



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: Samuel James Billy Graham

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 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.

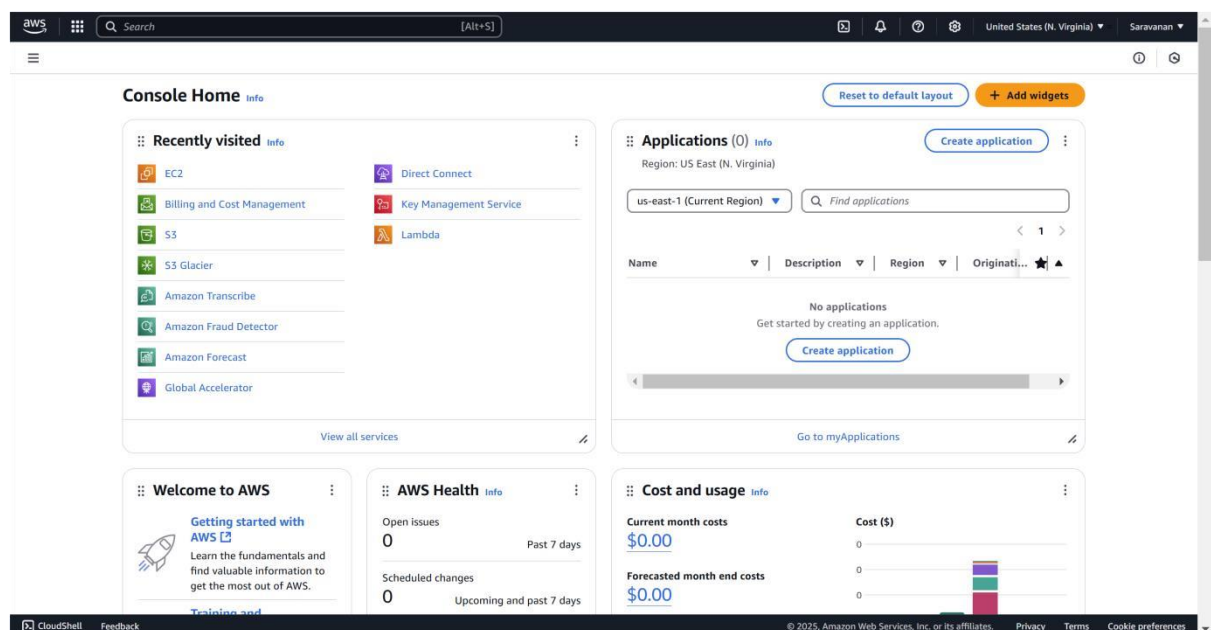
4. Resilience to Failures: Auto Scaling can replace unhealthy instances automatically, ensuring consistent application performance.

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

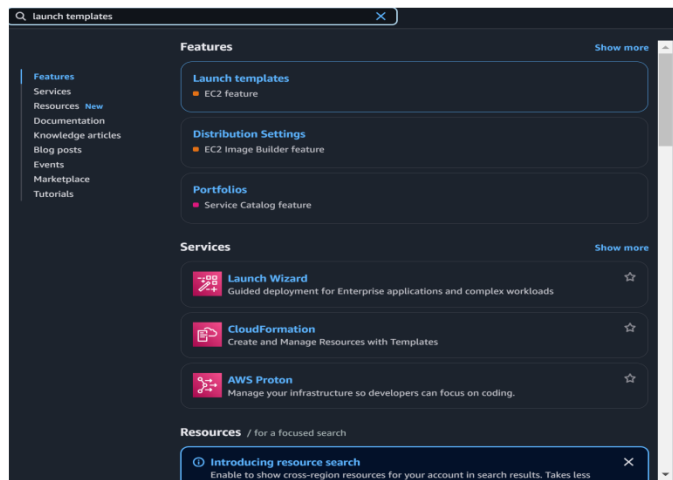
Step 1:

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



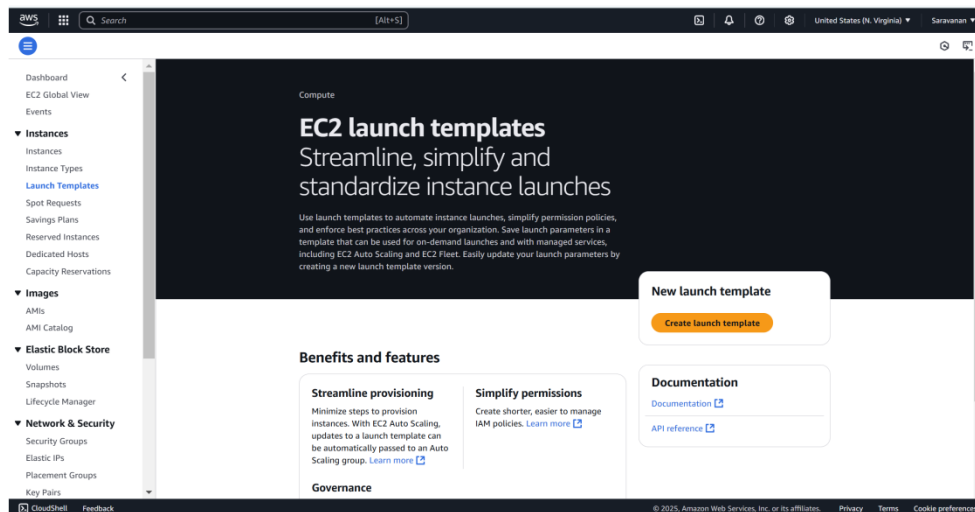
Step 2:

Search for Launch Templates.



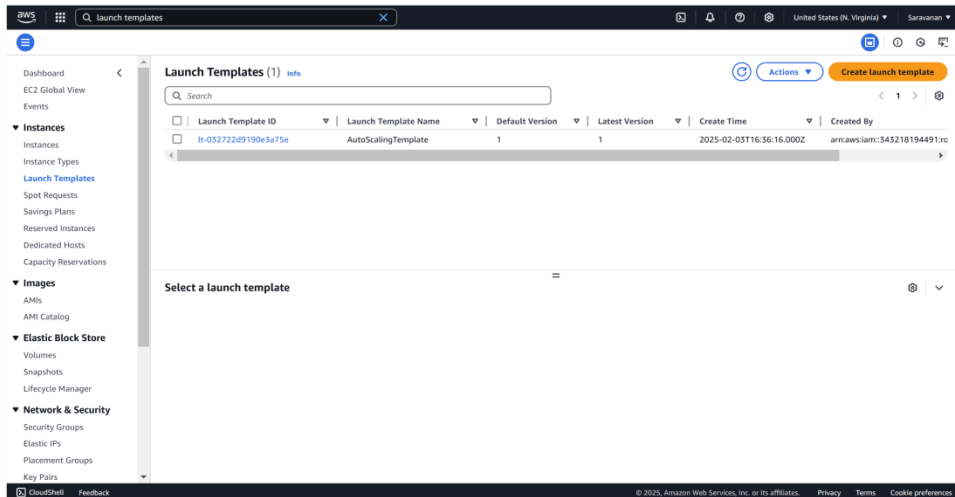
Step 3:

Click on the Create launch template.



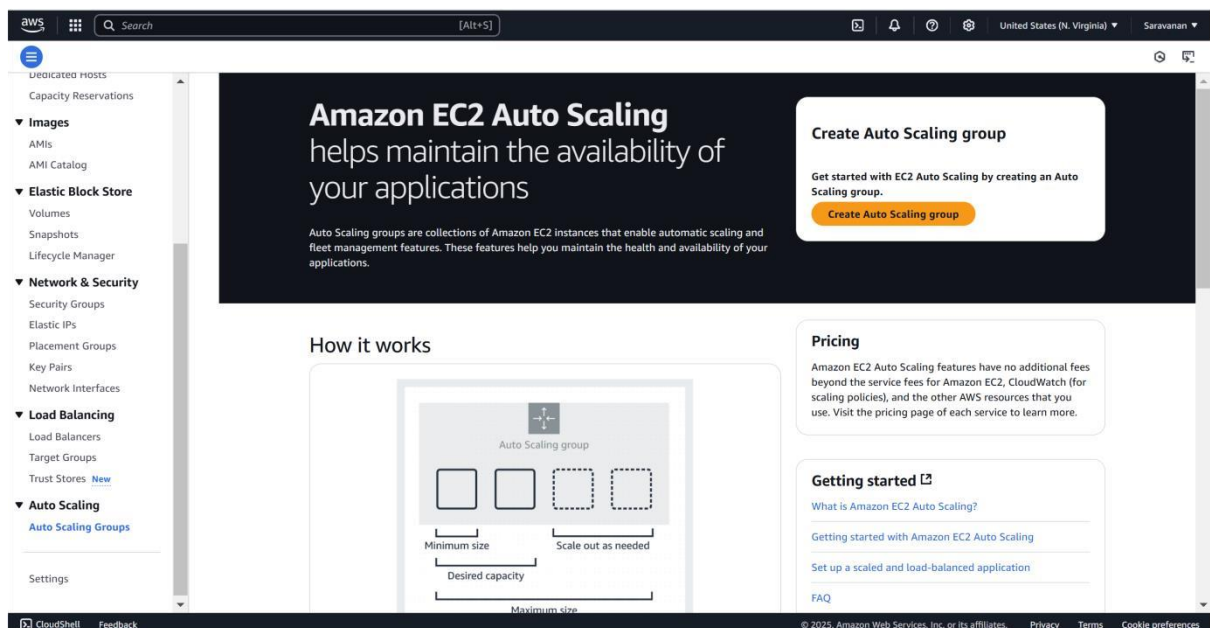
Step 4:

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.



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 1: Choose launch template

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

Name

Auto Scaling group name
Enter a name to identify the group.
MyAutoScalingGroup
Must be unique to this account in the current Region and no more than 255 characters.

Launch template

For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

Launch template
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.
AutoScalingTemplate
Create a launch template

Version
Default (1)
Create a launch template version

Description
-

Launch template
AutoScalingTemplate
lt-032722d9190e3a75e

Instance type
t2.micro

AMI ID

Security groups

Resourced Spot Instances

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 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

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
AutoScalingTemplate
lt-032722d9190e3a75e

Version
Default

Description
-

Instance type
t2.micro

Network

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

Step 8:

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

 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

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

Step 9:

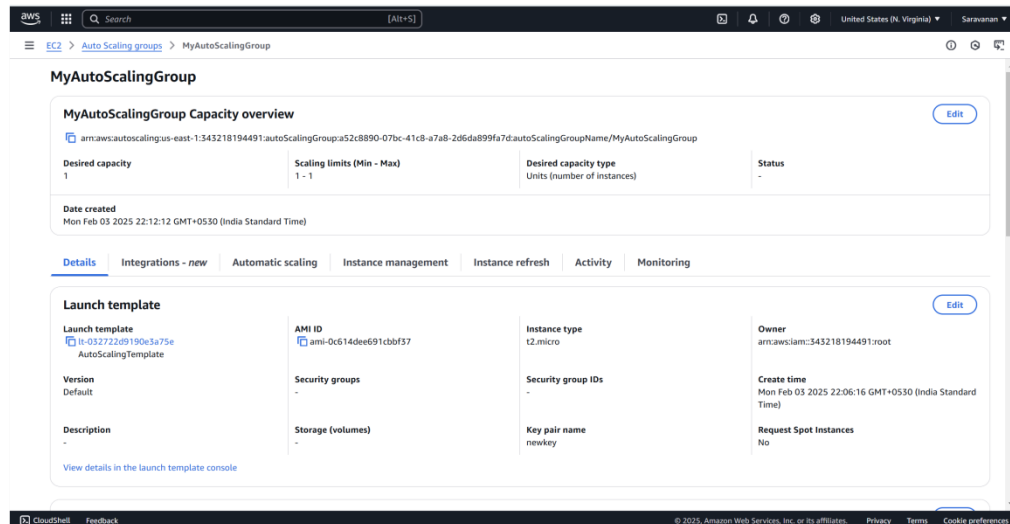
Review all the settings you've configured. Once satisfied, click **Create Auto Scaling Group**.

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: 1. Choose launch template, 2. Choose instance launch options, 3 - 6 (optional), and 7. Review. Step 7 is currently selected. The main content area is divided into two sections: 'Step 1: Choose launch template' and 'Step 2: Choose instance launch options', each with an 'Edit' button. The 'Group details' section shows the 'Auto Scaling group name' as 'MyAutoScalingGroup'. The 'Launch template' section shows 'Launch template' as 'AutoScalingTemplate' and 'Version' as 'Default'. The 'Network' section shows 'VPC' as 'vpc-0f36f0944c12862e5'. The 'Availability Zones and subnets' table lists 'us-east-1a' with subnet 'subnet-0f01d5eb9f48d5fc4' and CIDR range '172.31.80.0/20'. The 'Availability Zone distribution' is set to 'Balanced best effort'.

Availability Zone	Subnet	Subnet CIDR range
us-east-1a	subnet-0f01d5eb9f48d5fc4	172.31.80.0/20

The screenshot shows the 'Auto Scaling groups' page in the AWS Management Console. At the top, there are tabs for 'Launch configurations', 'Launch templates', 'Actions', and a prominent 'Create Auto Scaling group' button. Below the tabs is a search bar and a table of Auto Scaling groups. The table has columns for Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and Availability Zones. One group, 'MyAutoScalingGroup', is listed with launch template 'AutoScalingTemplate', version 'Default', 0 instances, and a status of 'Updating capacity...'. The bottom of the page shows '0 Auto Scaling groups selected'.

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



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.