

## Placement Empowerment Program

### Cloud Computing and DevOps Centre

Implement Auto-scaling in the CloudSet up an auto-scaling group for your cloud VMs to handle variable workloads.

Name: Vishali V

Department:  
CSE

# Introduction

As modern applications face varying workloads, ensuring optimal performance and availability is critical. Auto Scaling, a feature provided by cloud platforms like AWS, dynamically adjusts computing resources in response to demand changes. This Proof of Concept (PoC) demonstrates how to set up an Auto Scaling Group (ASG) for virtual machines (VMs) to handle fluctuating workloads effectively. It explores defining launch configurations, setting scaling policies, and testing automatic scaling based on CPU usage.

## Overview

This PoC focuses on implementing a scalable architecture using AWS Auto Scaling Groups. The workflow includes:

1. Defining a Launch Template: Configuring virtual machines (VMs) with required specifications like instance type, AMI, key pairs, and security groups.
2. Creating an Auto Scaling Group: Setting initial group size and linking it to the launch template to manage instances dynamically.
3. Configuring Scaling Policies: Setting up metrics like CPU utilization to trigger scaling actions (e.g., scaling up during high CPU usage).
4. Testing Auto Scaling: Simulating high CPU load to verify that the ASG launches additional instances to handle demand.

This PoC will demonstrate the reliability, flexibility, and cost-efficiency of dynamic scaling in a cloud environment.

# Objective

The primary objective of this PoC is to:

1. Implement an Auto Scaling Group (ASG) to manage workloads effectively.
2. Define and configure a Launch Template for virtual machines.
3. Set up and test scaling policies based on predefined metrics, such as CPU utilization.
4. Validate the scaling process by simulating real-world scenarios (e.g., high CPU usage).

By completing this PoC, the goal is to gain hands-on experience with Auto Scaling and to understand its importance in ensuring application availability and cost management.

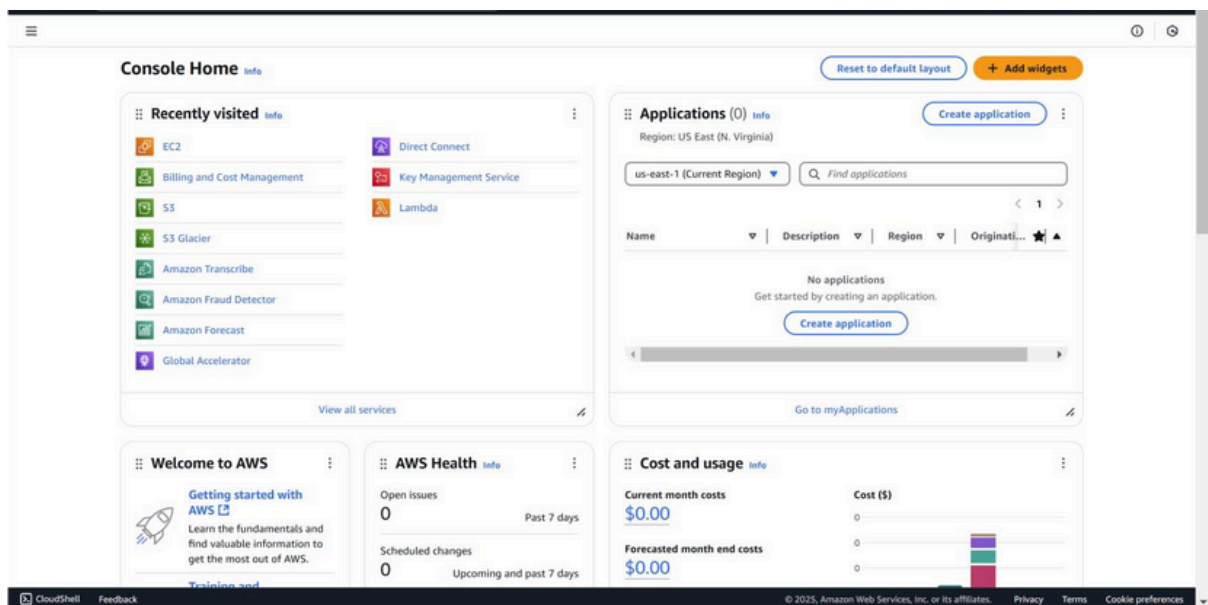
# Importance

1. Improved Application Availability: Auto Scaling ensures that applications remain available even during traffic spikes by automatically adding more VMs to meet demand.
2. Cost Optimization: It dynamically reduces the number of VMs during low traffic periods, minimizing unnecessary costs.
3. Efficient Resource Utilization: By scaling resources based on actual demand, Auto Scaling prevents over-provisioning and underutilization.
4. Resilience to Failures: Auto Scaling can replace unhealthy instances automatically, ensuring consistent application performance.
5. Real-World Relevance: The ability to manage variable workloads is a critical skill in cloud computing and aligns with industry practices.

# Step-by-Step Overview

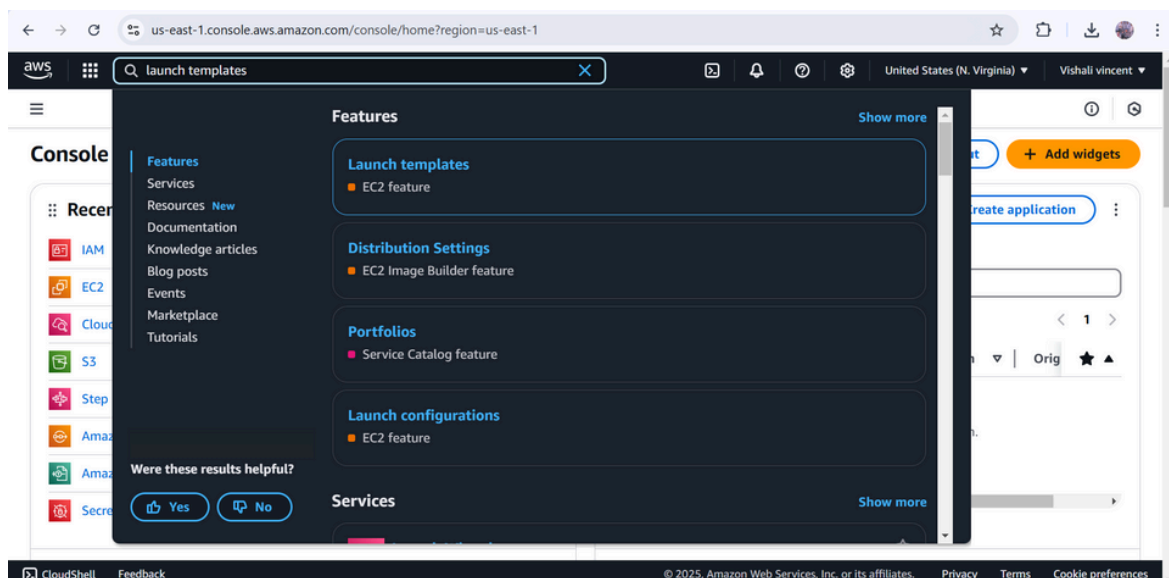
## Step 1:

1. Go to [AWS Management Console](#).
2. Enter your username and password to log in.



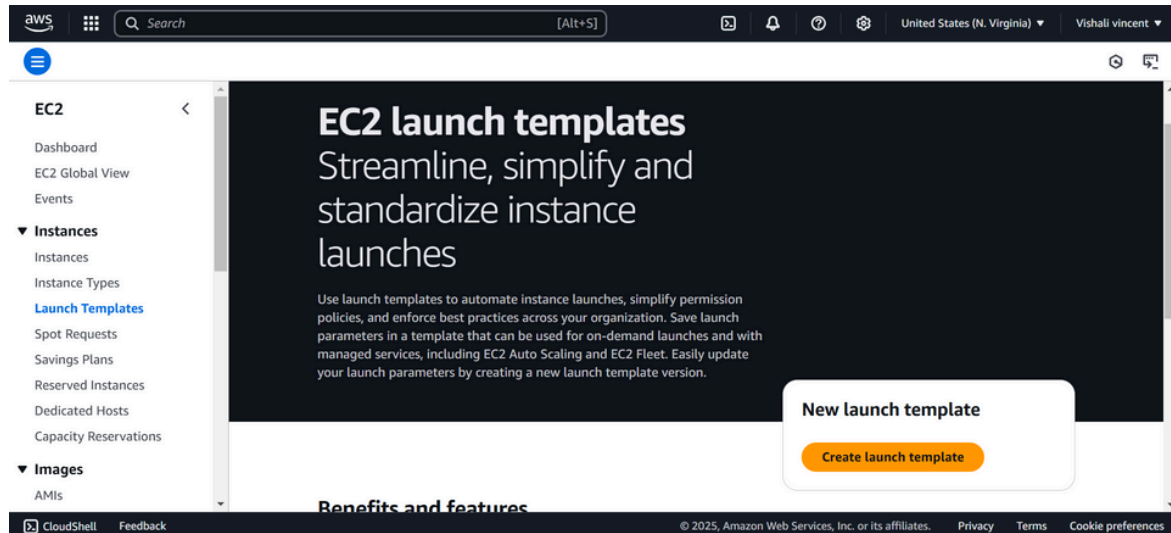
## Step 2:

Search for Launch Templates.



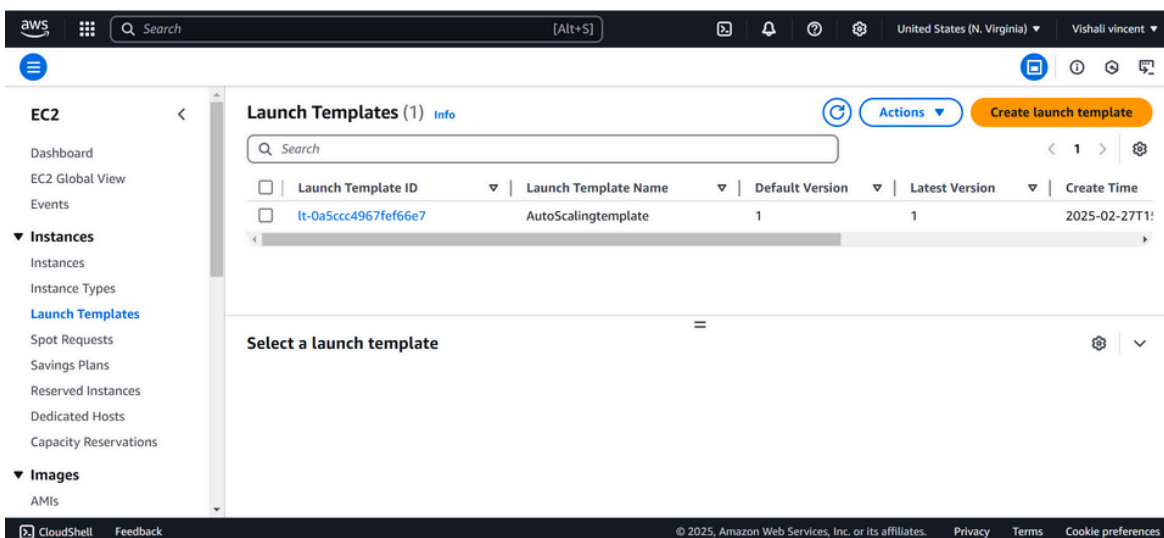
## Step 3:

Click on the Create launch template.



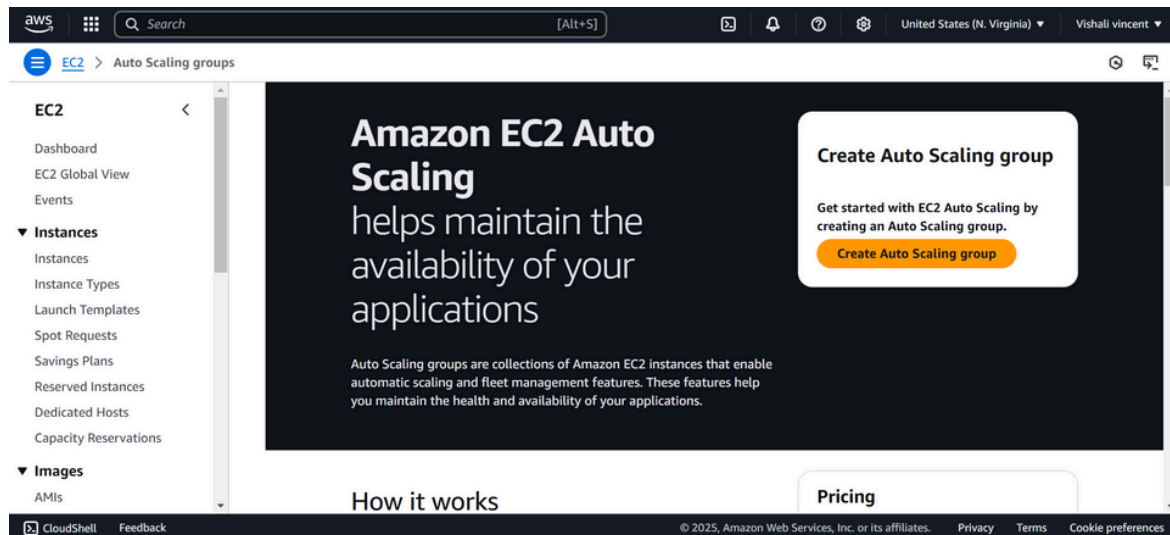
## Step 4:

Create a Launch Template named AutoScalingTemplate using an Amazon Machine Image (AMI) like Amazon Linux 2 or any default image, and choose an instance type such as t2.micro for free-tier eligibility. Select an existing key pair (or create a new one) to enable SSH access, and configure a security group that allows HTTP (port 80) and SSH (port 22). Once all details are filled out, click Create launch template to complete the setup.



## Step 5:

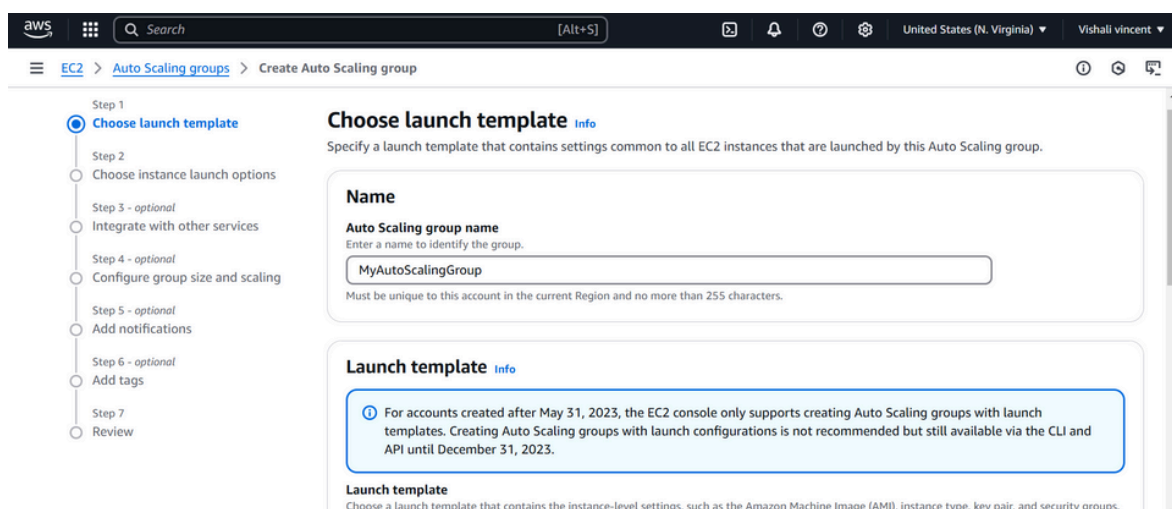
Go to the EC2 Dashboard . On the left sidebar, click on Auto Scaling Groups. Click on Create an Auto Scaling group.



## Step 6:

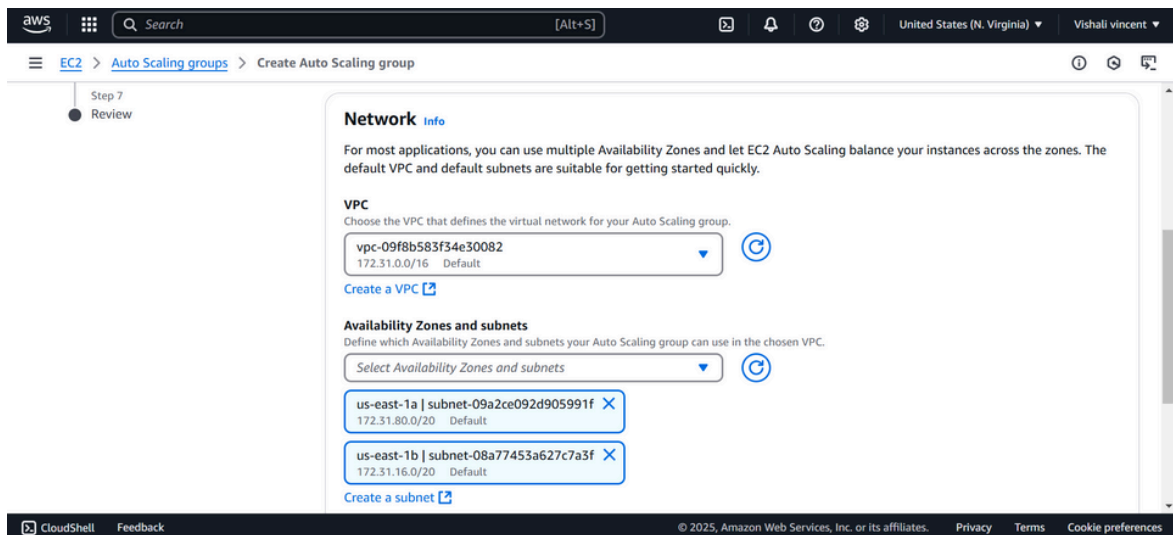
Auto Scaling group name: Give it a name (e.g., MyAutoScalingGroup).

Launch Template: Select the launch template you created earlier (AutoScalingTemplate).



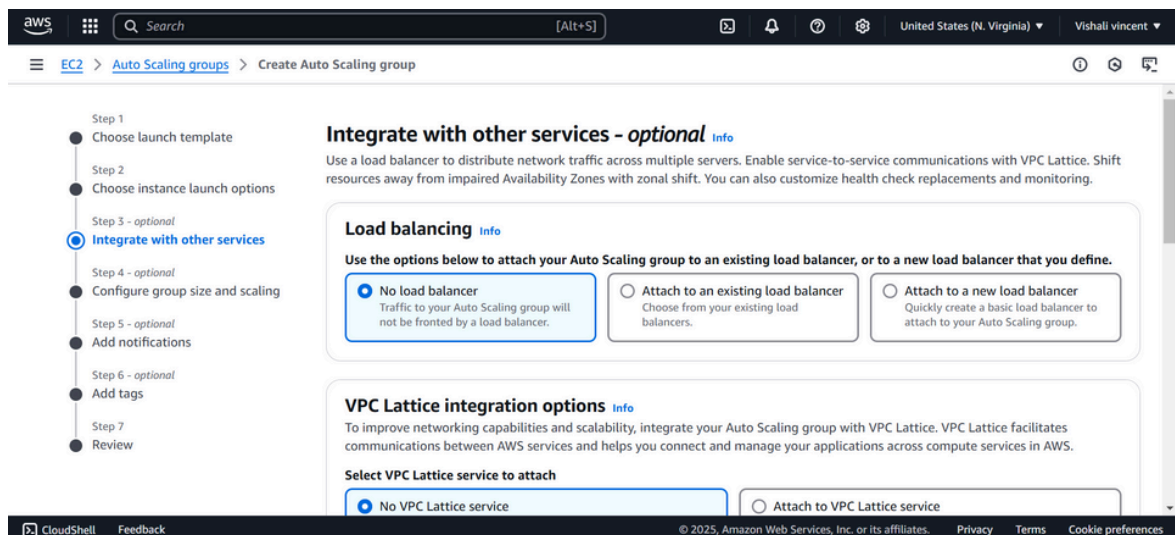
## Step 7:

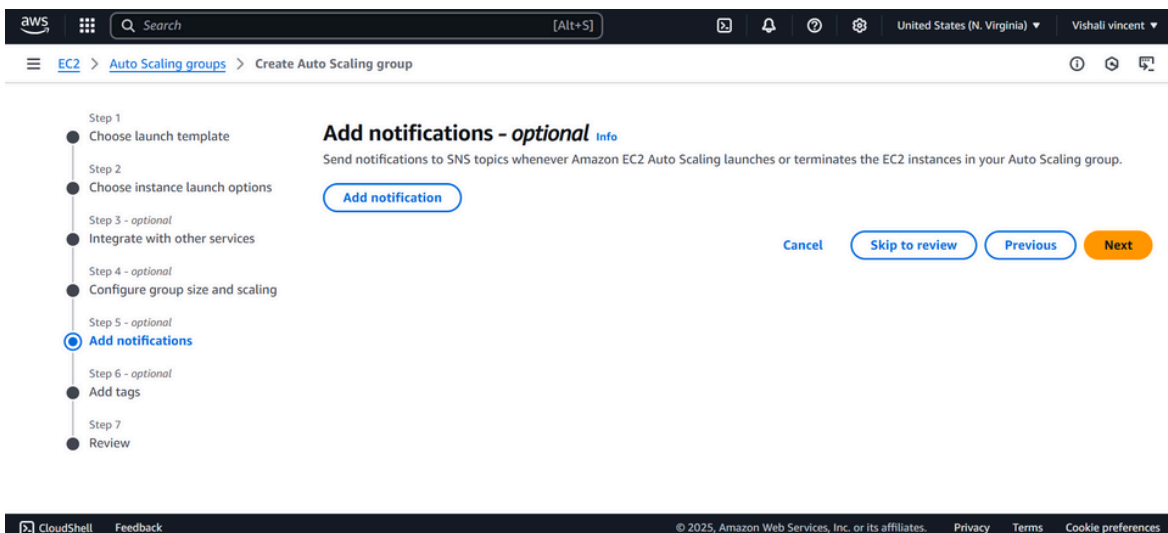
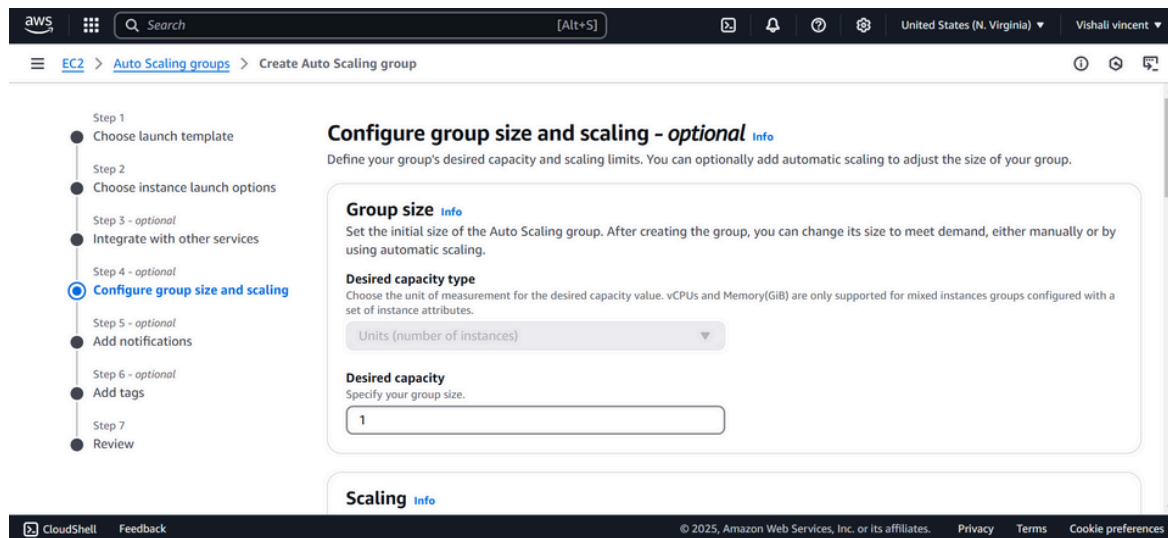
VPC and Subnets: Choose your VPC (it's fine to use the default one). Select at least two subnets in different Availability Zones (this ensures high availability).



## Step 8:

For this PoC leave the next settings as default and click next .





## Step 9:

Review all the settings you've configured. Once satisfied, click Create Auto Scaling Group.



aws

Search

[Alt+S]

United States (N. Virginia)

Vishali vincent

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1

Choose launch template

Step 2

Choose instance launch options

Step 3 - optional

Integrate with other services

Step 4 - optional

Configure group size and scaling

Step 5 - optional

Add notifications

Step 6 - optional

Add tags

Step 7

Review

Review

Info

Step 1: Choose launch template

Edit

Group details

Auto Scaling group name

MyAutoScalingGroup

Launch template

Launch template

myautoscallingtemplate

Version

Default

Description

Step 2: Choose instance launch options

Edit

Network

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws

Search

[Alt+S]

United States (N. Virginia)

Vishali vincent

EC2 > Auto Scaling groups

CloudShell

Auto Scaling groups (1)

Info

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	Ma
<input type="checkbox"/>	MyAutoScalingGroup	myautoscallingtemplate   Version Default	0	Updating capacity...	1	1	1

aws

Search

[Alt+S]

United States (N. Virginia)

Vishali vincent

EC2 > Auto Scaling groups > MyAutoScalingGroup

Info

CloudShell

MyAutoScalingGroup

MyAutoScalingGroup Capacity overview

Edit

arn:aws:autoscaling:us-east-1:970547350598:autoScalingGroup:b7440976-68dc-4706-afb0-4099263a633b:autoScalingGroupName/MyAutoScalingGroup

Desired capacity

1

Scaling limits (Min - Max)

1 - 1

Desired capacity type

Units (number of instances)

Status

-

Date created

Thu Feb 27 2025 21:40:47 GMT+0530 (India Standard Time)

Details

Integrations - new

Automatic scaling

Instance management

Instance refresh

Activity

Monitoring

Launch template

Edit

Launch template

lt-059d0ecfe86115358

AMI ID

ami-04b4f1a9cf54c11d0

Instance type

t2.micro

Owner

arn:aws:iam::970547350598:root

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Step 10:

### Testing Auto Scaling:

#### Important Note

Do Not Perform This Test If You Want to Avoid Costs:

1. Launching and running additional EC2 instances will incur charges beyond the AWS Free Tier.
2. Simulating high CPU usage and triggering scaling may increase costs temporarily due to additional resource allocation.

#### 1. Simulate High CPU Usage on an EC2 Instance

Connect to one of your EC2 instances in the Auto Scaling Group using SSH.

Run a command to create artificial CPU load. For example:

```
sudo yum install -y stress
```

```
stress --cpu 2 --timeout 300
```

This command will utilize 2 CPU cores for 5 minutes, simulating high CPU usage.

#### 2. Monitor Scaling Activities

Navigate to the AWS Management Console > EC2 Dashboard > Auto Scaling Groups.

Select your Auto Scaling Group and go to the Activity History tab.

Check if a new instance is being launched based on your scaling policy (e.g., CPU utilization exceeding 50%).

### 3. Terminate the Stress Test

Once testing is done, stop the CPU load by pressing Ctrl+C in the terminal or by terminating the stress process.

### 4. Verify Scaling Down

After the CPU usage drops, monitor the Auto Scaling Group again to confirm that unnecessary instances are terminated, returning to the desired capacity.

## Outcome

This Proof of Concept (PoC) aimed to implement Auto Scaling in AWS to dynamically manage EC2 instances based on workload demand, ensuring efficient resource utilization and cost-effectiveness.

Here's the outcome of the PoC:

1. Launch Template and Auto Scaling Group Setup: Successfully created a launch template and configured an Auto Scaling Group with scaling policies to dynamically manage EC2 instances based on workload.

2. Dynamic Scaling and Monitoring: Implemented scaling policies triggered by CPU utilization and verified automatic scaling actions using the Auto Scaling Group's Activity History.

3. Cost Awareness: Highlighted potential costs of running additional instances beyond the AWS Free Tier during testing and ensured resource usage was optimized.