

## **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: SAMRAJ K

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.

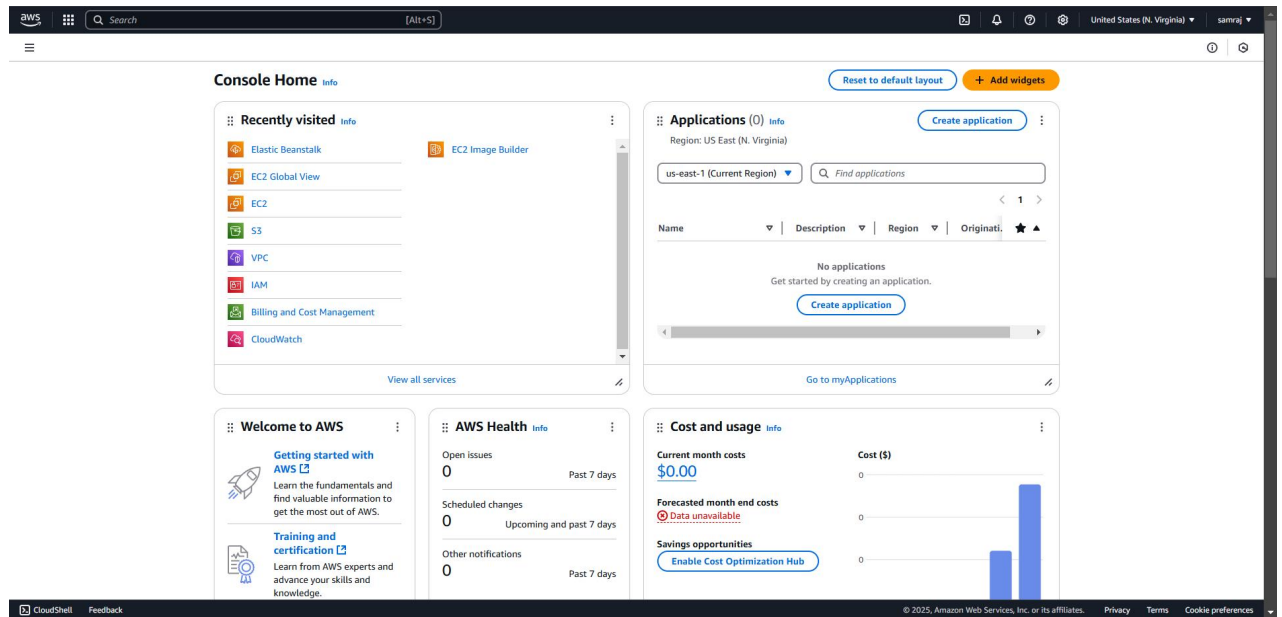
# Importance

- 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

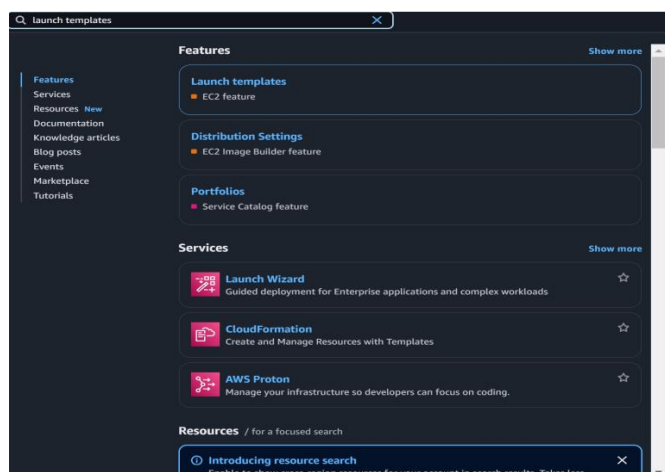
## Step 1:

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



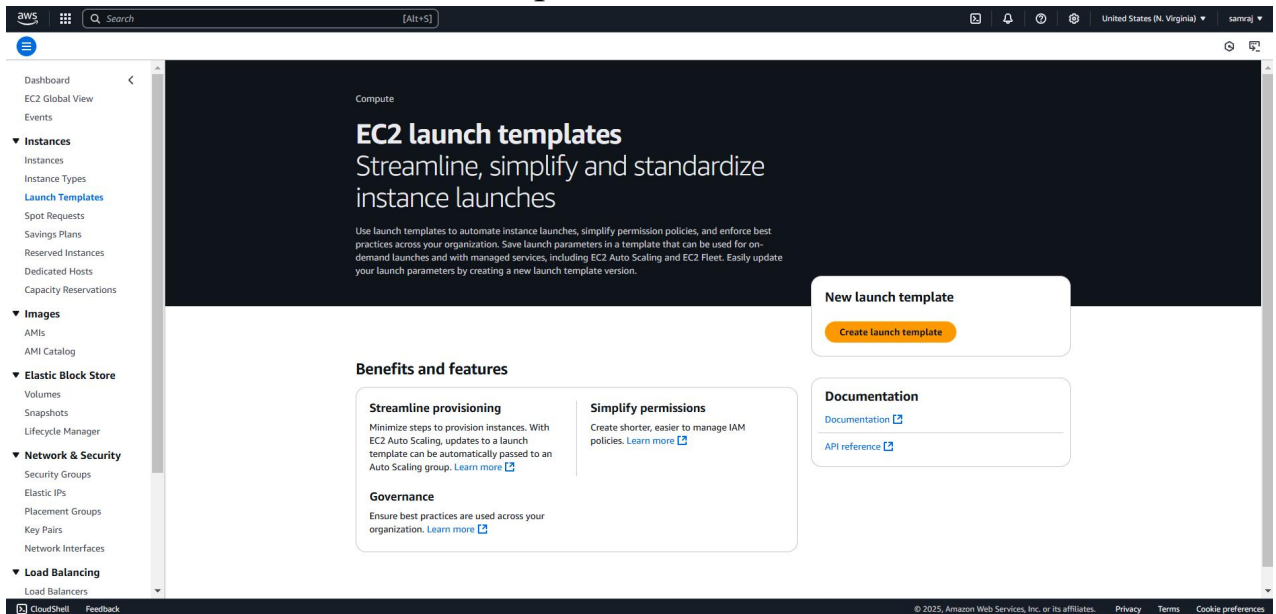
## Step 2:

Search for Launch Templates.



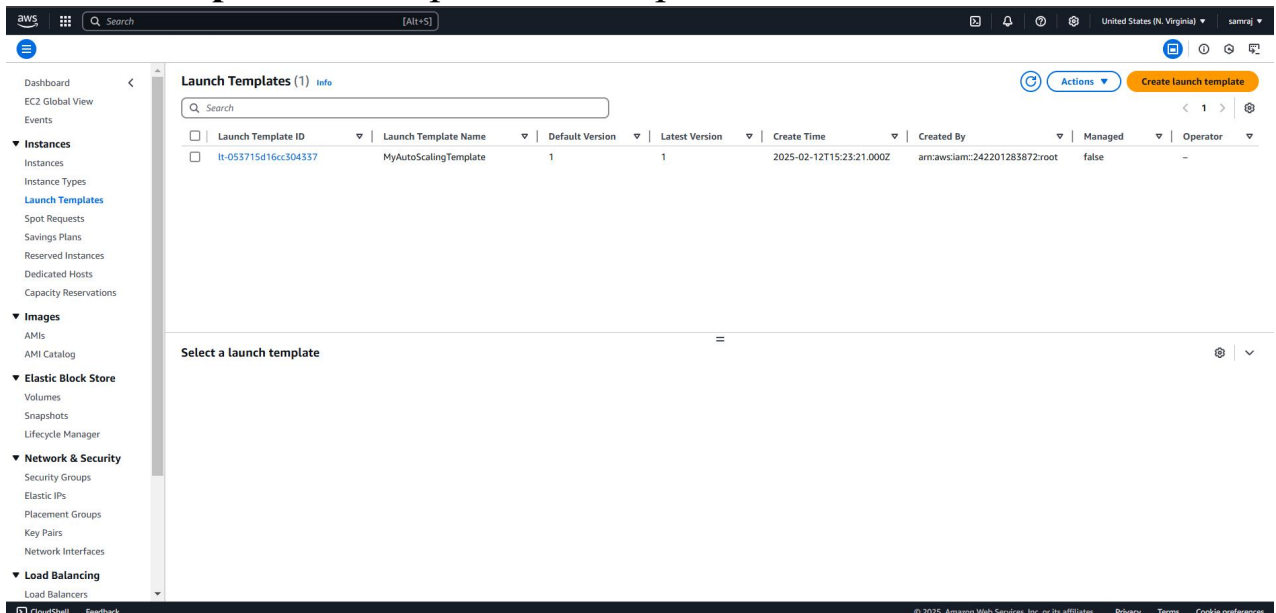
## Step 3:

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**.

The screenshot shows the AWS Management Console interface for the 'Create Auto Scaling group' page. The left sidebar contains navigation links for various AWS services, with 'Auto Scaling' selected. The main content area features a header with the title 'Amazon EC2 Auto Scaling helps maintain the availability of your applications' and a 'Create Auto Scaling group' button. Below the header, there is a 'How it works' section with a diagram illustrating the scaling process: a central box labeled 'Auto Scaling group' contains four smaller boxes representing EC2 instances. Brackets below the instances indicate 'Minimum size', 'Desired capacity', and 'Maximum size'. A 'Scale out as needed' label points to the right side of the group. To the right of the diagram, there are sections for 'Pricing' and 'Getting started' with links to learn more.

## Step 6:

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

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

The screenshot shows the 'Choose launch template' step in the 'Create Auto Scaling group' wizard. The left sidebar displays a progress bar with steps 1 through 7, with step 1 'Choose launch template' selected. The main content area has a title 'Choose launch template' and a description: 'Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.' Below this, there are two main sections: 'Name' and 'Launch template'. The 'Name' section has a text input field for 'Auto Scaling group name' with the value 'MyAutoScalingGroup'. The 'Launch template' section has a dropdown menu for 'Launch template' with the value 'MyAutoScalingTemplate'. Below the dropdown, there are fields for 'Version' (Default (1)) and 'Description'. At the bottom, there is a table with details for the selected launch template:

Launch template	Instance type
MyAutoScalingTemplate lt-053715d16cc304337	t2.micro
Security groups	Request Spot Instances
sg-0200a4c689a5728a5	No

## 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).

The screenshot shows the AWS Management Console interface for creating an Auto Scaling group. The left sidebar indicates the current step is Step 7: Review. The main content area is titled 'Create Auto Scaling group' and shows the 'Instance type' as t2.micro. Under the 'Network' section, the 'VPC' is set to vpc-01cf88a0aaf6bb2a1. Under 'Availability Zones and subnets', two subnets are selected: us-east-1a and us-east-1b. The 'Availability Zone distribution' is set to 'Balanced best effort'. The 'Next' button is highlighted.

## Step 8:

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

The screenshot shows the AWS Management Console interface for creating an Auto Scaling group, Step 8: Integrate with other services - optional. The left sidebar indicates the current step is Step 8: Integrate with other services - optional. The main content area shows options for 'Load balancing' (No load balancer), 'VPC Lattice integration options' (No VPC Lattice service), and 'Application Recovery Controller (ARC) zonal shift' (Enable zonal shift). The 'Next' button is highlighted.

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

### Configure group size and scaling - optional [info](#)

Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

**Group size [info](#)**  
Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

**Desired capacity type**  
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups 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 to meet changes in demand.

**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**  
1  
Equal or greater than desired capacity

**Automatic scaling - optional**  
**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

☐ Target tracking scaling policy  
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired

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

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

### Add notifications - optional [info](#)

Send notifications to SNS topics whenever Amazon EC2 Auto Scaling launches or terminates the EC2 instances in your Auto Scaling group.

[Add notification](#)

Cancel [Skip to review](#) [Previous](#) [Next](#)

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

## Step 9:

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



aws

Search

[Alt+S]

United States (N. Virginia)

samraj

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

MyAutoScalingTemplate

Version

Default

Description

lt-053715d16cc304337

Step 2: Choose instance launch options

Edit

Network

VPC

vpc-01cf88a0aaf6bb2a1

Availability Zones and subnets

Availability Zone	Subnet	Subnet CIDR range
us-east-1a	subnet-083c1616fc9f5b13d	172.31.16.0/20
us-east-1b	subnet-0272ce91f62a7e990	172.31.32.0/20

Availability Zone distribution

Balanced best effort

Instance type requirements

CloudShell

Feedback

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

aws

Search

[Alt+S]

United States (N. Virginia)

samraj

EC2

Auto Scaling groups

Auto Scaling groups (1)

Info

Launch configurations

Launch templates

Actions

Create Auto Scaling group

Search your Auto Scaling groups

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
MyAutoScalingGroup	MyAutoScalingTemplate   Version Default	0	Updating capacity...	1	1	1	us-east-1a, us-east-1b

0 Auto Scaling groups selected

CloudShell

Feedback

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

aws

Search

[Alt+S]

United States (N. Virginia)

samraj

EC2

Auto Scaling groups

MyAutoScalingGroup

MyAutoScalingGroup

Capacity overview

Edit

arn:aws:autoscaling:us-east-1:242201283872:autoScalingGroup:dcba5c3d-6bc0-4f06-9b2d-39c5c7f3c1cf:autoScalingGroupName/MyAutoScalingGroup

Desired capacity

1

Scaling limits (Min - Max)

1 - 1

Desired capacity type

Units (number of instances)

Status

Updating capacity

Date created

Wed Feb 12 2025 20:56:41 GMT+0530 (India Standard Time)

Details

Integrations - new

Automatic scaling

Instance management

Instance refresh

Activity

Monitoring

Launch template

Edit

Launch template

lt-053715d16cc304337

MyAutoScalingTemplate

Version

Default

Description

-

View details in the launch template console

AMI ID

ami-085ad6ae776d8f09c

Security groups

-

Storage (volumes)

-

Instance type

t2.micro

Security group IDs

sg-0200a4c689a5728a5

Key pair name

DEV2

Owner

arn:aws:iam::242201283872:root

Create time

Wed Feb 12 2025 20:53:21 GMT+0530 (India Standard Time)

Request Spot Instances

No

Network

Edit

Availability Zones

us-east-1a, us-east-1b

Subnet ID

subnet-083c1616fc9f5b13d, subnet-0272ce91f62a7e990

Availability Zone distribution

Balanced best effort

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.