



## **Placement Empowerment Program**

### ***Cloud Computing and DevOps Centre***

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

Name: Jayadasan S

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.

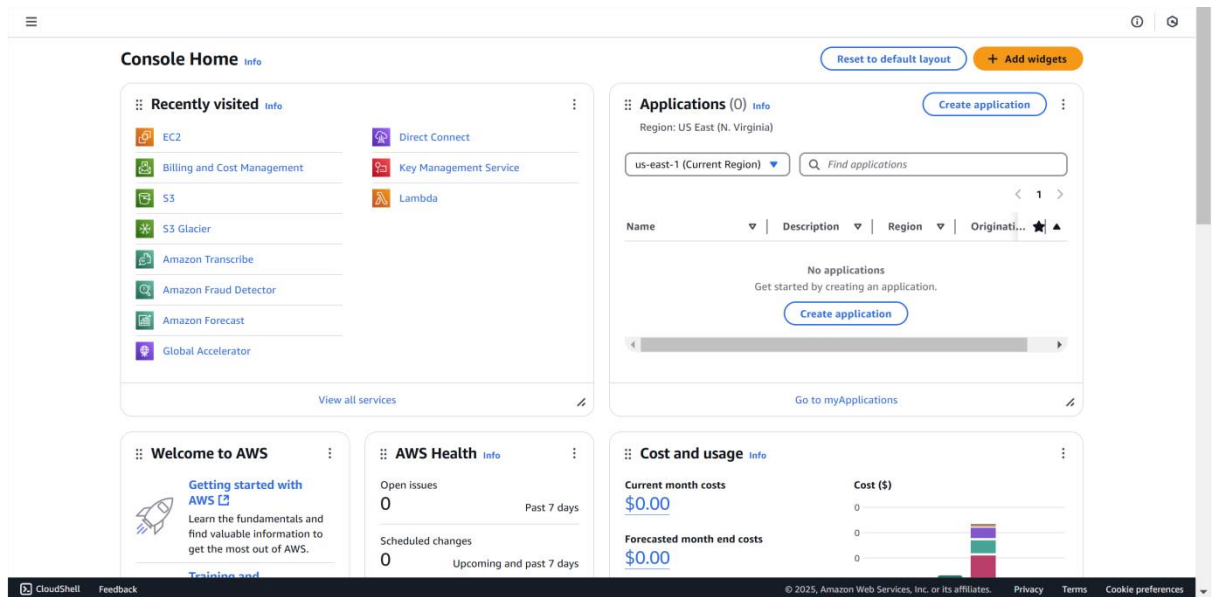
## Importance

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.

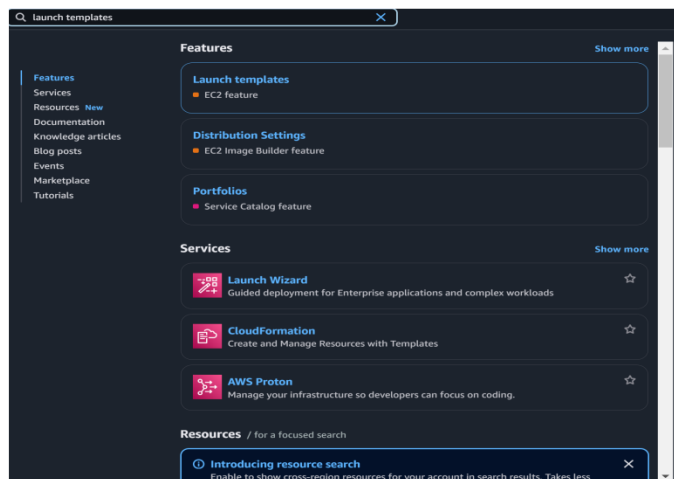
## Step-by-Step Overview Step

1:

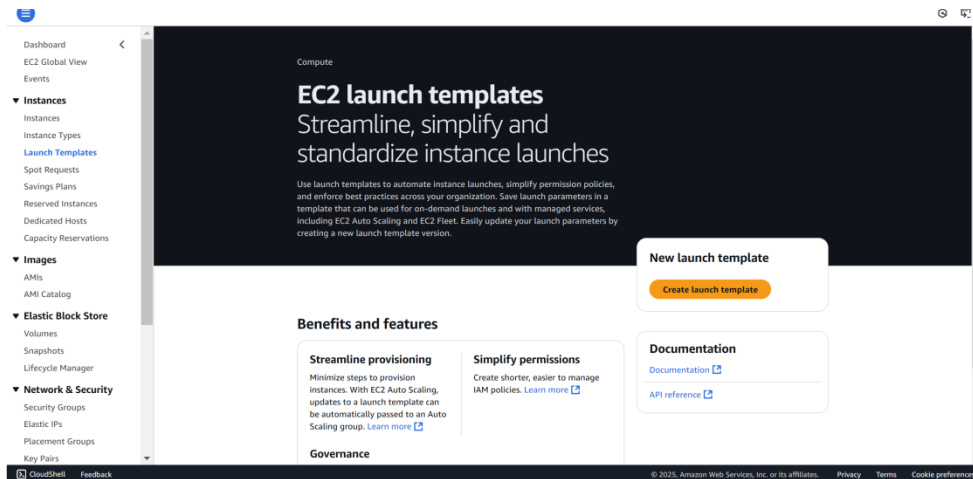
1. Go to [AWS Management Console](#).
2. Enter your username and password to log in.



Step 2:  
Search for Launch Templates.

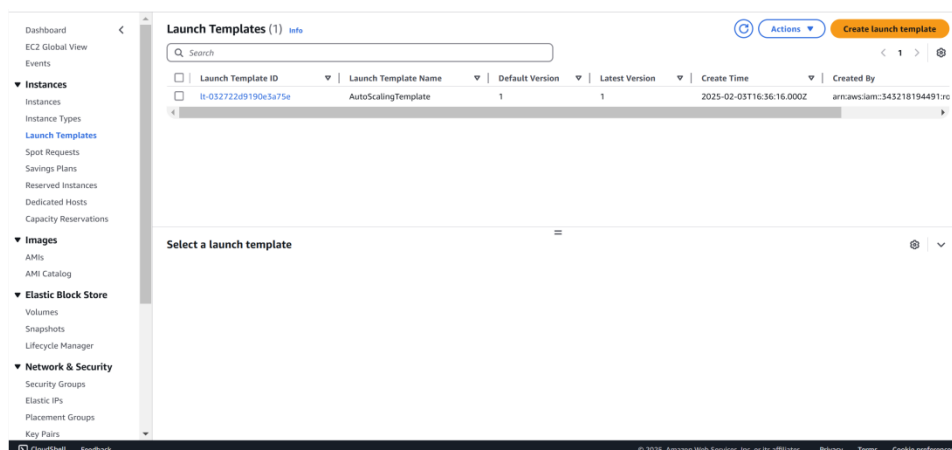


Step 3:  
Click on the Create launch template.



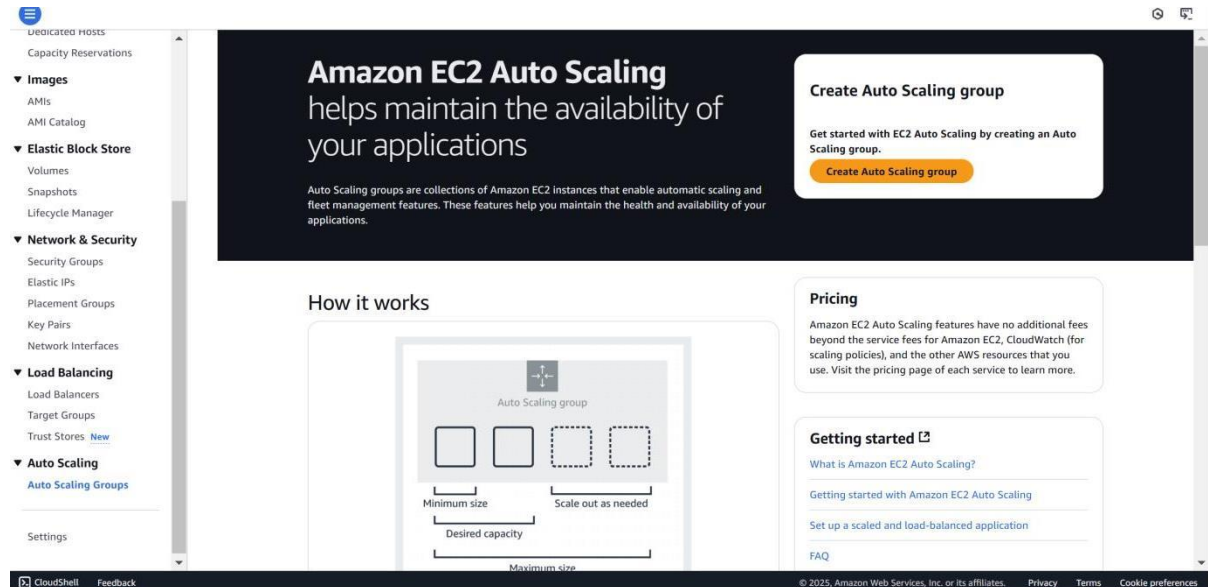
Step 4:

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:

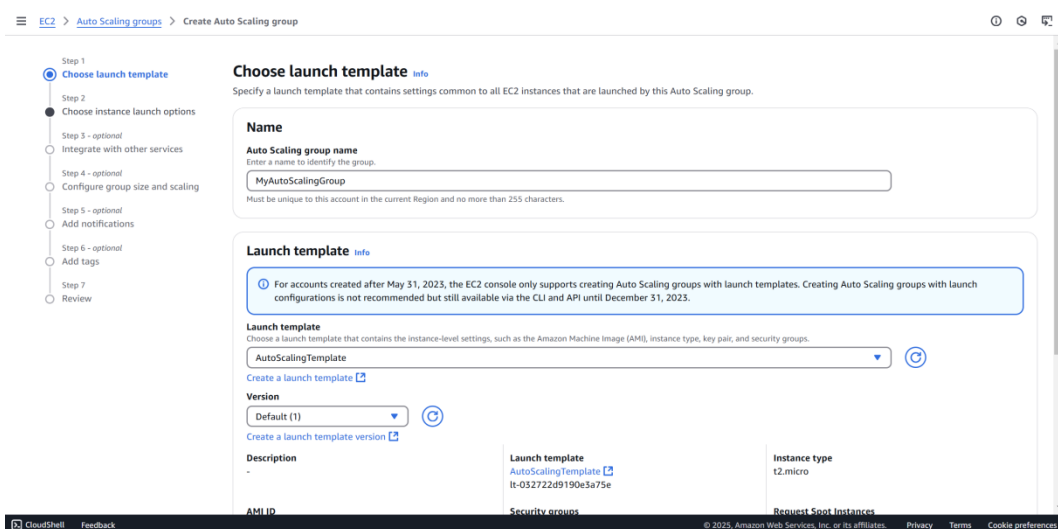
Go to the **EC2 Dashboard** . On the left sidebar, click on **Auto Scaling Groups**. Click on **Create an Auto Scaling group**.



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).

The screenshot shows the 'Create Auto Scaling group' wizard in the AWS Management Console, specifically Step 2: 'Choose instance launch options'. The left sidebar shows the progress of the steps: Step 2 is active, followed by Step 3 (optional), Step 4 (optional), Step 5 (optional), Step 6 (optional), Step 7 (optional), and Step 8 (Review). The main content area is titled 'Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.' It contains two main sections: 'Instance type requirements' and 'Network'. The 'Instance type requirements' section has a table with columns for 'Launch template', 'Version', and 'Description'. The 'Launch template' is 'AutoScalingTemplate lt-032722d9190e3a75e' and the 'Instance type' is 't2.micro'. The 'Network' section has a 'VPC' dropdown menu set to 'vpc-0f36f0944c12862e5' and an 'Availability Zones and subnets' dropdown menu set to 'us-east-1a | subnet-0f01daeb9f48d5fc4'. There are links to 'Override launch template', 'Create a VPC', and 'Create a subnet'.

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 2: Choose instance launch options

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

**Instance type requirements** [Info](#) [Override launch template](#)

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template	Version	Description
<a href="#">AutoScalingTemplate lt-032722d9190e3a75e</a>	Default	-

**Instance type**  
t2.micro

**Network** [Info](#)

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

**VPC**  
Choose the VPC that defines the virtual network for your Auto Scaling group.

[vpc-0f36f0944c12862e5](#)  
172.31.0.0/16 Default

[Create a VPC](#)

**Availability Zones and subnets**  
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

[Select Availability Zones and subnets](#)

[us-east-1a | subnet-0f01daeb9f48d5fc4](#)  
172.31.80.0/20 Default

[Create a subnet](#)

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

Step 8:

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



EC2 > Auto Scaling groups > Create Auto Scaling group

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

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**.

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1: Choose launch template

Group details

Auto Scaling group name  
MyAutoScalingGroup

Launch template

Launch template	Version	Description
AutoScalingTemplate lt-032722d9190e3a75e	Default	

Step 2: Choose instance launch options

Network

VPC  
vpc-f036f0944c12862e5

Availability Zones and subnets

Availability Zone	Subnet	Subnet CIDR range
us-east-1a	subnet-0f01dae9b948d5fcd	172.31.80.0/20

Availability Zone distribution  
Balanced best effort

EC2 > Auto Scaling groups

Auto Scaling groups (1)

Search your Auto Scaling groups

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

0 Auto Scaling groups selected

EC2 > Auto Scaling groups > MyAutoScalingGroup

MyAutoScalingGroup

MyAutoScalingGroup Capacity overview

arn:aws:autoscaling:us-east-1:343218194491:autoScalingGroup:a52c8890-07bc-41c8-a7a8-2d6da899fa7d:autoScalingGroupName/MyAutoScalingGroup

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

Date created  
Mon Feb 03 2025 22:12:12 GMT+0530 (India Standard Time)

Details | Integrations - new | Automatic scaling | Instance management | Instance refresh | Activity | Monitoring

Launch template

Launch template lt-032722d9190e3a75e AutoScalingTemplate	AMI ID ami-0c614dee691cbbf37	Instance type t2.micro	Owner arn:aws:iam::343218194491:root
Version Default	Security groups -	Security group IDs -	Create time Mon Feb 03 2025 22:06:16 GMT+0530 (India Standard Time)
Description -	Storage (volumes) -	Key pair name newkey	Request Spot Instances No

View details in the launch template console

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.