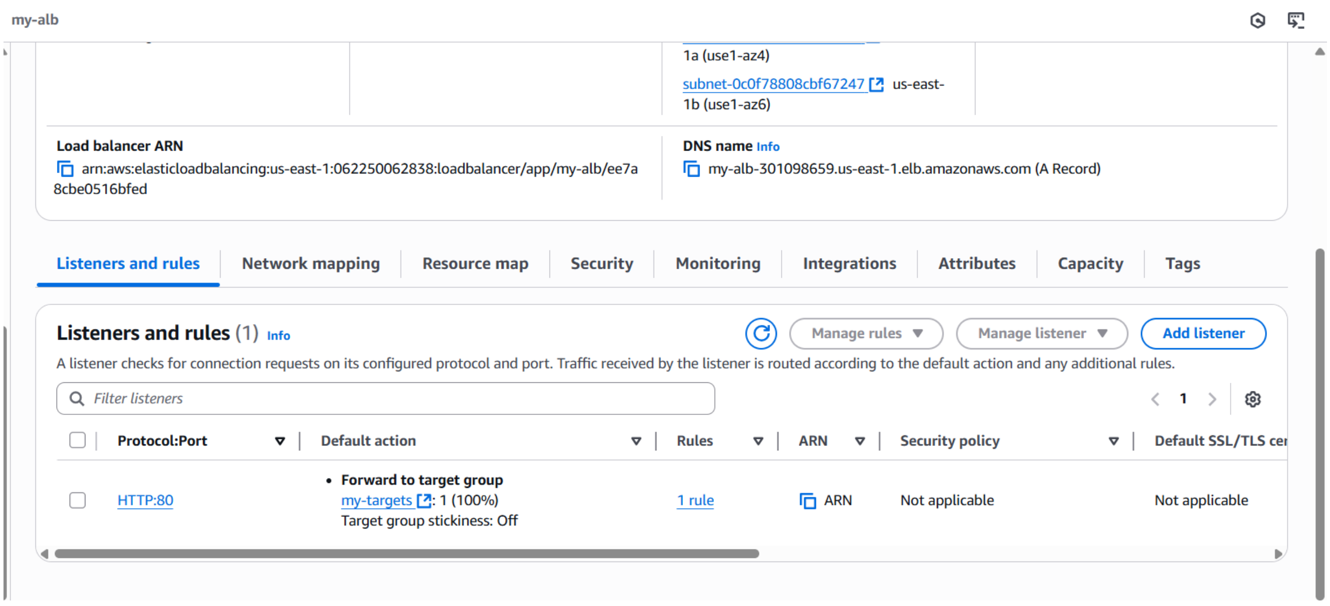
| 1 | ✓ 1 | ⊗ 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|
| Total targets | Healthy | Unhealthy | Unused | Initial | Draining |
| | 0 Anomalous | | | | |

▶ Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

**Health check settings** — Edit

| Protocol | Path | Port | Healthy threshold |
|---|---|---|---|
| HTTP | / | Traffic port | 5 consecutive health check successes |
| **Unhealthy threshold** | **Timeout** | **Interval** | **Success codes** |
| 2 consecutive health check failures | 5 seconds | 30 seconds | 200 |


my-alb

1a (use1-az4)
subnet-0c0f78808cbf67247 us-east-1b (use1-az6)

Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:062250062838:loadbalancer/app/my-alb/ee7a8cbe0516bfed

DNS name Info
my-alb-301098659.us-east-1.elb.amazonaws.com (A Record)

Listeners and rules (1) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

| Protocol:Port | Default action | Rules | ARN | Security policy | Default SSL/TLS cer |
|---|---|---|---|---|---|
| HTTP:80 | • Forward to target group my-targets: 1 (100%) Target group stickiness: Off | 1 rule | ARN | Not applicable | Not applicable |

## STEP 4: Configure Dynamic Scaling Policies

## PART A — Create the Scale Out Policy (CPU > 80%)

Step 1 — Open Your Auto Scaling Group

- Go to EC2 Console → Auto Scaling Groups
- Click on your my asg name
- Go to the Automatic scaling tab
- Under Dynamic scaling policies, click Create dynamic scaling policy

Step 2 — Choose Policy Type

- Policy type: Simple scaling
- Policy name: ScaleOut-80

Step 3 — Create the Alarm (CloudWatch)

We'll now connect this policy to a CloudWatch Alarm that monitors CPU.

- Under CloudWatch alarms, click Create a new alarm
- A new tab opens for CloudWatch → Create Alarm

Step 4 — Select Metric

- Click Select metric
- Navigate like this:

  EC2 → By Auto Scaling Group → my asg → CPUUtilization
- Check the box next to CPUUtilization
- Click Select metric

Step 5 — Configure Metric

- Statistic: Average
- Period: 5 minutes
- Click Next

Step 6 — Define the Alarm Condition

- Threshold type: Static
- Whenever CPUUtilization is… → Greater than
- Threshold value: 80
- Datapoints to alarm: 2 of 2 (two 5-min periods = 10 mins sustained high CPU)
- Click Next

Step 7 — Notification (Optional)

- You can skip SNS notification → click Next

Step 8 — Name the Alarm

- Alarm name: ASG-CPUHigh-80
- Click Create alarm

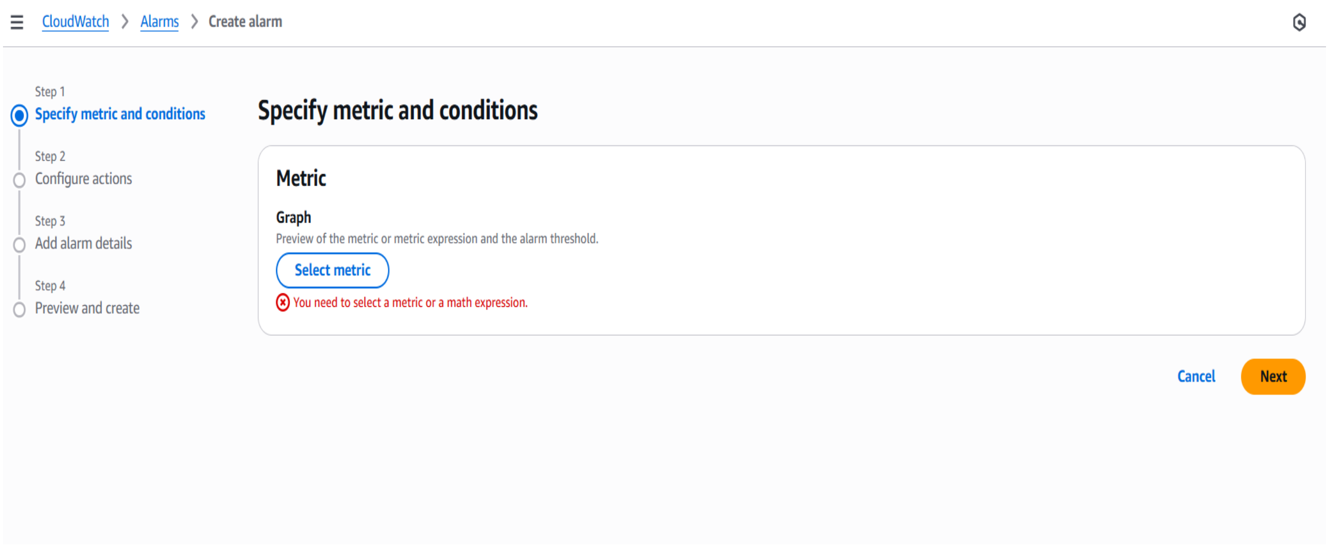
You'll automatically return to the ASG policy creation screen.

Step 9 — Define Scaling Action

- Take the action: Add
- Number of instances: 1
- Cooldown period: 300 seconds (default)
- Click Create ✅

Now your ScaleOut-80 policy is ready!

---


CloudWatch > Alarms > Create alarm

Step 1
Specify metric and conditions

Step 2
Configure actions

Step 3
Add alarm details

Step 4
Preview and create

Specify metric and conditions

Metric

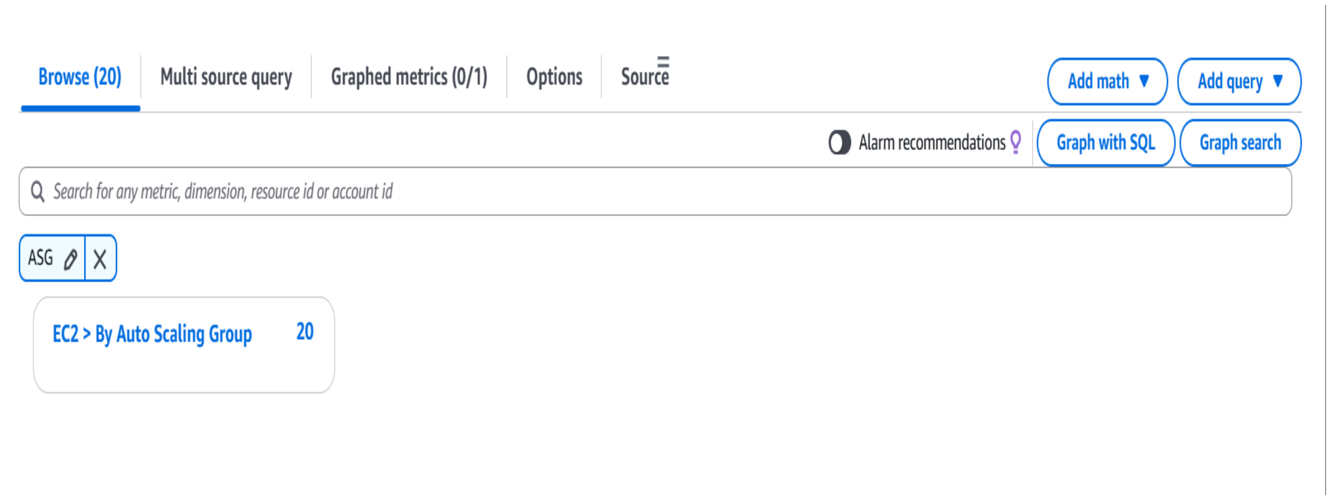
Graph

Preview of the metric or metric expression and the alarm threshold.

Select metric

You need to select a metric or a math expression.

Cancel Next


Browse (20) | Multi source query | Graphed metrics (0/1) | Options | Source

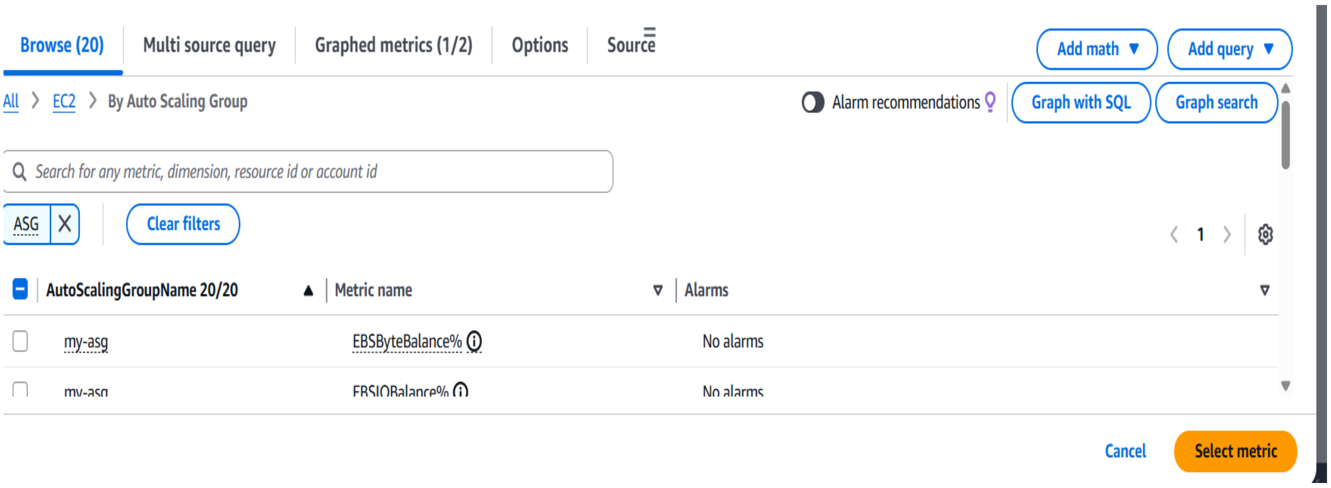
Add math ▼ | Add query ▼

Alarm recommendations | Graph with SQL | Graph search

Search for any metric, dimension, resource id or account id

ASG

EC2 > By Auto Scaling Group 20


Browse (20) | Multi source query | Graphed metrics (1/2) | Options | Source

Add math ▼ | Add query ▼

All > EC2 > By Auto Scaling Group

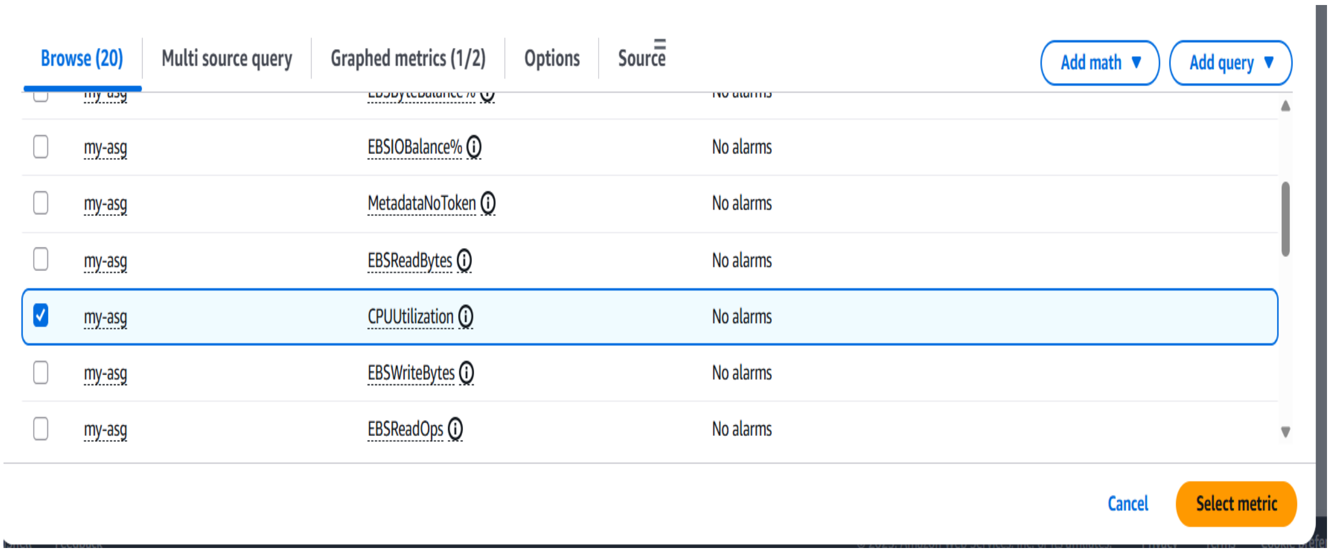
Alarm recommendations | Graph with SQL | Graph search

Search for any metric, dimension, resource id or account id

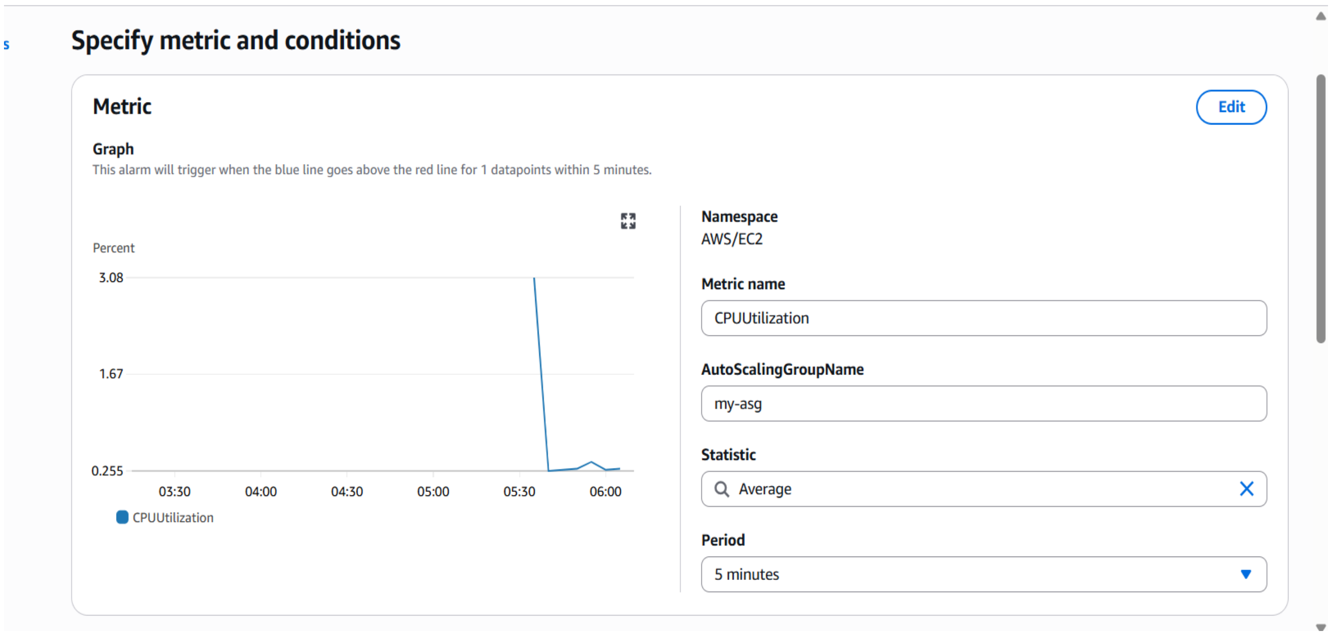
ASG | Clear filters

| | AutoScalingGroupName 20/20 | Metric name | Alarms |
|---|---|---|---|
| ☐ | my-asg | EBSByteBalance% | No alarms |
| ☐ | my-asg | EBSIOBalance% | No alarms |

Cancel | Select metric



| | | | |
|---|---|---|---|
| ☐ | my-asg | EBSIOBalance% | No alarms |
| ☐ | my-asg | MetadataNoToken | No alarms |
| ☐ | my-asg | EBSReadBytes | No alarms |
| ☑ | my-asg | CPUUtilization | No alarms |
| ☐ | my-asg | EBSWriteBytes | No alarms |
| ☐ | my-asg | EBSReadOps | No alarms |



## Specify metric and conditions

**Metric**

**Graph**

This alarm will trigger when the blue line goes above the red line for 1 datapoints within 5 minutes.

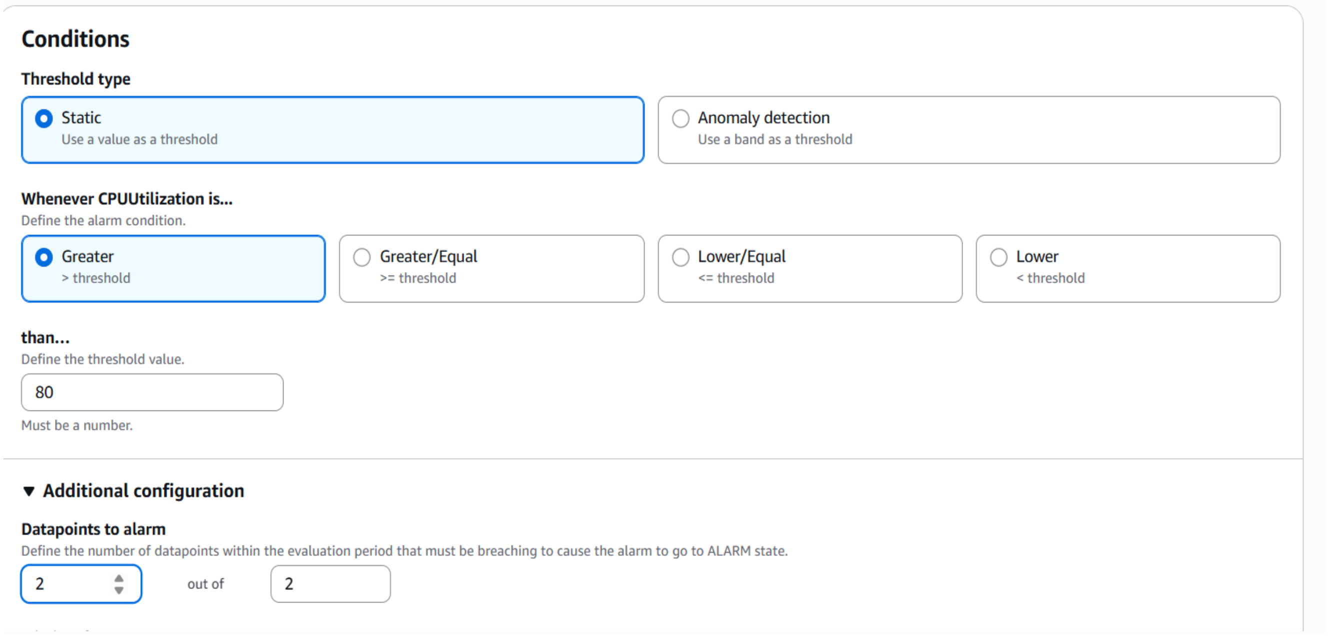
**Namespace**
AWS/EC2

**Metric name**
CPUUtilization

**AutoScalingGroupName**
my-asg

**Statistic**
Average

**Period**
5 minutes



## Conditions

**Threshold type**

[✓ Static] Use a value as a threshold
Anomaly detection Use a band as a threshold

**Whenever CPUUtilization is...**
Define the alarm condition.

[✓ Greater] > threshold
Greater/Equal >= threshold
Lower/Equal <= threshold
Lower < threshold

**than...**
Define the threshold value.

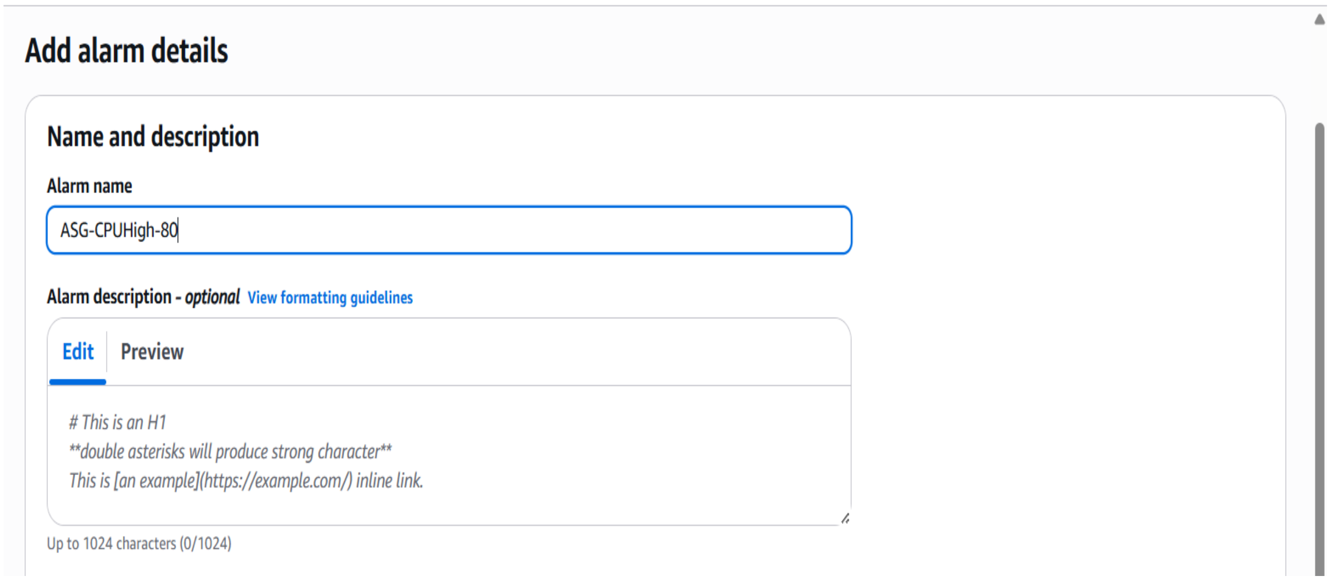
80

Must be a number.

**▼ Additional configuration**

**Datapoints to alarm**
Define the number of datapoints within the evaluation period that must be breaching to cause the alarm to go to ALARM state.

2 out of 2



Add alarm details

Name and description

Alarm name

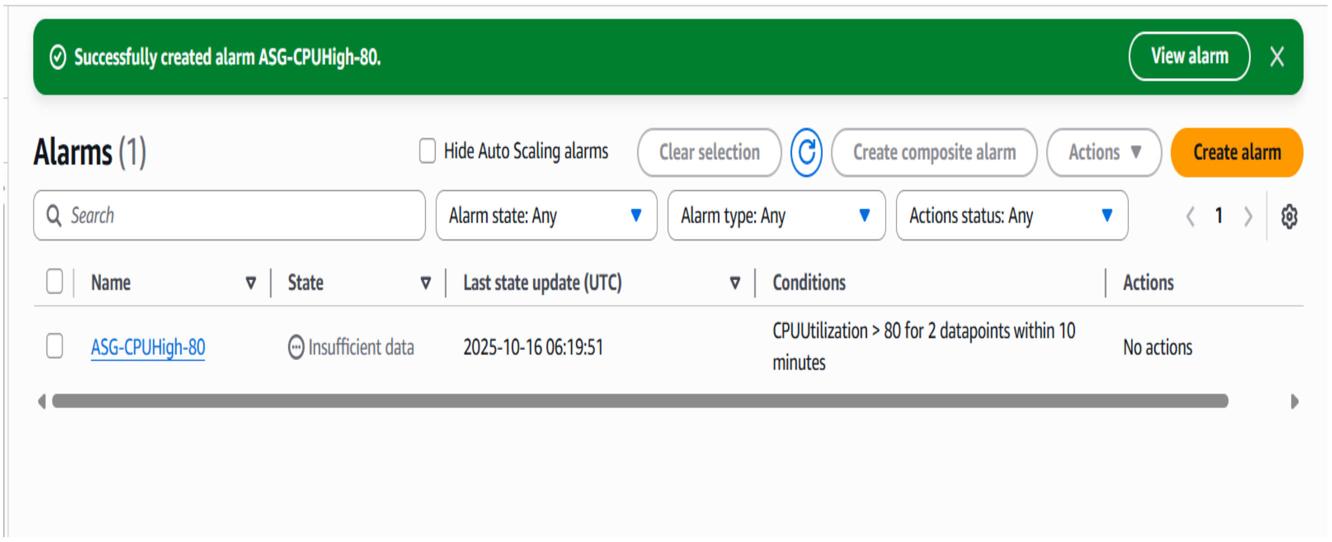
ASG-CPUHigh-80

Alarm description - *optional* View formatting guidelines

Edit | Preview

*# This is an H1*
***double asterisks will produce strong character***
*This is [an example](https://example.com/) inline link.*

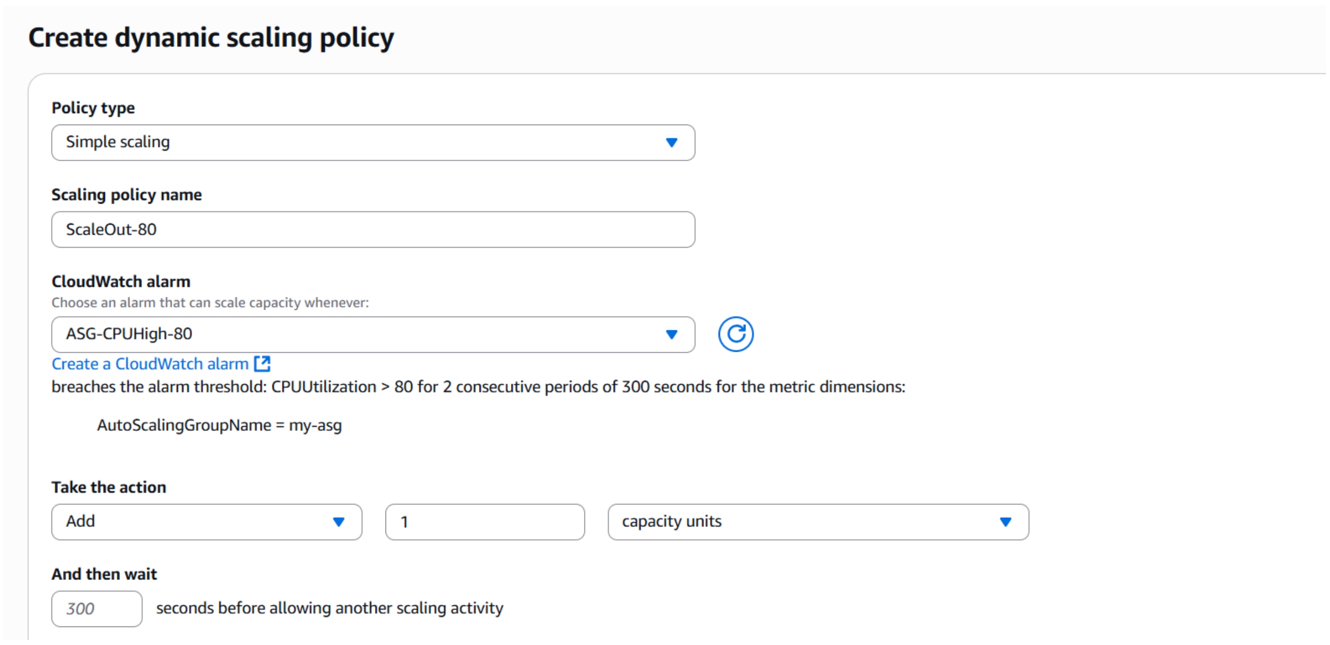
Up to 1024 characters (0/1024)



Successfully created alarm ASG-CPUHigh-80.

Alarms (1)

| Name | State | Last state update (UTC) | Conditions | Actions |
|---|---|---|---|---|
| ASG-CPUHigh-80 | Insufficient data | 2025-10-16 06:19:51 | CPUUtilization > 80 for 2 datapoints within 10 minutes | No actions |



Create dynamic scaling policy

Policy type

Simple scaling

Scaling policy name

ScaleOut-80

CloudWatch alarm

Choose an alarm that can scale capacity whenever:

ASG-CPUHigh-80

Create a CloudWatch alarm

breaches the alarm threshold: CPUUtilization > 80 for 2 consecutive periods of 300 seconds for the metric dimensions:

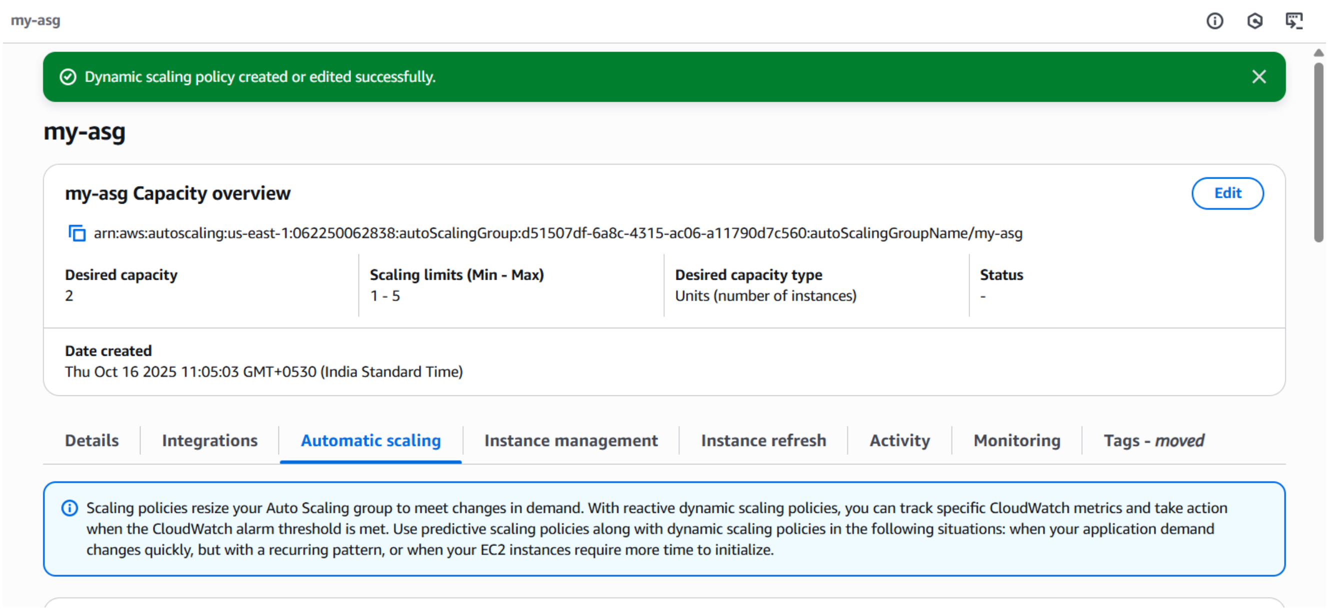
AutoScalingGroupName = my-asg

Take the action

Add | 1 | capacity units

And then wait

300 seconds before allowing another scaling activity


my-asg

Dynamic scaling policy created or edited successfully.

my-asg

**my-asg Capacity overview**

arn:aws:autoscaling:us-east-1:062250062838:autoScalingGroup:d51507df-6a8c-4315-ac06-a11790d7c560:autoScalingGroupName/my-asg

| Desired capacity | Scaling limits (Min - Max) | Desired capacity type | Status |
|---|---|---|---|
| 2 | 1 - 5 | Units (number of instances) | - |

Date created
Thu Oct 16 2025 11:05:03 GMT+0530 (India Standard Time)

Scaling policies resize your Auto Scaling group to meet changes in demand. With reactive dynamic scaling policies, you can track specific CloudWatch metrics and take action when the CloudWatch alarm threshold is met. Use predictive scaling policies along with dynamic scaling policies in the following situations: when your application demand changes quickly, but with a recurring pattern, or when your EC2 instances require more time to initialize.

## PART B — Create the Scale In Policy (CPU < 60%)

Step 1 — Add Another Policy

- In the same Automatic scaling tab, click Add policy again.
- Policy type: Simple scaling
- Policy name: ScaleIn-60

Step 2 — Create Another Alarm

- Click Create a new alarm
- You'll go to CloudWatch again.

Step 3 — Select Metric

- Same path:

  EC2 → By Auto Scaling Group → my asg → CPUUtilization
- Select it → Click Select metric

Step 4 — Define Condition

- Threshold type: Static
- Whenever CPUUtilization is… → Less than
- Threshold value: 60
- Datapoints to alarm: 2 of 2
- Period: 5 minutes
- Statistic: Average
- Click Next → Skip notification → Next

Step 5 — Name the Alarm

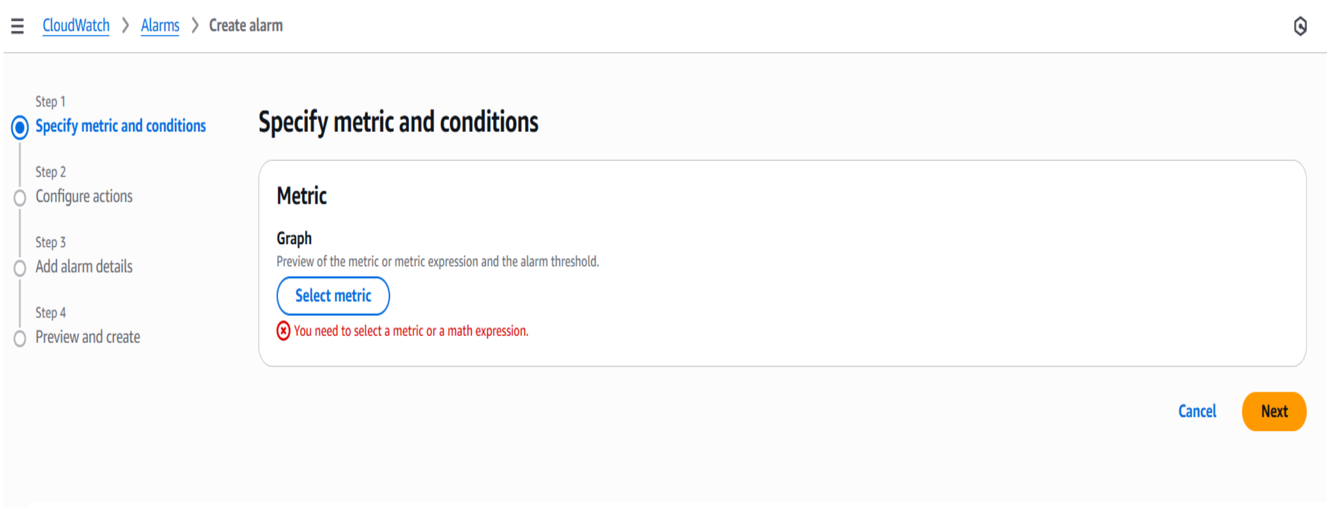
- Alarm name: ASG-CPULow-60
- Click Create alarm

Step 6 — Define Scaling Action

- Back on the ASG screen:
  - Action: Remove
  - Number of instances: 1
  - Cooldown period: 300 seconds
  - Click Create

☑ Done — both scaling policies are now active.

---



CloudWatch > Alarms > Create alarm

Step 1
Specify metric and conditions

Step 2
Configure actions

Step 3
Add alarm details

Step 4
Preview and create

## Specify metric and conditions

**Metric**

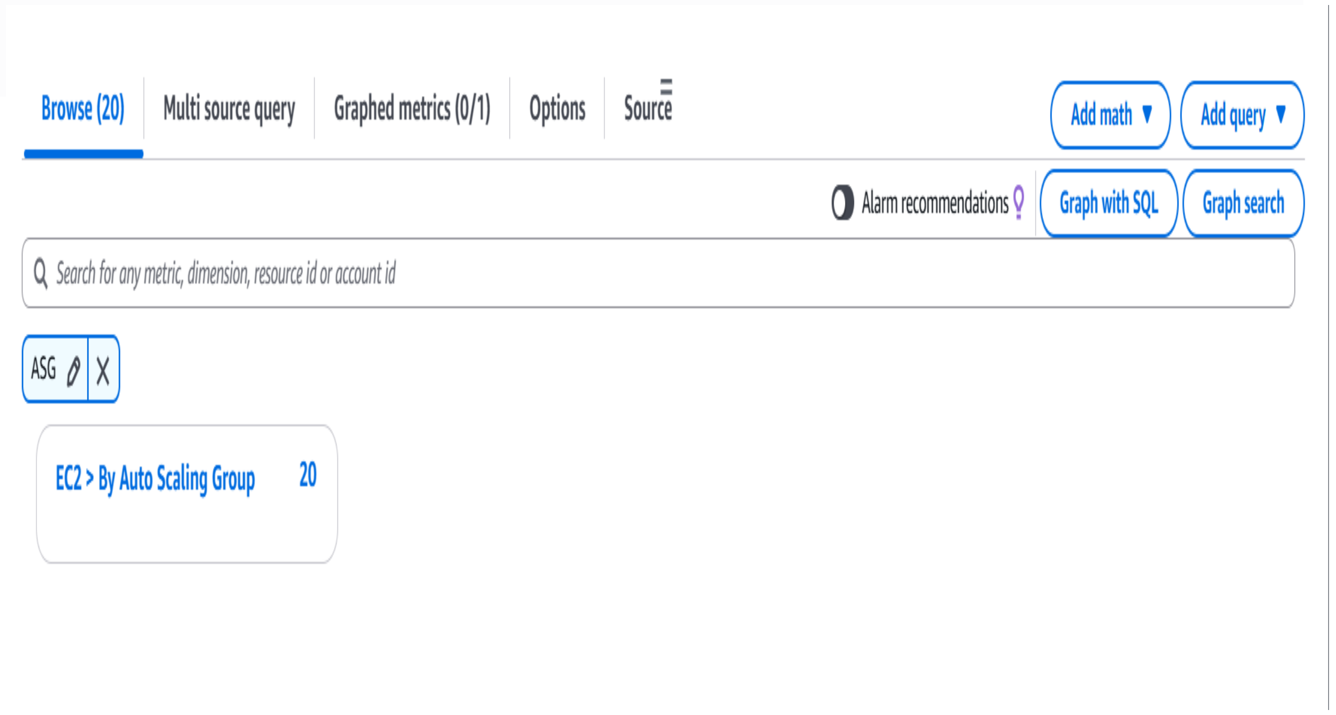
**Graph**

Preview of the metric or metric expression and the alarm threshold.

Select metric

You need to select a metric or a math expression.

Cancel   Next



Browse (20) | Multi source query | Graphed metrics (0/1) | Options | Source

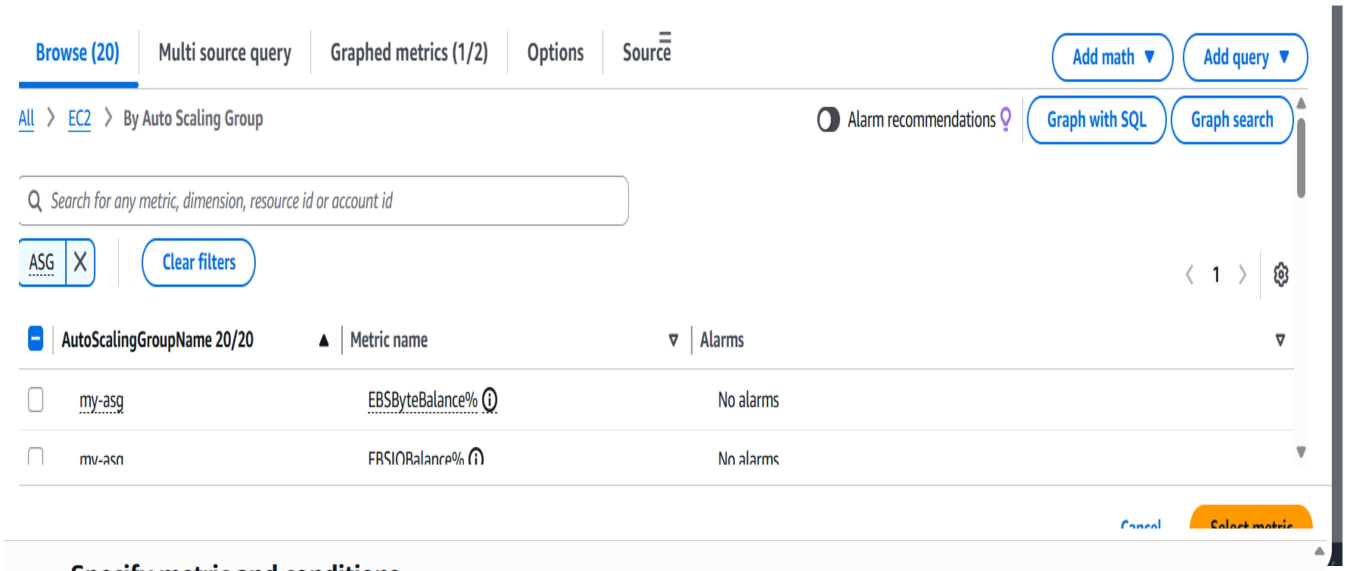
Add math ▼   Add query ▼

Alarm recommendations   Graph with SQL   Graph search

Search for any metric, dimension, resource id or account id

ASG

EC2 > By Auto Scaling Group   20



Browse (20) | Multi source query | Graphed metrics (1/2) | Options | Source

Add math ▼ | Add query ▼

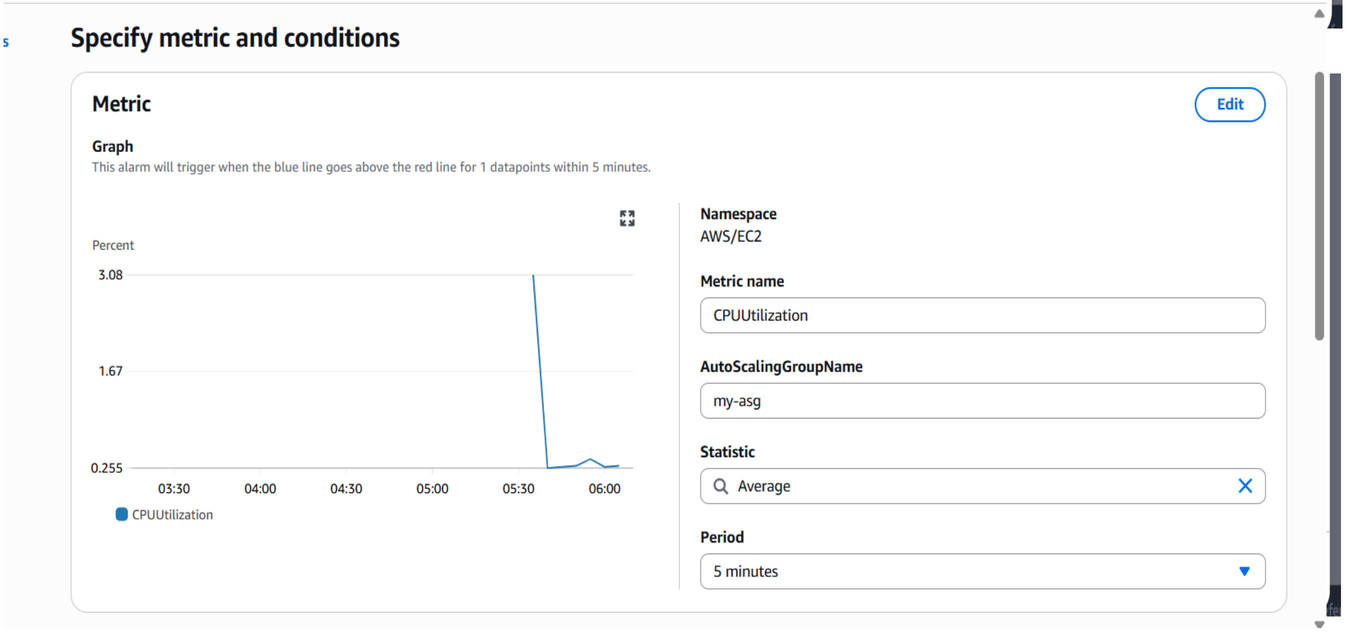
All > EC2 > By Auto Scaling Group

Alarm recommendations | Graph with SQL | Graph search

Search for any metric, dimension, resource id or account id

ASG X | Clear filters

| | AutoScalingGroupName 20/20 | Metric name | Alarms |
|---|---|---|---|
| ☐ | my-asg | EBSByteBalance% | No alarms |
| ☐ | my-asg | EBSIOBalance% | No alarms |



## Specify metric and conditions

### Metric

Edit

**Graph**

This alarm will trigger when the blue line goes above the red line for 1 datapoints within 5 minutes.

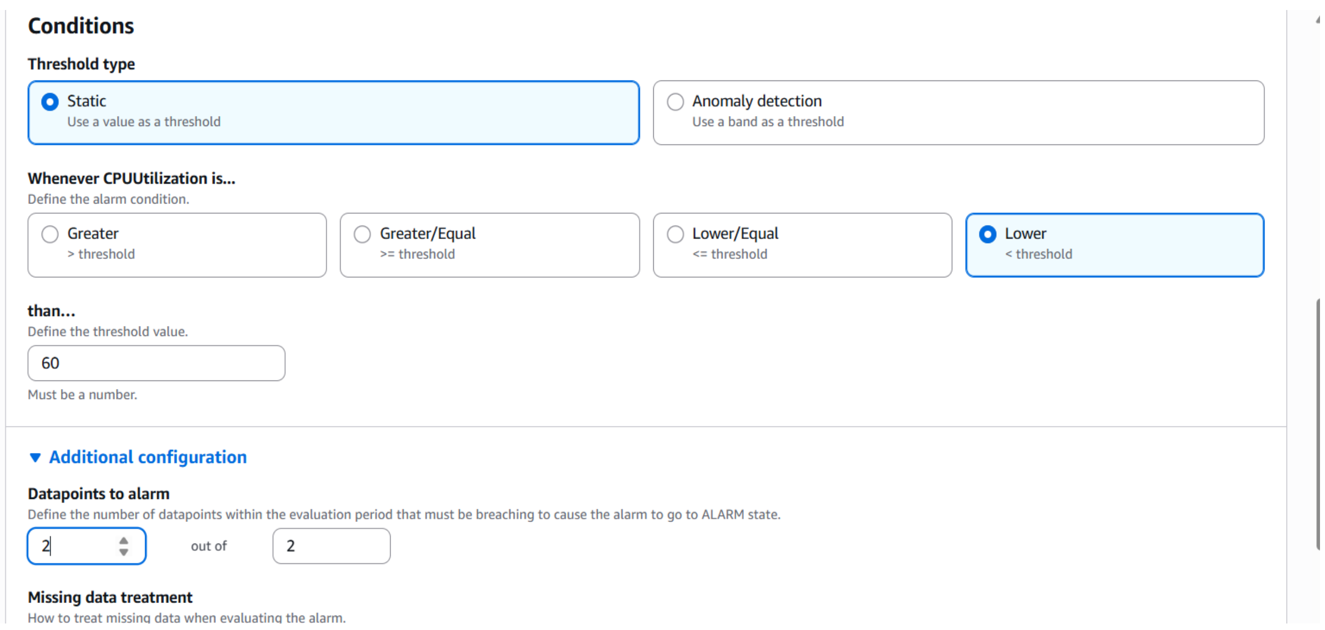
**Namespace**
AWS/EC2

**Metric name**
CPUUtilization

**AutoScalingGroupName**
my-asg

**Statistic**
Average

**Period**
5 minutes



### Conditions

**Threshold type**

[✓ Static — Use a value as a threshold] Anomaly detection — Use a band as a threshold

**Whenever CPUUtilization is...**

Define the alarm condition.

Greater — > threshold | Greater/Equal — >= threshold | Lower/Equal — <= threshold | [✓ Lower — < threshold]

**than...**

Define the threshold value.

60

Must be a number.

▼ Additional configuration

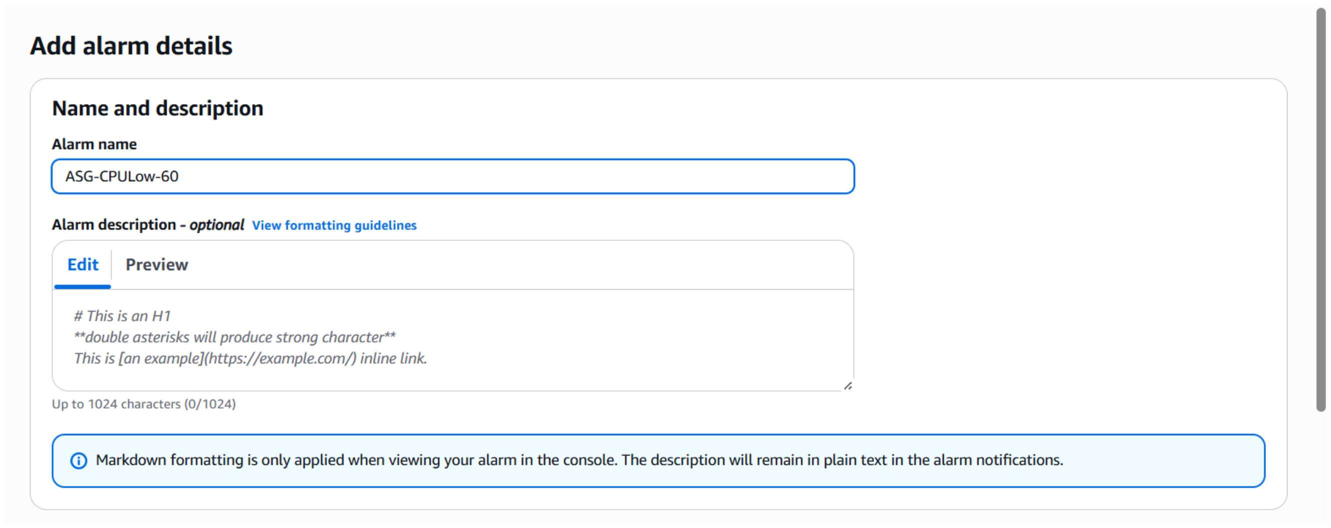
**Datapoints to alarm**

Define the number of datapoints within the evaluation period that must be breaching to cause the alarm to go to ALARM state.

2 out of 2

**Missing data treatment**

How to treat missing data when evaluating the alarm.



**Add alarm details**

**Name and description**

**Alarm name**

ASG-CPULow-60

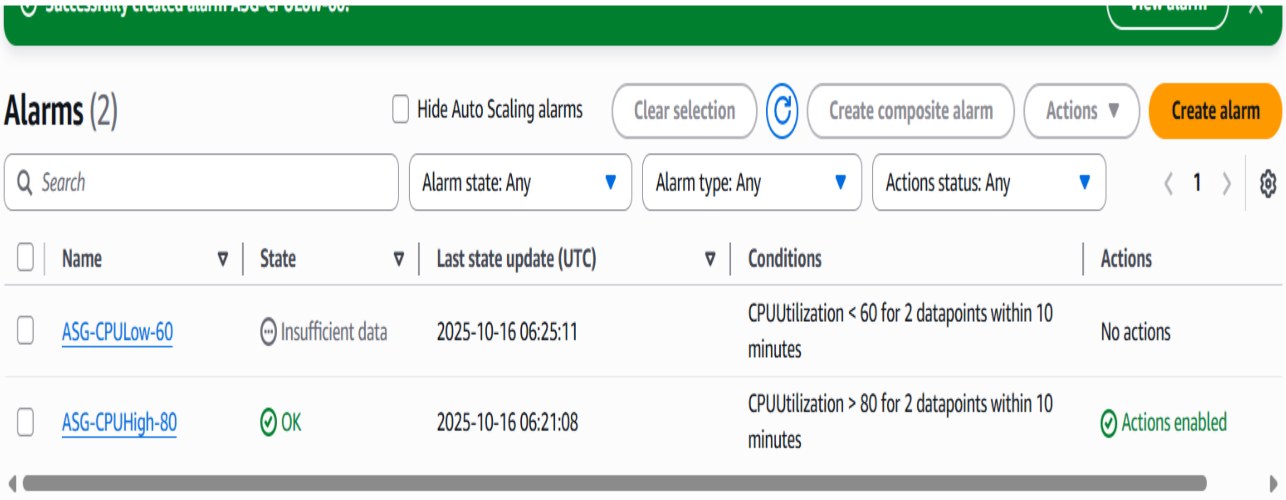
**Alarm description - *optional*** View formatting guidelines

Edit | Preview

*# This is an H1*
***double asterisks will produce strong character***
*This is [an example](https://example.com/) inline link.*

Up to 1024 characters (0/1024)

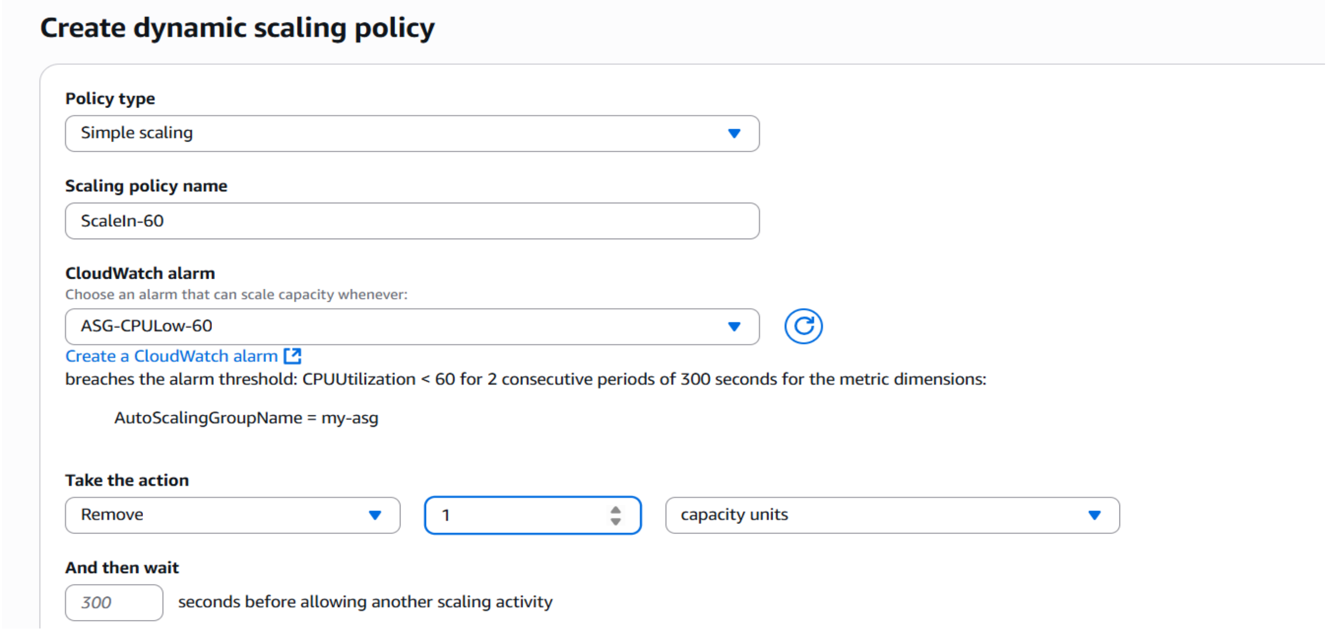
ⓘ Markdown formatting is only applied when viewing your alarm in the console. The description will remain in plain text in the alarm notifications.



Successfully created alarm ASG-CPULow-60. View alarm

**Alarms (2)**

| | Name | State | Last state update (UTC) | Conditions | Actions |
|---|---|---|---|---|---|
| ☐ | ASG-CPULow-60 | Insufficient data | 2025-10-16 06:25:11 | CPUUtilization < 60 for 2 datapoints within 10 minutes | No actions |
| ☐ | ASG-CPUHigh-80 | ✓ OK | 2025-10-16 06:21:08 | CPUUtilization > 80 for 2 datapoints within 10 minutes | ✓ Actions enabled |



**Create dynamic scaling policy**

**Policy type**

Simple scaling

**Scaling policy name**

ScaleIn-60

**CloudWatch alarm**

Choose an alarm that can scale capacity whenever:

ASG-CPULow-60

Create a CloudWatch alarm

breaches the alarm threshold: CPUUtilization < 60 for 2 consecutive periods of 300 seconds for the metric dimensions:

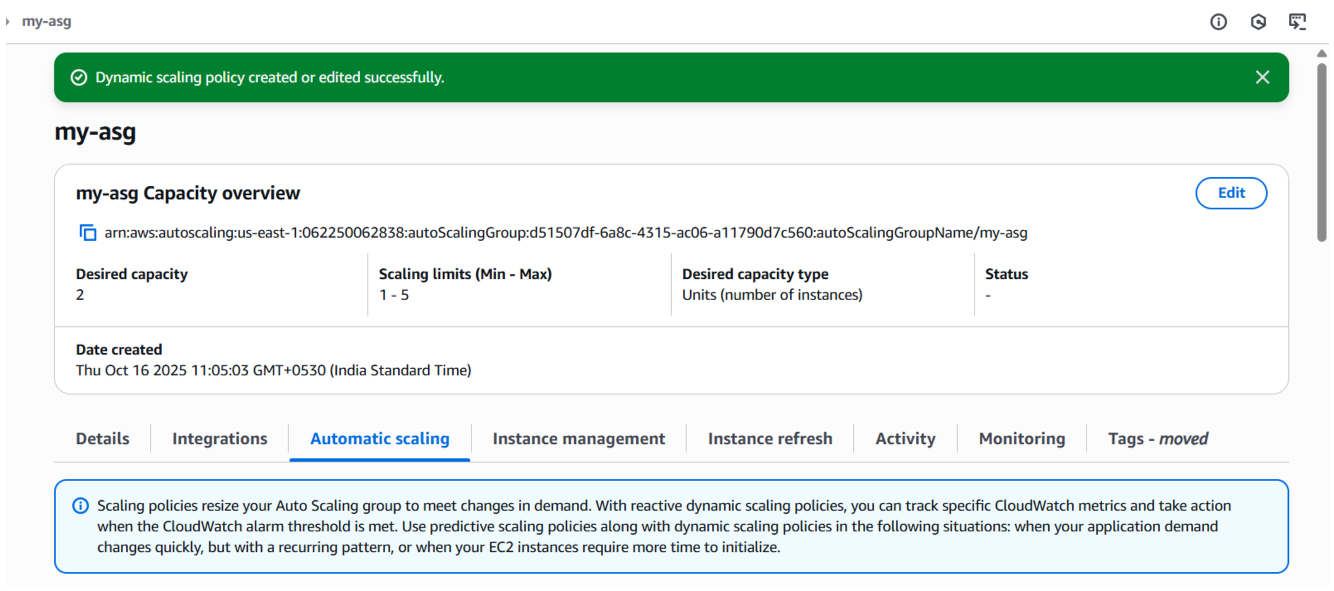
AutoScalingGroupName = my-asg

**Take the action**

Remove | 1 | capacity units

**And then wait**

300 seconds before allowing another scaling activity



my-asg

Dynamic scaling policy created or edited successfully.

my-asg

my-asg Capacity overview

Edit

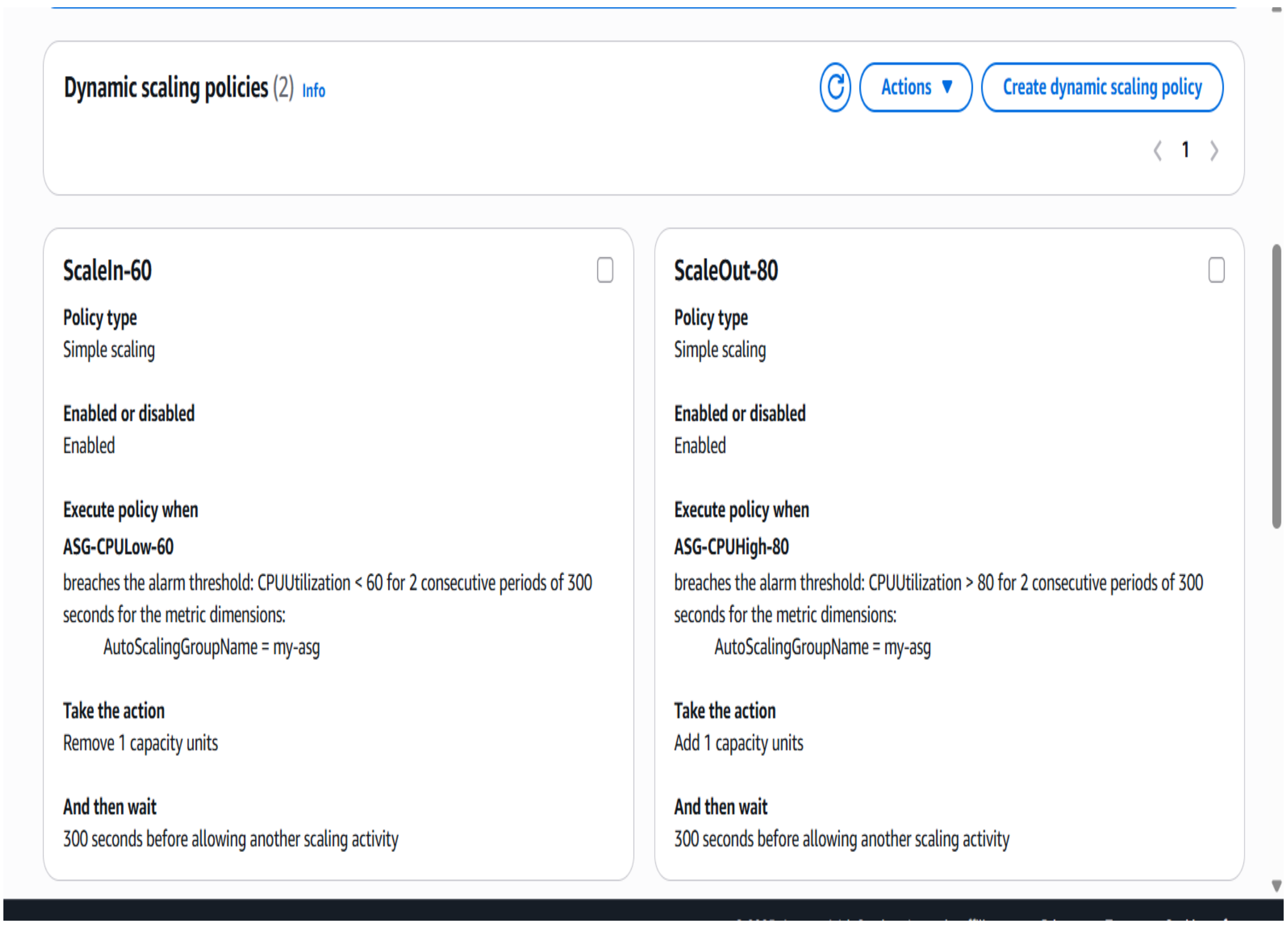
arn:aws:autoscaling:us-east-1:062250062838:autoScalingGroup:d51507df-6a8c-4315-ac06-a11790d7c560:autoScalingGroupName/my-asg

| Desired capacity | Scaling limits (Min - Max) | Desired capacity type | Status |
|---|---|---|---|
| 2 | 1 - 5 | Units (number of instances) | - |

Date created
Thu Oct 16 2025 11:05:03 GMT+0530 (India Standard Time)

Details | Integrations | Automatic scaling | Instance management | Instance refresh | Activity | Monitoring | Tags - *moved*

Scaling policies resize your Auto Scaling group to meet changes in demand. With reactive dynamic scaling policies, you can track specific CloudWatch metrics and take action when the CloudWatch alarm threshold is met. Use predictive scaling policies along with dynamic scaling policies in the following situations: when your application demand changes quickly, but with a recurring pattern, or when your EC2 instances require more time to initialize.



Dynamic scaling policies (2) Info

Actions ▼ | Create dynamic scaling policy

‹ 1 ›

**ScaleIn-60**

Policy type
Simple scaling

Enabled or disabled
Enabled

Execute policy when
ASG-CPULow-60
breaches the alarm threshold: CPUUtilization < 60 for 2 consecutive periods of 300 seconds for the metric dimensions:
AutoScalingGroupName = my-asg

Take the action
Remove 1 capacity units

And then wait
300 seconds before allowing another scaling activity

**ScaleOut-80**

Policy type
Simple scaling

Enabled or disabled
Enabled

Execute policy when
ASG-CPUHigh-80
breaches the alarm threshold: CPUUtilization > 80 for 2 consecutive periods of 300 seconds for the metric dimensions:
AutoScalingGroupName = my-asg

Take the action
Add 1 capacity units

And then wait
300 seconds before allowing another scaling activity

## STEP 5: Test the Setup

Open the ALB DNS name in your browser — you'll see:

Welcome to my App - ip-xx-xx-xx-xx

SSH into an instance:

sudo yum install stress -y

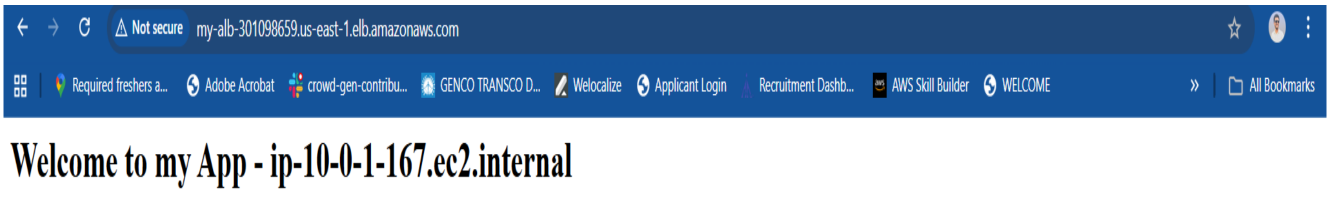
stress --cpu 4 --timeout 300

Go to EC2 → Auto Scaling → Activity tab.

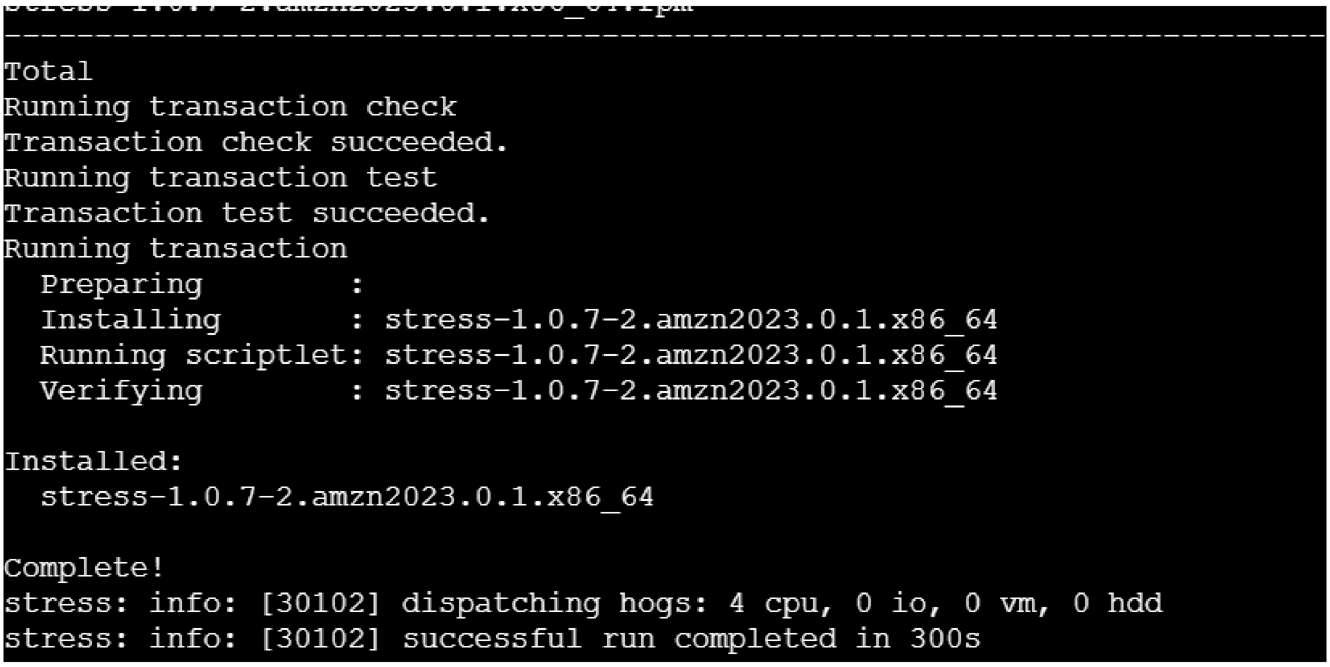
You'll see new instances launch when CPU > 80%

They'll terminate when CPU < 60%

---



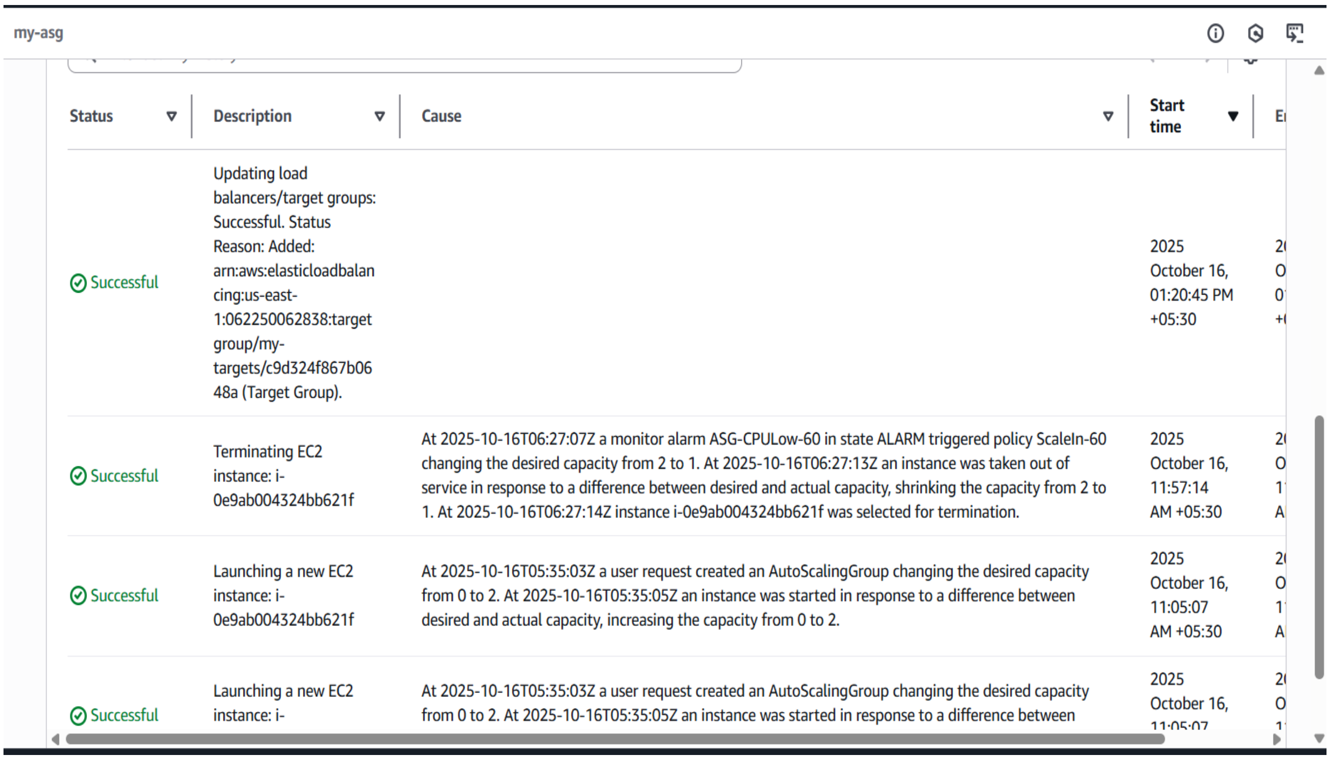
Welcome to my App - ip-10-0-1-167.ec2.internal



```
--------------------------------------------------------------------------
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Installing       : stress-1.0.7-2.amzn2023.0.1.x86_64
  Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64
  Verifying        : stress-1.0.7-2.amzn2023.0.1.x86_64

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

Complete!
stress: info: [30102] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
stress: info: [30102] successful run completed in 300s
```


my-asg

| Status | Description | Cause | Start time | E |
|---|---|---|---|---|
| Successful | Updating load balancers/target groups: Successful. Status Reason: Added: arn:aws:elasticloadbalancing:us-east-1:062250062838:targetgroup/my-targets/c9d324f867b0648a (Target Group). | | 2025 October 16, 01:20:45 PM +05:30 | 2 O 0 + |
| Successful | Terminating EC2 instance: i-0e9ab004324bb621f | At 2025-10-16T06:27:07Z a monitor alarm ASG-CPULow-60 in state ALARM triggered policy ScaleIn-60 changing the desired capacity from 2 to 1. At 2025-10-16T06:27:13Z 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 2025-10-16T06:27:14Z instance i-0e9ab004324bb621f was selected for termination. | 2025 October 16, 11:57:14 AM +05:30 | 2 O 1 A |
| Successful | Launching a new EC2 instance: i-0e9ab004324bb621f | At 2025-10-16T05:35:03Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-10-16T05:35:05Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2. | 2025 October 16, 11:05:07 AM +05:30 | 2 O 1 A |
| Successful | Launching a new EC2 instance: i- | At 2025-10-16T05:35:03Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-10-16T05:35:05Z an instance was started in response to a difference between | 2025 October 16, 11:05:07 | 2 O 1 |

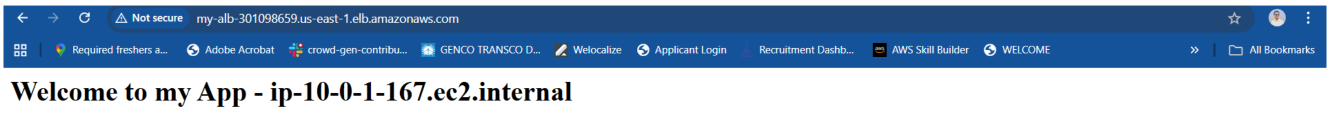
## STEP 6: Routing (No Domain Version)

### Route the Company Domain

Since no custom domain was available, I used the default AWS Load Balancer DNS name to access the application.

Example: http://XYZ-ALB-1234567890.ap-south-1.elb.amazonaws.com

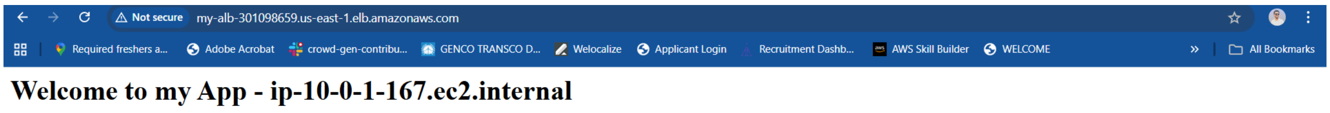
This DNS name is automatically managed by AWS and routes traffic to the EC2 instances via the Application Load Balancer.


Not secure my-alb-301098659.us-east-1.elb.amazonaws.com

Welcome to my App - ip-10-0-1-167.ec2.internal

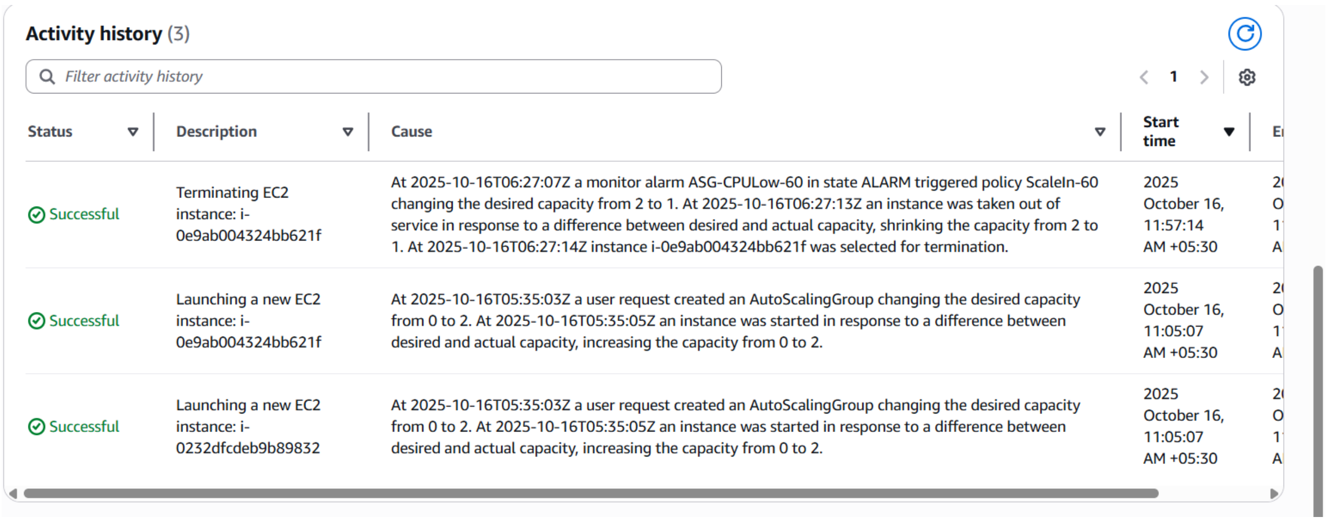
## ☑ Final Verification

- Open the Load Balancer DNS name — it should display your EC2 web page (Welcome to my App - <hostname>).
- Refresh multiple times — you should see different instance hostnames, confirming load balancing.
- During high CPU load, Auto Scaling automatically adds new instances based on your scaling policies.
- When the load decreases, Auto Scaling terminates the extra instances automatically.



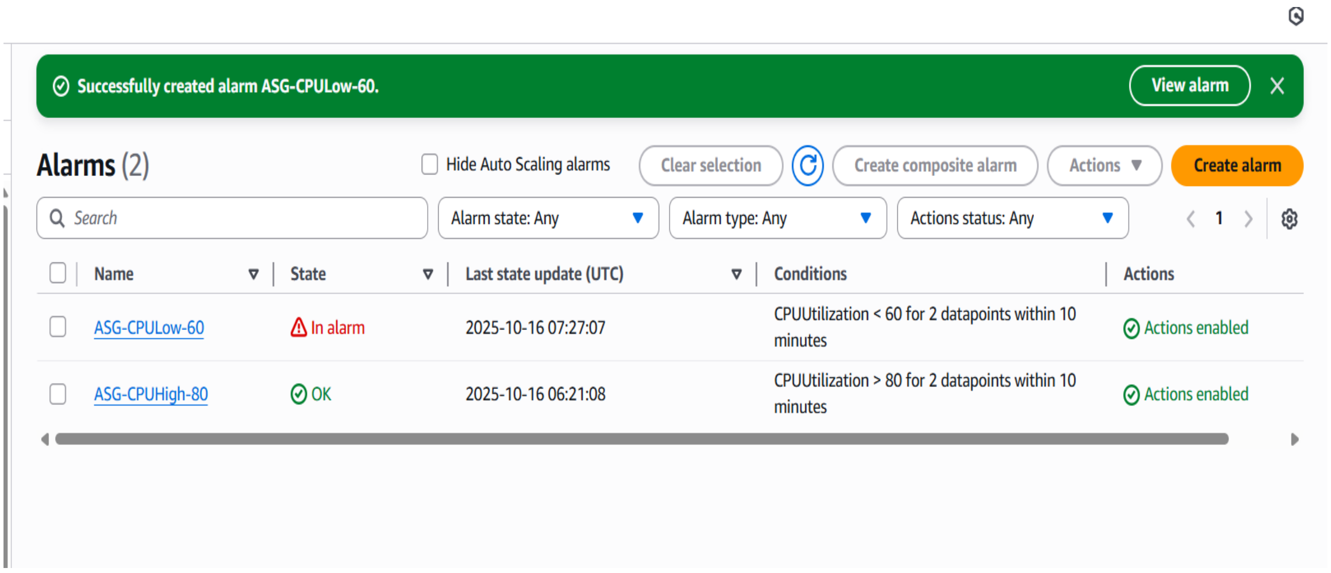
Not secure my-alb-301098659.us-east-1.elb.amazonaws.com

Welcome to my App - ip-10-0-1-167.ec2.internal



Activity history (3)

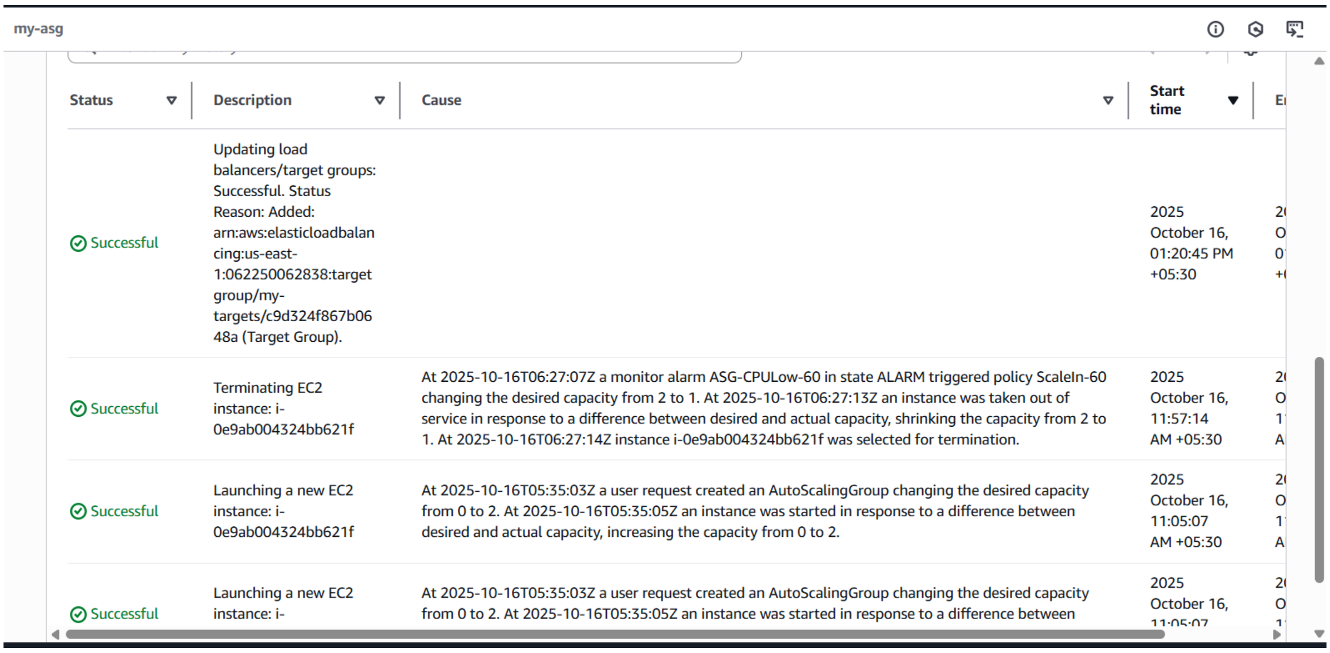
| Status | Description | Cause | Start time |
|---|---|---|---|
| Successful | Terminating EC2 instance: i-0e9ab004324bb621f | At 2025-10-16T06:27:07Z a monitor alarm ASG-CPULow-60 in state ALARM triggered policy ScaleIn-60 changing the desired capacity from 2 to 1. At 2025-10-16T06:27:13Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-10-16T06:27:14Z instance i-0e9ab004324bb621f was selected for termination. | 2025 October 16, 11:57:14 AM +05:30 |
| Successful | Launching a new EC2 instance: i-0e9ab004324bb621f | At 2025-10-16T05:35:03Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-10-16T05:35:05Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2. | 2025 October 16, 11:05:07 AM +05:30 |
| Successful | Launching a new EC2 instance: i-0232dfcdeb9b89832 | At 2025-10-16T05:35:03Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-10-16T05:35:05Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2. | 2025 October 16, 11:05:07 AM +05:30 |



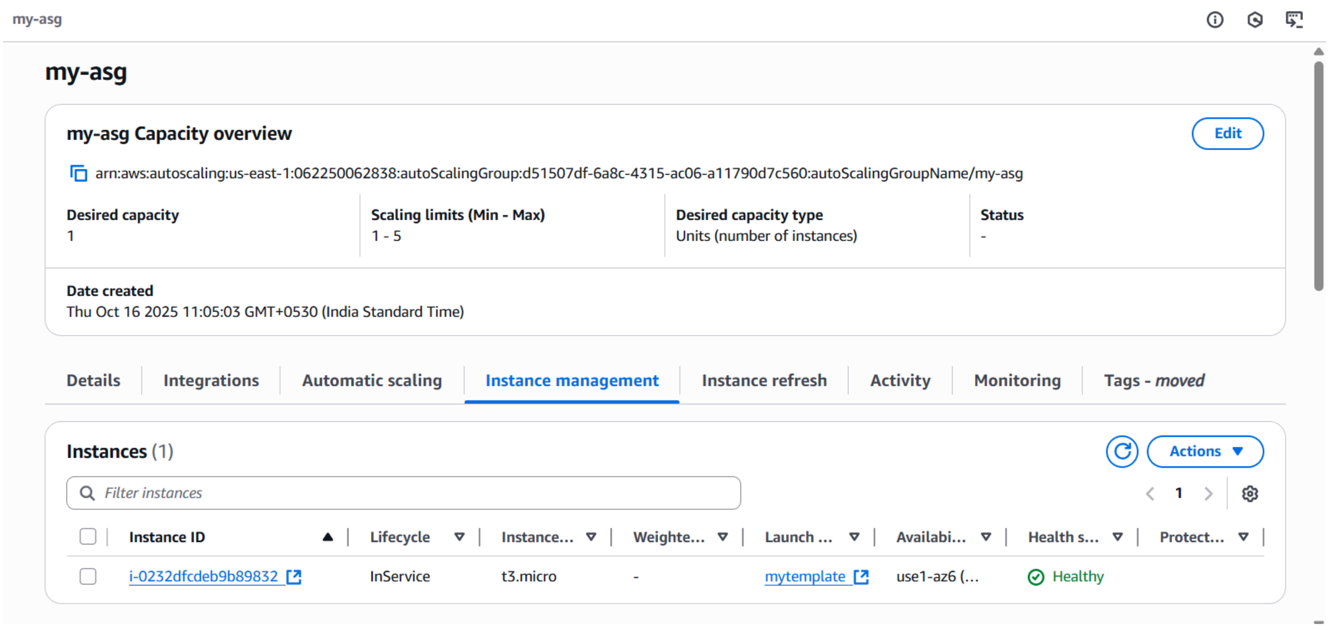
Successfully created alarm ASG-CPULow-60.

Alarms (2)

| Name | State | Last state update (UTC) | Conditions | Actions |
|---|---|---|---|---|
| ASG-CPULow-60 | In alarm | 2025-10-16 07:27:07 | CPUUtilization < 60 for 2 datapoints within 10 minutes | Actions enabled |
| ASG-CPUHigh-80 | OK | 2025-10-16 06:21:08 | CPUUtilization > 80 for 2 datapoints within 10 minutes | Actions enabled |


my-asg

| Status | Description | Cause | Start time |
|---|---|---|---|
| Successful | Updating load balancers/target groups: Successful. Status Reason: Added: arn:aws:elasticloadbalancing:us-east-1:062250062838:targetgroup/my-targets/c9d324f867b0648a (Target Group). | | 2025 October 16, 01:20:45 PM +05:30 |
| Successful | Terminating EC2 instance: i-0e9ab004324bb621f | At 2025-10-16T06:27:07Z a monitor alarm ASG-CPULow-60 in state ALARM triggered policy ScaleIn-60 changing the desired capacity from 2 to 1. At 2025-10-16T06:27:13Z 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 2025-10-16T06:27:14Z instance i-0e9ab004324bb621f was selected for termination. | 2025 October 16, 11:57:14 AM +05:30 |
| Successful | Launching a new EC2 instance: i-0e9ab004324bb621f | At 2025-10-16T05:35:03Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-10-16T05:35:05Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2. | 2025 October 16, 11:05:07 AM +05:30 |
| Successful | Launching a new EC2 instance: i- | At 2025-10-16T05:35:03Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-10-16T05:35:05Z an instance was started in response to a difference between | 2025 October 16, 11:05:07 |


my-asg

## my-asg

**my-asg Capacity overview**

Edit

arn:aws:autoscaling:us-east-1:062250062838:autoScalingGroup:d51507df-6a8c-4315-ac06-a11790d7c560:autoScalingGroupName/my-asg

| Desired capacity | Scaling limits (Min - Max) | Desired capacity type | Status |
|---|---|---|---|
| 1 | 1 - 5 | Units (number of instances) | - |

**Date created**
Thu Oct 16 2025 11:05:03 GMT+0530 (India Standard Time)

**Instances** (1)

| Instance ID | Lifecycle | Instance… | Weighte… | Launch … | Availabi… | Health s… | Protect… |
|---|---|---|---|---|---|---|---|
| i-0232dfcdeb9b89832 | InService | t3.micro | - | mytemplate | use1-az6 (… | Healthy | |