

# Autoscaling Group

## Step 1: Create a Launch Template

A launch template is a reusable configuration for launching EC2 instances with predefined settings like AMI, instance type, key pair, etc.

### 1.1 Navigate to Launch Templates

1. Go to the **AWS Management Console** and navigate to the **EC2 Dashboard**.
2. From the left-hand menu, select **Launch Templates**.
3. Click on the **Create Launch Template** button.

### 1.2 Configure the Launch Template

1. **Template Name:** Give your launch template a unique name (e.g., MyApp-Launch-Template).
2. **Template Version:** Leave this as 1 (for the first version).
3. **AMI (Amazon Machine Image):** Choose the Amazon Machine Image (AMI) that contains the OS and applications you need. You can select Windows, Ubuntu, or any other OS. Make sure the AMI includes the software your EC2 instances will run.
4. **Instance Type:** Select an instance type (e.g., t2.micro for basic workloads).
5. **Key Pair:** Choose or create an existing key pair for RDP (Windows) access.
6. **Network Settings:**
  - **VPC:** Select the VPC where you want the instances to launch.
  - **Subnet:** Select the subnet(s) (if needed). If you want instances to be launched in multiple Availability Zones, you can leave this unset.

### 1.3 Review and Create

1. Review the configuration.
2. Click **Create Launch Template**.

## Step 2: Create an Auto Scaling Group (ASG)

An Auto Scaling Group automatically manages a group of EC2 instances, scaling them based on demand.

### 2.1 Navigate to Auto Scaling Groups

1. Go to the **EC2 Dashboard**.
2. From the left-hand menu, select **Auto Scaling Groups**.
3. Click **Create Auto Scaling Group**.

### 2.2 Configure the Auto Scaling Group

1. **Auto Scaling Group Name:** Provide a unique name (e.g., myasg).
2. **Launch Template:** Select the launch template created in Step 1.
3. **Version:** Choose the latest version of the launch template.

### 2.3 Select Network Settings

1. **VPC:** Choose the VPC where your instances will run.
2. **Subnets:** Select multiple subnets in different Availability Zones to improve fault tolerance

### 2.4 Configure Group Size and Scaling Policies

1. **Desired Capacity:** Set the number of instances that should always be running (e.g., 2).
2. **Minimum Capacity:** Set the minimum number of instances (e.g., 1).
3. **Maximum Capacity:** Set the maximum number of instances (e.g., 3).

## 2.5 Health Check

1. **Health Check Type:** Choose **EC2** or **ELB** depending on whether you're using a load balancer.
2. **Health Check Grace Period:** Set a time (in seconds) for how long an instance can be in the "initializing" state before it's considered healthy (e.g., 120 seconds).

## 2.6 Review and Create

1. Review the settings.
2. Click **Create Auto Scaling Group**.

## Step 3: Configure Target Tracking Scaling

Target tracking scaling automatically adjusts the number of instances based on a target metric (e.g., CPU utilization).

### 3.1 Navigate to Auto Scaling Group Policies

1. Once your ASG is created, navigate to the **Auto Scaling Group** you just created.
2. Click the **Automatic Scaling** tab.
3. Click **Add Policy** to create a scaling policy.

### 3.2 Create Target Tracking Scaling Policy

1. **Policy Type:** Select **Target Tracking Scaling**.
2. **Metric Type:** Choose **Average CPU Utilization** (other metrics like request count, network throughput, or custom CloudWatch metrics can also be used).
3. **Target Value:** Set the CPU utilization target. For your case, set this to **35%**.
  - This means AWS will automatically adjust the number of instances to maintain an average CPU utilization of 35%.
4. **Instance Warm-up Time:** Specify a warm-up period (e.g., 120 seconds) to allow new instances to stabilize before contributing to metrics.
5. **Disable Scale-in (Optional):** If you only want to scale out and not scale in, you can enable this. Usually, you leave it unchecked to allow both scaling out and in.

### 3.3 Review and Create Policy

1. Review your settings and click **Create Policy**.

## Step 4: Monitor and Adjust Auto Scaling

1. **Monitor Scaling:** Go to **Amazon CloudWatch** to monitor the metrics (e.g., CPU utilization) and see how instances are scaling.
2. **Tune Settings:** If needed, adjust the **target value**, **min/max capacity**, or other parameters based on how your application behaves under load.

## Target Tracking Scaling Overview

**Target tracking scaling** automatically adjusts the size of your Auto Scaling Group to maintain a specified target for a predefined metric, typically **average CPU utilization**. When you set a target value (e.g., 35% CPU utilization), the Auto Scaling Group adjusts the number of instances to keep this target value. If CPU usage goes above 35%, it adds more instances. If usage falls below 35%, it terminates instances.

### Key Benefits:

- **Hands-off Scaling:** AWS automatically manages the scaling of your instances, requiring minimal manual intervention.

- **Improved Performance:** Ensures consistent performance for your application by keeping CPU (or other metrics) within the desired range.
- **Cost Efficiency:** Reduces costs by terminating instances when demand decreases, ensuring you are only paying for what you need.

Below are screenshot of implementation

Launching template:

The screenshot shows the AWS Management Console interface for 'Launch Templates'. At the top, there's a search bar and a table of launch templates. The table has columns for Launch Template ID, Launch Template Name, Default Version, Latest Version, Create Time, and Created By. One template is listed: 'mytemplate' with ID 'lt-0ebe81dbe8ec9b577'. Below the table, the details for 'mytemplate' are shown. The 'Launch template details' section includes the Launch template ID, Launch template name, Default version, and Owner. The 'Launch template version details' section shows the Version (1 (Default)), Description (template for autoscaling groups), Date created (2024-10-03T21:48:50.000Z), and Created by (arn:aws:iam::022499044209:root). The 'Instance details' tab is selected, showing the AMI ID (ami-0888db1202897905c), Instance type (t2.micro), Availability Zone (-), Key pair name (mytemplate\_key\_pair), Security groups (-), and Security group IDs (sg-00e4a32c4349e99d5).

Creating autoscaling group with template created

EC2 > Auto Scaling groups

Auto Scaling groups (1/1) [Info](#) [Refresh](#) [Launch configurations](#) [Launch templates](#) [Actions](#) [Create Auto Scaling group](#)

<input checked="" type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min
<input checked="" type="checkbox"/>	<a href="#">my_asg</a>	<a href="#">mytemplate</a>   Version Default	0	Updating capacity...	2	1

Auto Scaling group: **my\_asg** [Settings](#) [Close](#)

[Details](#) [Activity](#) [Automatic scaling](#) [Instance management](#) [Monitoring](#) [Instance refresh](#)

**Group details** [Edit](#)

Auto Scaling group name	my_asg	Desired capacity	2	Desired capacity type	Units (number of instances)	Amazon Resource Name (ARN)
Date created	Fri Oct 04 2024 03:20:18 GMT+0530 (India Standard Time)	Minimum capacity	1	Status	Updating capacity	arn:aws:autoscaling:us-east-1:022499044209:autoScalingGroup:f52aafb2-05ab-461d-b622-6d7408587e3f:autoScalingGroupName/my_asg
		Maximum capacity	3			

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Asg creating desired instances after launching it.

Status	Description	Cause	Start time
Successful	Launching a new EC2 instance: i-021829a375a7ddcbc	At 2024-10-03T21:50:18Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-10-03T21:50:29Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2024 October 04, 03:20:31 AM +05:30
Successful	Launching a new EC2 instance: i-0ef0307ca4899a173	At 2024-10-03T21:50:18Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-10-03T21:50:29Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2024 October 04, 03:20:31 AM +05:30

Auto Scaling group: **my\_asg** [Settings](#) [Close](#)

[Details](#) [Activity](#) [Automatic scaling](#) [Instance management](#) [Monitoring](#) [Instance refresh](#)

**Instances (2)** [Refresh](#) [Actions](#)

<input type="checkbox"/>	Instance ID	Lifecycle	Instanc...	Weight...	Launch ...	Availab...	Health ...	Protect...
<input type="checkbox"/>	<a href="#">i-021829a375a7ddcbc</a>	InService	t2.micro	-	<a href="#">mytemplate</a>	us-east-1a	Healthy	
<input type="checkbox"/>	<a href="#">i-0ef0307ca4899a173</a>	InService	t2.micro	-	<a href="#">mytemplate</a>	us-east-1b	Healthy	

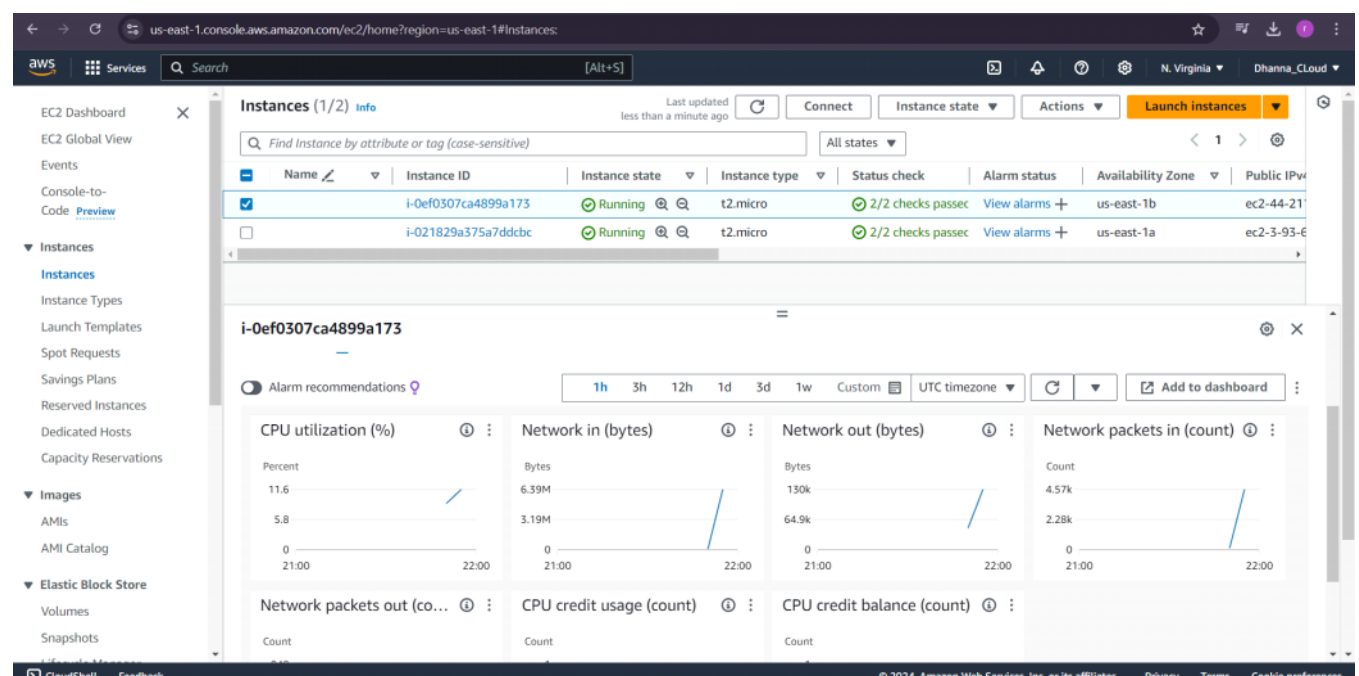
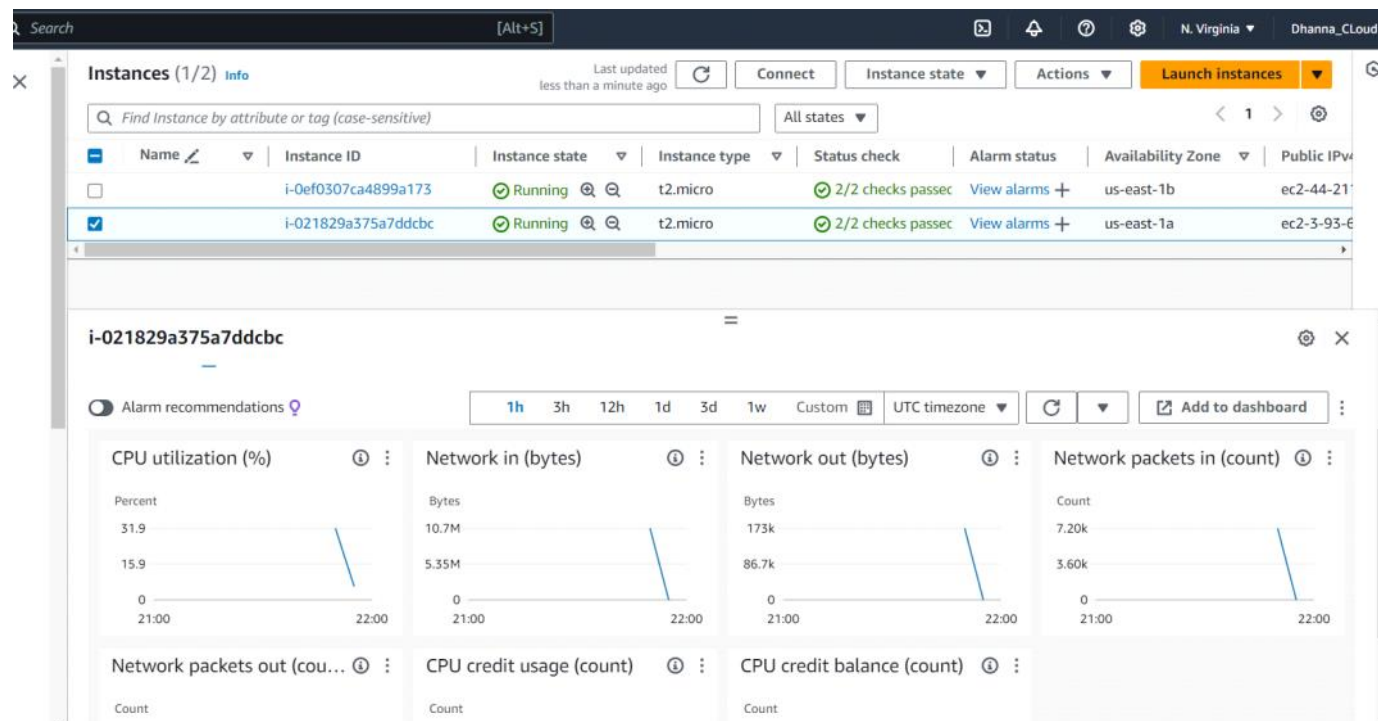
[Alt+S] [N. Virginia](#) [Dhanna\\_Cloud](#)

Instances (1/3) [Info](#) [Last updated less than a minute ago](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

[All states](#)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv...
<input type="checkbox"/>		i-066e6c6ca91155e63	Running	t2.micro	Initializing	<a href="#">View alarms</a>	us-east-1a	ec2-54-165
<input checked="" type="checkbox"/>		i-0ef0307ca4899a173	Running	t2.micro	2/2 checks pass	<a href="#">View alarms</a>	us-east-1b	ec2-44-21

Cpu utilization stats before updating dynamic policies.



Now setting dynamic policies for automating asg based on cpu utilization.

Setting cpu utilization : 35 percent ( this will maintain the minimum instances as the cpu utilization till now is very less)

## Dynamic scaling policies (1) [Info](#)

### Target Tracking Policy ☐

Policy type

Target tracking scaling

Enabled or disabled

Enabled

Execute policy when

As required to maintain Average CPU utilization at 35

Take the action

Add or remove capacity units as required

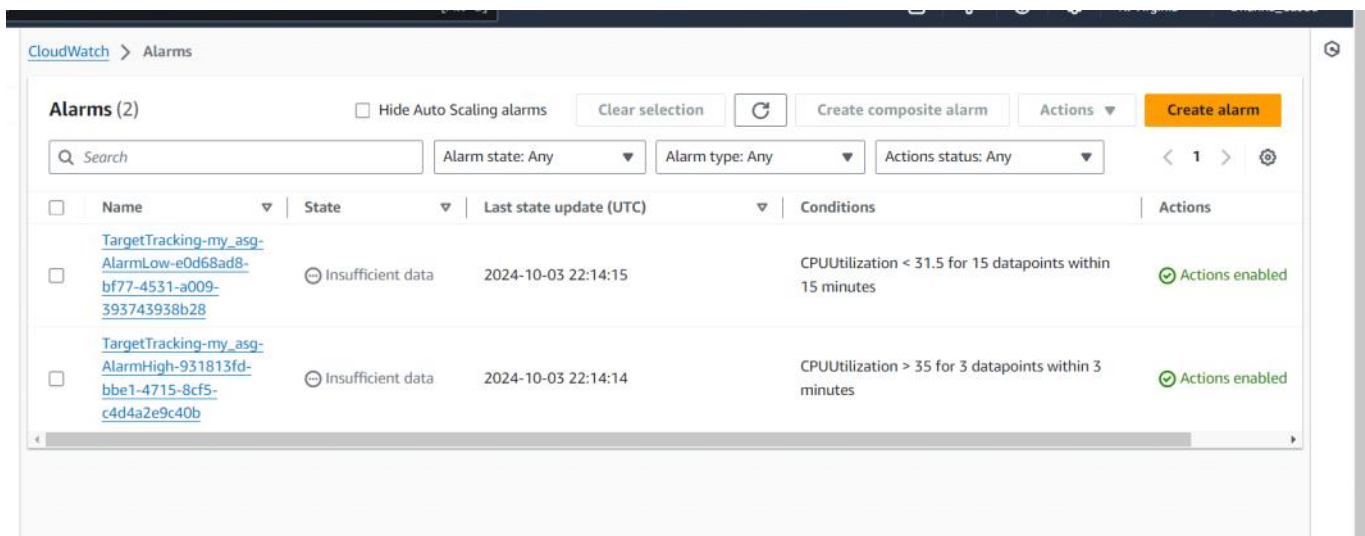
Instances need

120 seconds to warm up before including in metric

Scale in

Enabled

Cloudwatch alarm created automatically after setting dynamic policies.



The screenshot shows the AWS CloudWatch Alarms console. At the top, there's a header 'CloudWatch > Alarms'. Below it, a summary bar shows 'Alarms (2)' with a 'Hide Auto Scaling alarms' checkbox, 'Clear selection', 'Create composite alarm', and a 'Create alarm' button. A search bar and filters for 'Alarm state: Any', 'Alarm type: Any', and 'Actions status: Any' are present. The main table lists two alarms:

<input type="checkbox"/>	Name	State	Last state update (UTC)	Conditions	Actions
<input type="checkbox"/>	<a href="#">TargetTracking-my_asg-AlarmLow-e0d68ad8-bf77-4531-a009-393743938b28</a>	Insufficient data	2024-10-03 22:14:15	CPUUtilization < 31.5 for 15 datapoints within 15 minutes	Actions enabled
<input type="checkbox"/>	<a href="#">TargetTracking-my_asg-AlarmHigh-931813fd-bbe1-4715-8cf5-c4d4a2e9c40b</a>	Insufficient data	2024-10-03 22:14:14	CPUUtilization > 35 for 3 datapoints within 3 minutes	Actions enabled

As the utilization is very low it is automatically switing off the unnecessary instances and maintaining the minimum quantity specifies by me while setting syanamic policy ( i.e 1)



Instances (1/2) Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

All states

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
<input type="checkbox"/>		i-0ef0307ca4899a173	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-44-21...
<input checked="" type="checkbox"/>		i-021829a375a7ddcbc	Shutting-d...	t2.micro	-	View alarms +	us-east-1a	ec2-3-93-6...

i-021829a375a7ddcbc

Alarm recommendations

1h 3h 12h 1d 3d 1w Custom UTC timezone

Add to dashboard

CPU utilization (%)

Percent

36.4 18.2 0

21:20 22:20

Network in (bytes)

Bytes

11.7M 5.84M 0

21:20 22:20

Network out (bytes)

Bytes

173k 86.7k 0

21:20 22:20

Network packets in (count)

Count

7.83k 3.92k 0

21:20 22:20

Network packets out (co...)

Count

CPU credit usage (count)

Count

CPU credit balance (count)

Count

Activity history (3)

Filter activity history

Status	Description	Cause	Start time
Successful	Terminating EC2 instance: i-021829a375a7ddcbc	At 2024-10-03T22:21:09Z a monitor alarm TargetTracking-my_asg-AlarmLow-e78bc2ec-3a31-46c0-b31f-fbd9cae656aa in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 1. At 2024-10-03T22:21:22Z 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 2024-10-03T22:21:22Z instance i-021829a375a7ddcbc was selected for termination.	2024 October 04, 03:51:22 AM +05:30
Successful	Launching a new EC2 instance: i-021829a375a7ddcbc	At 2024-10-03T21:50:18Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-10-03T21:50:29Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2024 October 04, 03:20:31 AM +05:30
Successful	Launching a new EC2 instance: i-0ef0307ca4899a173	At 2024-10-03T21:50:18Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-10-03T21:50:29Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2024 October 04, 03:20:31 AM +05:30

my\_asg

Details Activity Automatic scaling Instance management Monitoring Instance refresh

Instances (1)

Filter instances

	Instance ID	Lifecycle	Instanc...	Weighte...	Launch ...	Availabi...	Health s...	Protect...
<input type="checkbox"/>	i-0ef0307ca4899a173	InService	t2.micro	-	mytemplate	us-east-1b	Healthy	

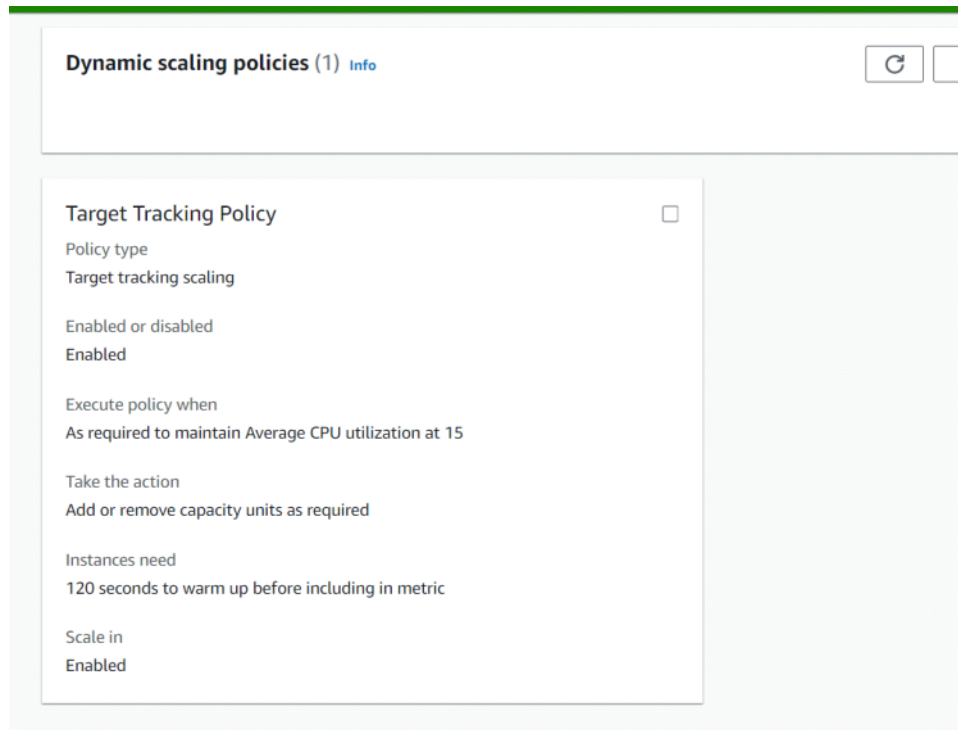
Lifecycle hooks (0) Info

Filter lifecycle hooks

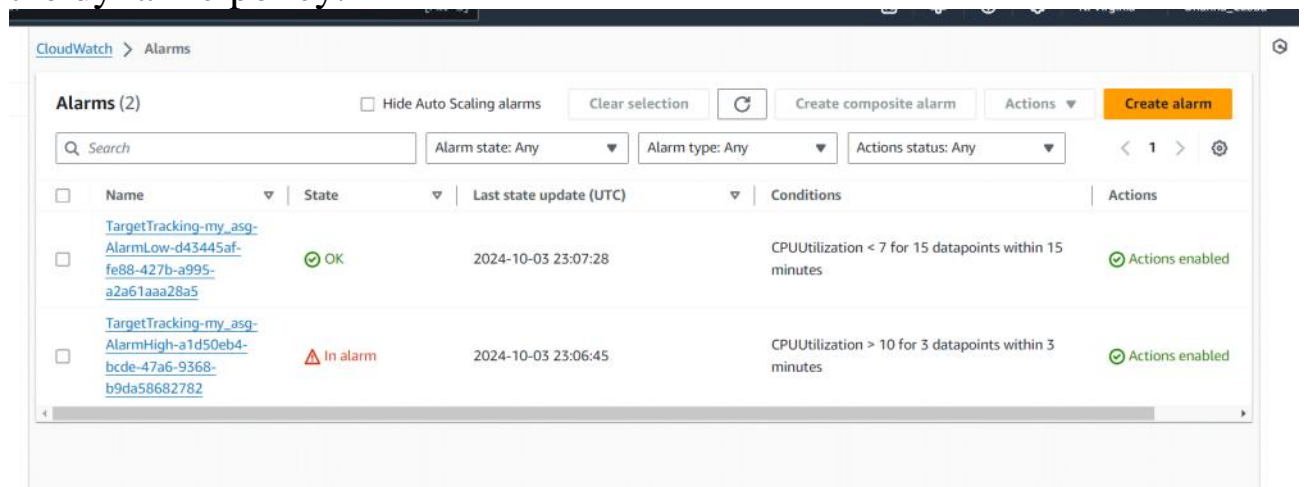
Create lifecycle hook

For maximum capacity of asg, updating the dynamic policy i.e to decreasing the cpu utilization to 15 percent. ( the cpu utilization

was not going above 35 percent as no that much load to ec2s , that's why for showing decreasing the cpu utilization in dynamic policy)



Again the cloud watch alarm is created automatically after setting the dynamic policy.



Now as the cpu utilization goes up than the specified limit So the cloudwatch alarmed and hence started launching ec2 in to its maximum level. ( i.e here 3)



Activity history (5)			
<div> <input type="text" value="Filter activity history"/> <div> <div>&lt; 1 &gt;</div> <div>⚙</div> </div> </div>			
Status	Description	Cause	Start time
⌚ Waiting for instance warmup	Launching a new EC2 instance: i-040569cbd14d599da	At 2024-10-03T23:09:45Z a monitor alarm TargetTracking-my_asg-AlarmHigh-a1d50eb4-bcde-47a6-9368-b9da58682782 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 3. At 2024-10-03T23:09:57Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 3.	2024 October 04, 04:39:59 AM +05:30
✔ Successful	Launching a new EC2 instance: i-066e6c6ca91155e63	At 2024-10-03T23:06:45Z a monitor alarm TargetTracking-my_asg-AlarmHigh-a1d50eb4-bcde-47a6-9368-b9da58682782 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2024-10-03T23:06:52Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2024 October 04, 04:36:53 AM +05:30

Status	Description	Cause	Start time
✔ Successful	Launching a new EC2 instance: i-040569cbd14d599da	At 2024-10-03T23:09:45Z a monitor alarm TargetTracking-my_asg-AlarmHigh-a1d50eb4-bcde-47a6-9368-b9da58682782 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 3. At 2024-10-03T23:09:57Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 3.	2024 October 04, 04:39:59 AM +05:30
✔ Successful	Launching a new EC2 instance: i-066e6c6ca91155e63	At 2024-10-03T23:06:45Z a monitor alarm TargetTracking-my_asg-AlarmHigh-a1d50eb4-bcde-47a6-9368-b9da58682782 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2024-10-03T23:06:52Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2024 October 04, 04:36:53 AM +05:30

EC2

>

Auto Scaling groups

>

my\_asg

my\_asg

Details

Activity

Automatic scaling

Instance management

Monitoring

Instance refresh

Instances (3)

Filter instances

< 1 >

<input type="checkbox"/>	Instance ID	Lifecycle	Instanc...	Weight...	Launch ...	Availab...	Health ...	Protect...
<input type="checkbox"/>	<a href="#">i-040569cbd14d599da</a>	InService	t2.micro	-	<a href="#">mytemplate</a>	us-east-1a	Healthy	
<input type="checkbox"/>	<a href="#">i-066e6c6ca91155e63</a>	InService	t2.micro	-	<a href="#">mytemplate</a>	us-east-1a	Healthy	
<input type="checkbox"/>	<a href="#">i-0ef0307ca4899a173</a>	InService	t2.micro	-	<a href="#">mytemplate</a>	us-east-1b	Healthy	

Lifecycle hooks (0)

info

Actions

Create lifecycle hook

Now three instances are in healthy running state.

