



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: Sylashri Rajendran

Department: IT

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 costefficiency 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

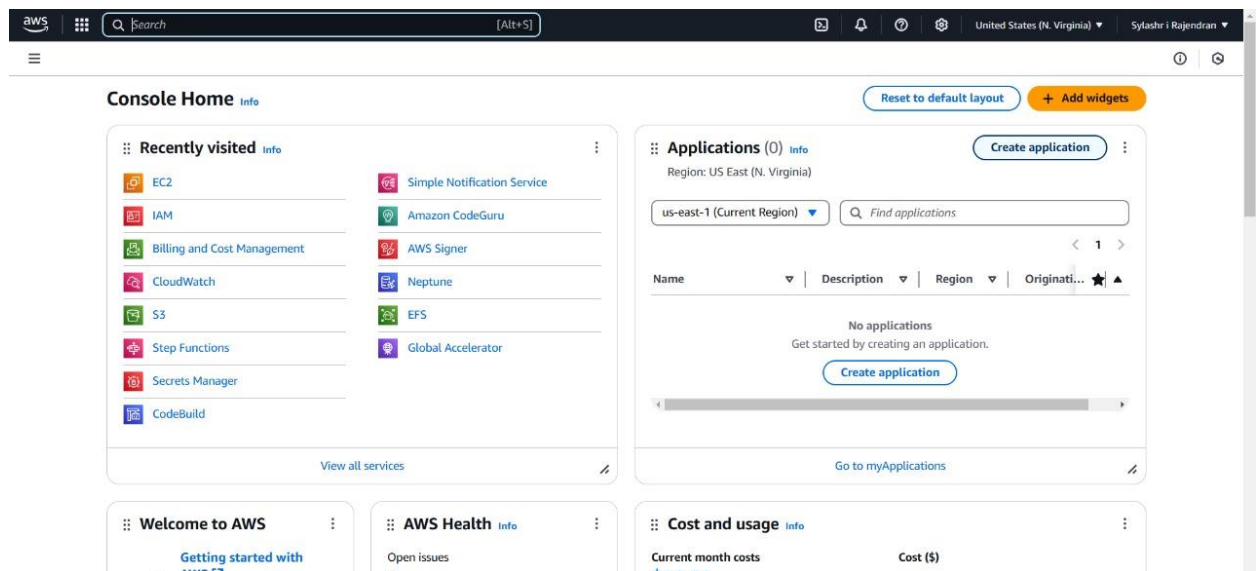
- 1. Improved Application Availability:** Auto Scaling ensures that applications remain available even during traffic spikes by automatically adding more VMs to meet demand.
- 2. Cost Optimization:** It dynamically reduces the number of VMs during low traffic periods, minimizing unnecessary costs.
- 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

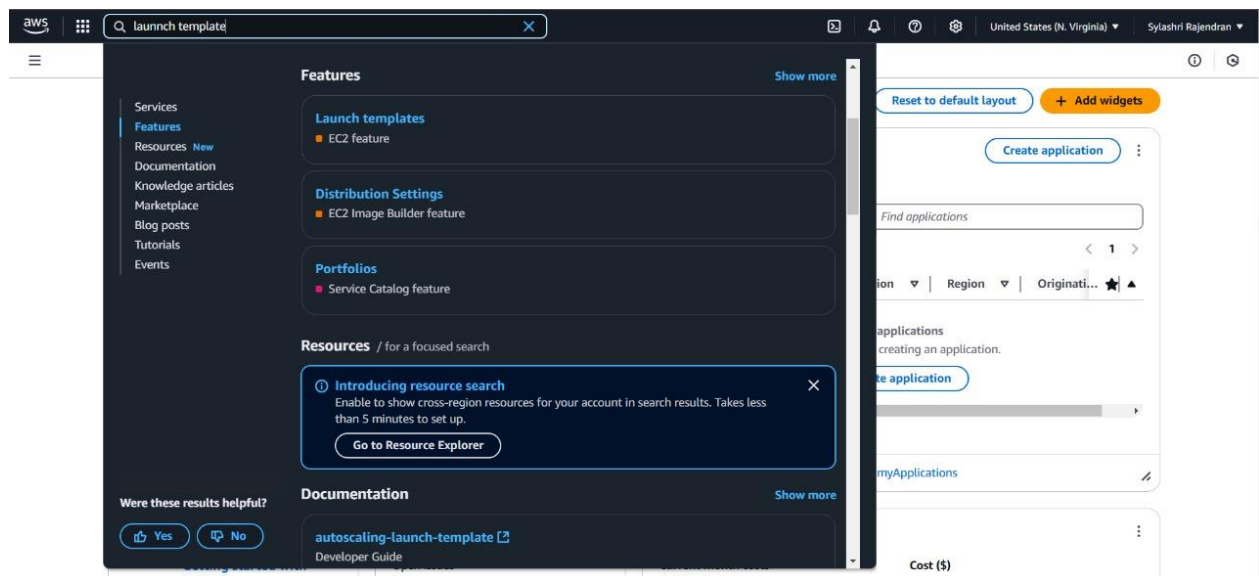
Step 1:

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



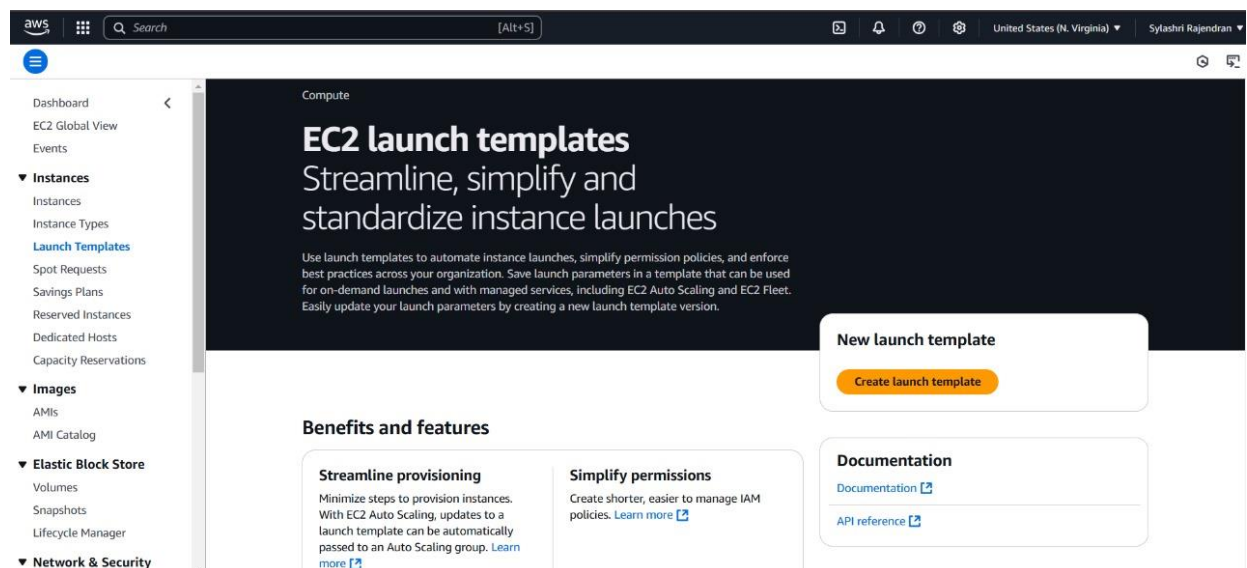
Step 2:

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

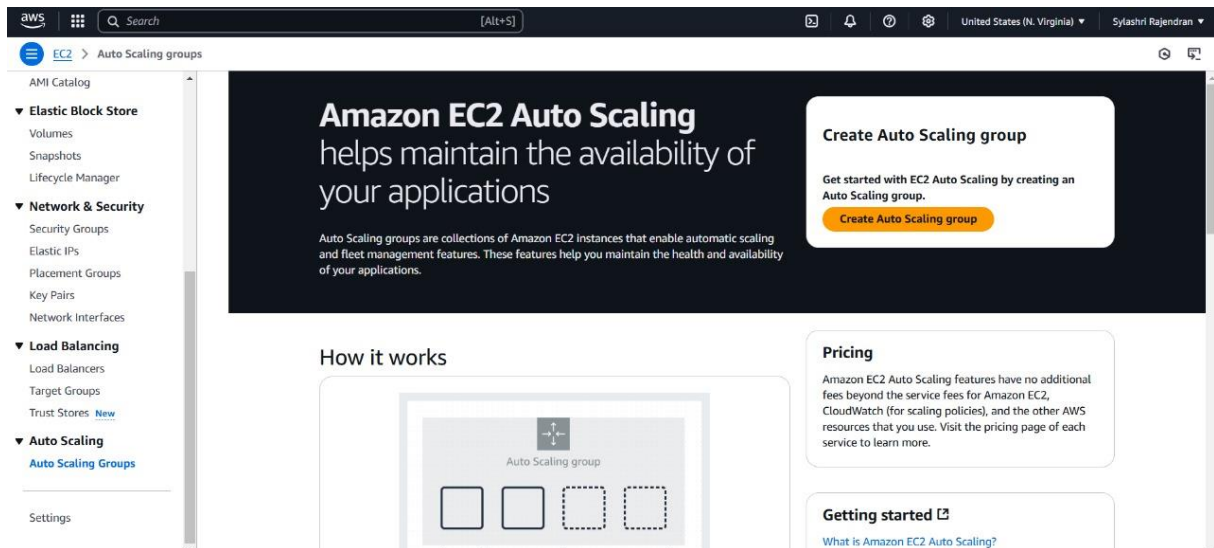
SSH access, and configure a **security group** that allows HTTP (port 80) and SSH (port 22). Once all details are filled out, click **Create launch template** to complete the setup.

The screenshot shows the AWS Management Console interface for the 'Launch Templates' section. The top navigation bar includes the AWS logo, a search bar, and the current region 'United States (N. Virginia)'. The left sidebar lists various AWS services, with 'Instances' and 'Launch Templates' highlighted. The main content area shows a table of launch templates. One template is listed with the ID 'lt-0cf39a55fd06bcab7' and the name 'AutoScalingTemplate'. Below the table, there is a section titled 'Select a launch template'.

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
lt-0cf39a55fd06bcab7	AutoScalingTemplate	1	1	2025-02-06T13:21:09.000Z	arn:aws:iam::842676

Step 5:

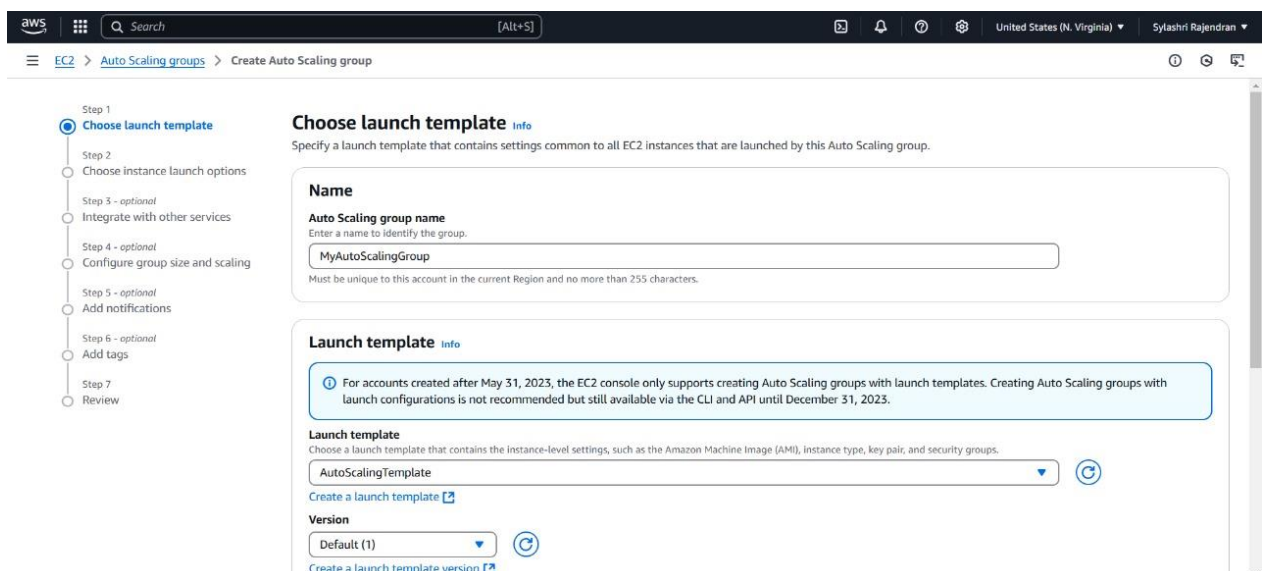
Go to the **EC2 Dashboard**. On the left sidebar, click on **Auto Scaling Groups**. Click on **Create an Auto Scaling group**.



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' page in the AWS Management Console, specifically the 'Network' section. The left sidebar shows the progress of the setup, with 'Step 7 - optional' (Network) highlighted. The main content area includes a 'Launch template' section with 'AutoScalingTemplate' and 't2.micro' instance type. Below this, the 'Network' section explains that multiple Availability Zones can be used. It features a 'VPC' dropdown menu with 'vpc-01d9aceb426983a46' selected, and an 'Availability Zones and subnets' dropdown menu with 'us-east-1a | subnet-0253186867ae22eaa' selected. Both dropdowns have a 'Create a VPC' or 'Create a subnet' link next to them.

Step 8:

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

The screenshot shows the 'Create Auto Scaling group' page in the AWS Management Console, specifically the 'Integrate with other services' section. The left sidebar shows the progress of the setup, with 'Step 8 - optional' (Integrate with other services) highlighted. The main content area includes a 'Load balancing' section with three radio button options: 'No load balancer' (selected), 'Attach to an existing load balancer', and 'Attach to a new load balancer'. Below this, the 'VPC Lattice integration options' section has two radio button options: 'No VPC Lattice service' (selected) and 'Attach to VPC Lattice service'. At the bottom, there is a section for 'Application Recovery Controller (ARC) zonal shift - new'.

aws

Search

[Alt+S]

United States (N. Virginia)

Sylashri Rajendran

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

aws

Search

[Alt+S]

United States (N. Virginia)

Sylashri Rajendran

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

aws

Search

[Alt+S]

United States (N. Virginia)

Sylashri Rajendran

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 tags - optional

Info

Add tags to help you search, filter, and track your Auto Scaling group across AWS. You can also choose to automatically add these tags to instances when they are launched.

You can optionally choose to add tags to instances (and their attached EBS volumes) by specifying tags in your launch template. We recommend caution, however, because the tag values for instances from your launch template will be overridden if there are any duplicate keys specified for the Auto Scaling group.

Tags (0)

Add tag

50 remaining

Cancel

Previous

Next

Step 9:

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

The screenshot shows the 'Review' step of the 'Create Auto Scaling group' wizard in the AWS Management Console. On the left, a progress bar lists seven steps: 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, and Step 7: Review (which is the current step). The main content area is divided into two sections: 'Step 1: Choose launch template' and 'Step 2: Choose instance launch options'. 'Step 1' includes 'Group details' with the 'Auto Scaling group name' set to 'MyAutoScalingGroup' and a 'Launch template' section showing 'AutoScalingTemplate' with version 'Default' and description 'AutoScalingTemplate'. 'Step 2' includes a 'Network' section showing 'VPC' as 'vpc-Def3f992780c8424' and 'Availability Zones and subnets' with 'us-east-1a' and 'subnet-062c941e7c4ad9874' having a 'Subnet CIDR range' of '172.31.16.0/20'. Each step has an 'Edit' button.

The screenshot shows the 'Auto Scaling groups' page in the AWS Management Console. At the top, there are buttons for 'Launch configurations', 'Launch templates', 'Actions', and a prominent orange 'Create Auto Scaling group' button. Below these is a search bar labeled 'Search your Auto Scaling groups'. The main part of the page is a table listing the Auto Scaling groups. The table has columns for Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and Availability Zones. One group is listed: 'MyAutoScalingGroup' with launch template 'AutoScalingTemplate', version 'Default', 0 instances, status 'Updating capacity...', desired capacity 1, min 1, max 1, and availability zone 'us-east-1a'.

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

MyAutoScalingGroup

MyAutoScalingGroup Capacity overview

arn:aws:autoscaling:us-east-1:842676011451:autoScalingGroup:53d4fcec-cc27-452f-8787-5169f010897e:autoScalingGroupName/MyAutoScalingGroup

Desired capacity

1

Scaling limits (Min - Max)

1 - 1

Desired capacity type

Units (number of instances)

Status

-

Date created

Thu Feb 06 2025 19:37:39 GMT+0530 (India Standard Time)

Edit

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

Launch template

Launch template

lt-Ocf39a55fd06bca7

AutoScalingTemplate

Version

Default

Description

-

AMI ID

ami-085ad6ae776d8f09c

Security groups

-

Storage (volumes)

-

Instance type

t2.micro

Security group IDs

sg-0cc8986342a16a2e6

Key pair name

fun

Owner

arn:aws:iam::842676011451:root

Create time

Thu Feb 06 2025 18:51:09 GMT+0530 (India Standard Time)

Request Spot Instances

No

Edit

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.