

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: Shana R S

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 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.

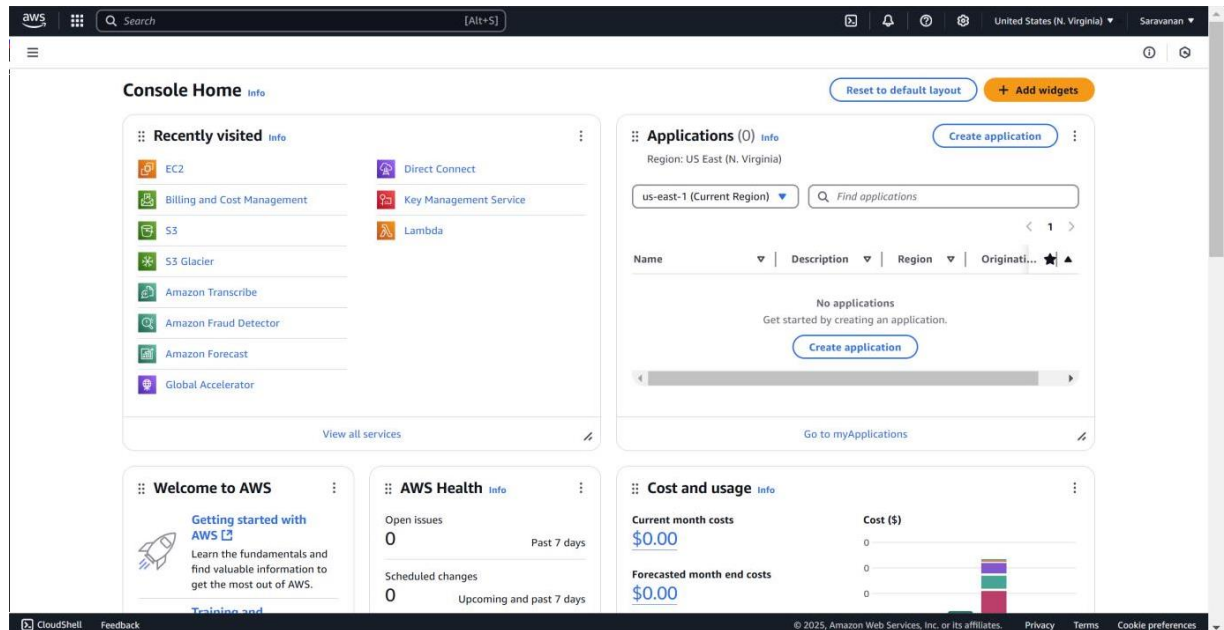
Importance

- 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

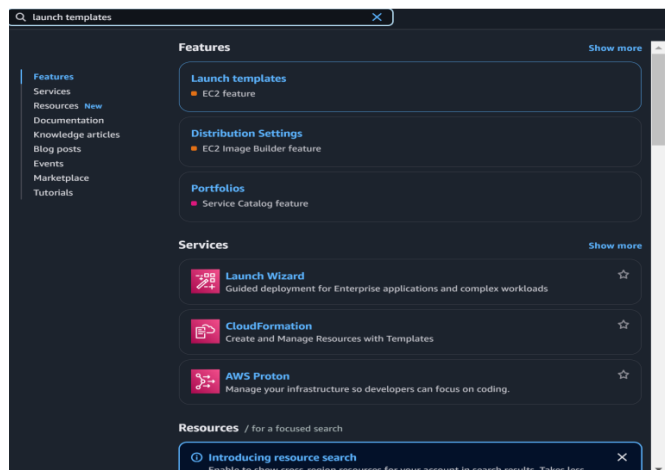
Step 1:

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



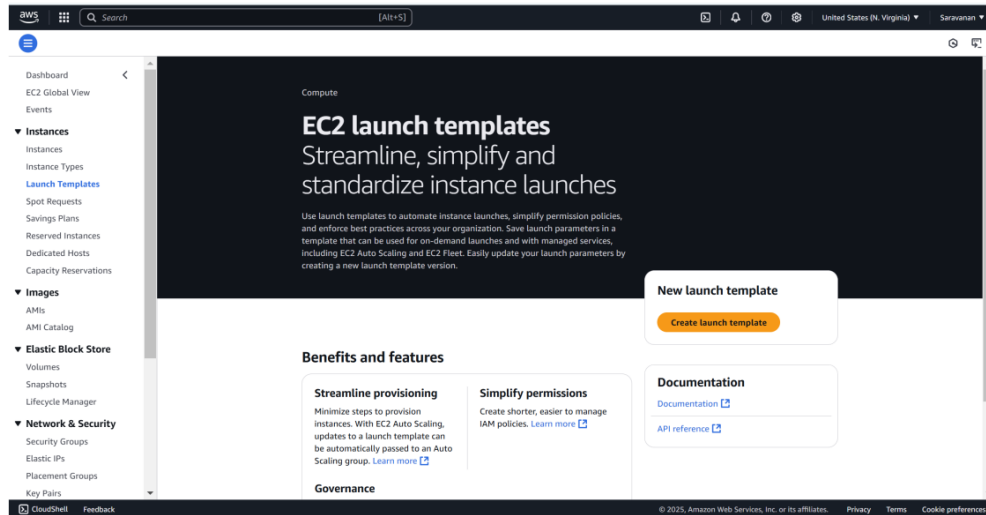
Step 2:

Search for Launch Templates.



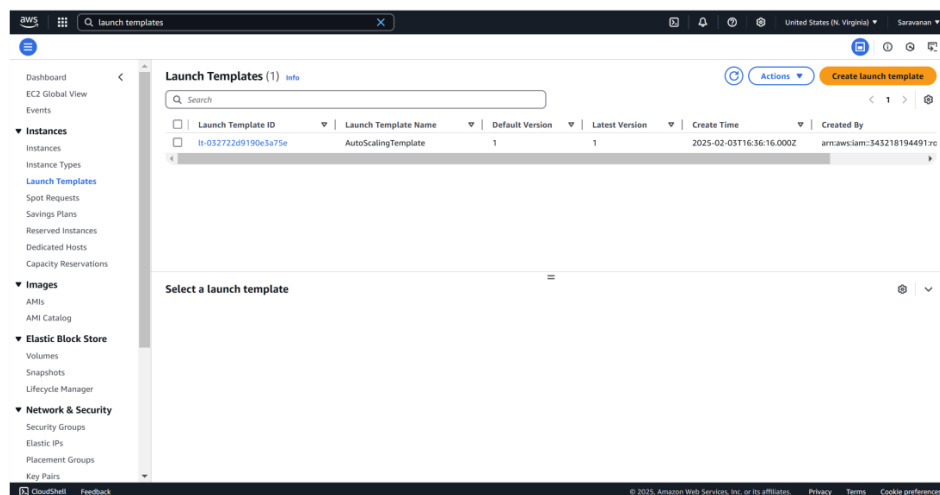
Step 3:

Click on the Create launch template.



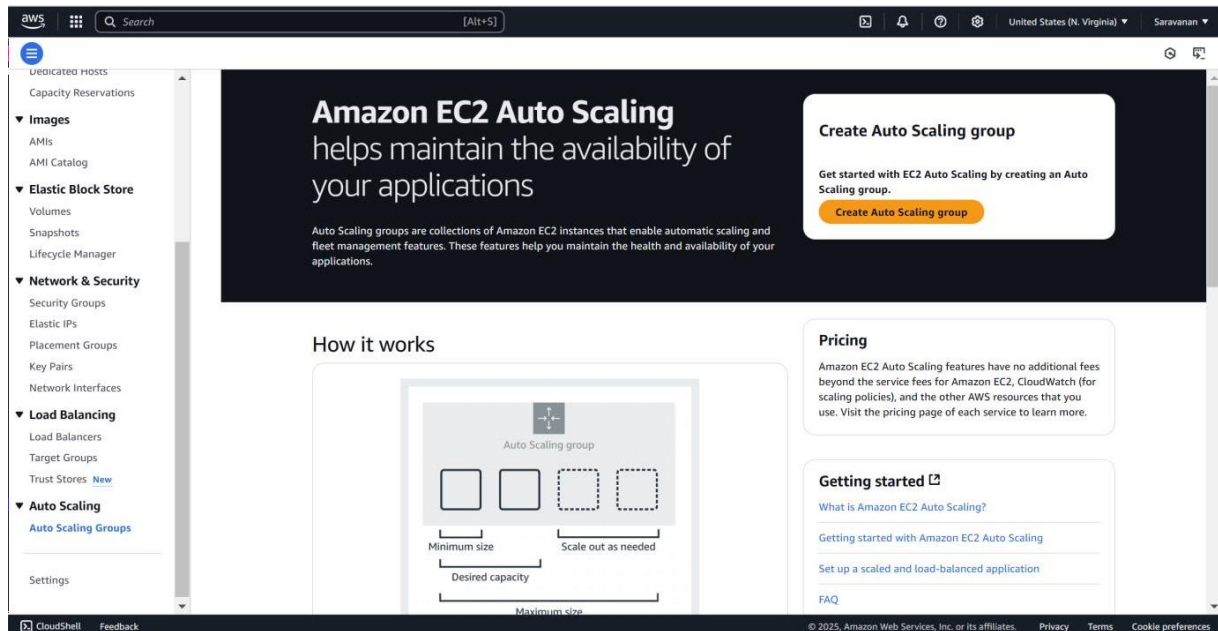
Step 4:

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:

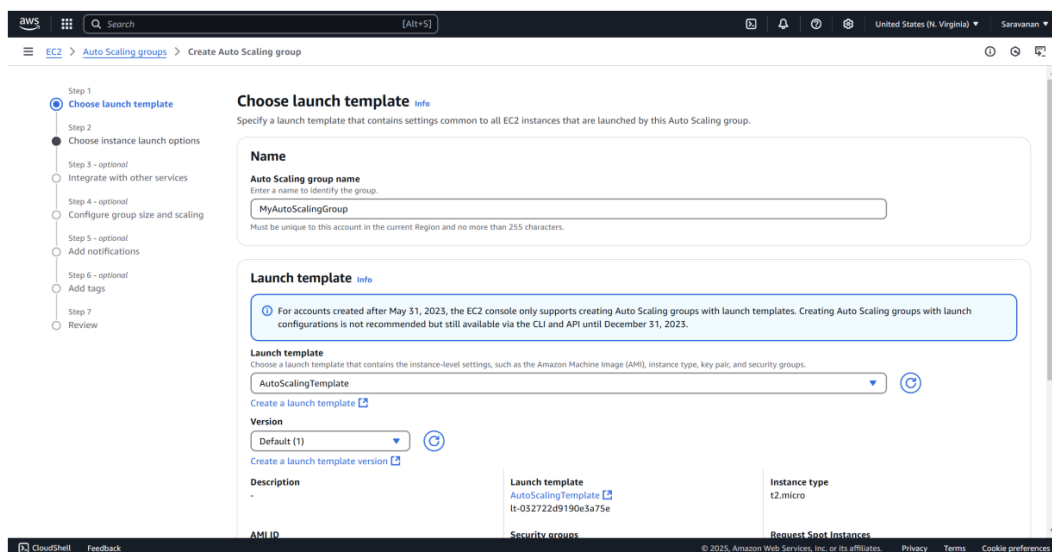
Go to the **EC2 Dashboard**. On the left sidebar, click on **Auto Scaling Groups**. Click on **Create an Auto Scaling group**.



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 7: Network. The left sidebar shows the progress: Step 2 (Choose instance launch options) is active, followed by Step 3 (optional), Step 4 (optional), Step 5 (optional), Step 6 (optional), and Step 7 (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 includes an 'Override launch template' button and a table for 'Instance type requirements' with columns for Launch template, Version, and Description. Below this is the 'Network' section, which includes a 'VPC' dropdown menu (currently showing 'vpc-0f36f0944c12862e5') and an 'Availability Zones and subnets' dropdown menu (currently showing 'us-east-1a | subnet-0f01daeb9f48d5fc4').

Step 8:

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

The screenshot shows the 'Create Auto Scaling group' wizard in the AWS Management Console, specifically Step 8: Integrate with other services. The left sidebar shows the progress: Step 1 (Choose launch template), Step 2 (Choose instance launch options), Step 3 (optional), Step 4 (optional), Step 5 (optional), Step 6 (optional), and Step 7 (Review) are all completed. The main content area is titled 'Integrate with other services - optional' and includes sections for 'Load balancing', 'VPC Lattice integration options', and 'Application Recovery Controller (ARC) zonal shift'. The 'Load balancing' section has three radio button options: 'No load balancer' (selected), 'Attach to an existing load balancer', and 'Attach to a new load balancer'. The 'VPC Lattice integration options' section has two radio button options: 'No VPC Lattice service' (selected) and 'Attach to VPC Lattice service'. The 'Application Recovery Controller (ARC) zonal shift' section has a checkbox for 'Enable zonal shift' which is currently unchecked.

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

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

Group size
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
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.

☒ 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

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

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

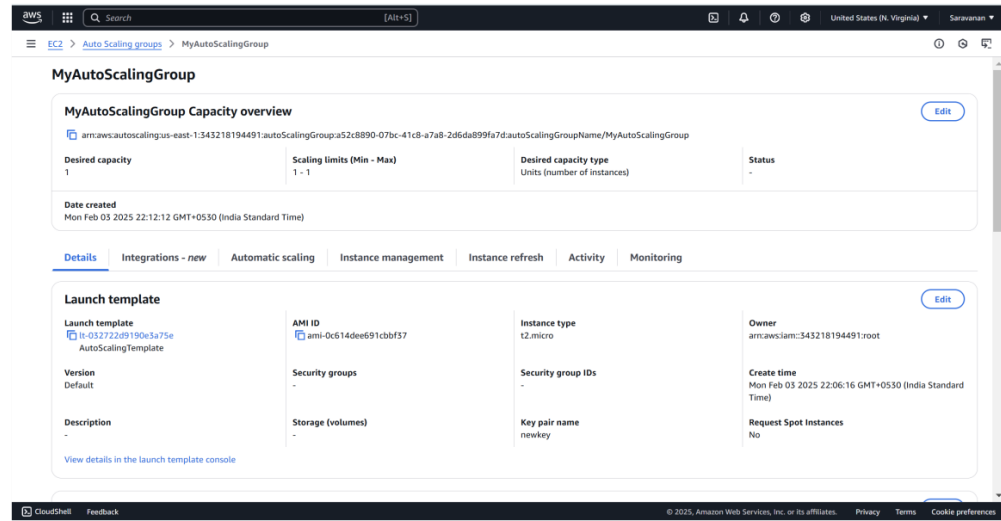
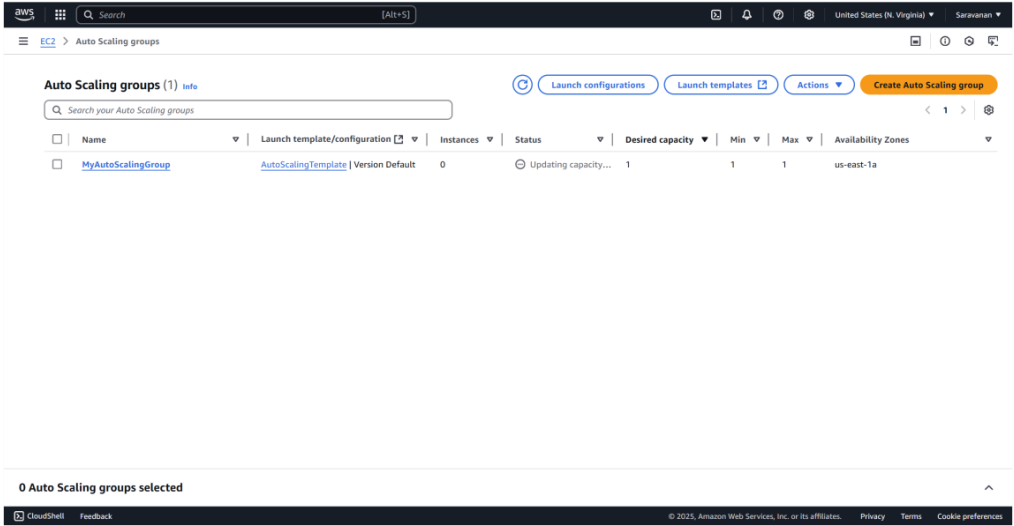
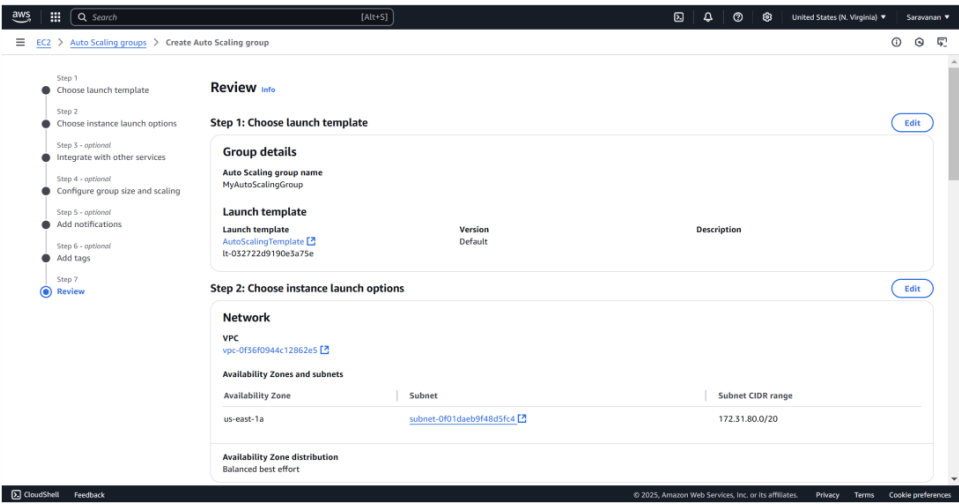
[Add notification](#)

[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**.



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.