**Project :  
Scaling the EC2 Instance Based on Monitored CloudWatch Metrics**

Description

Use Auto Scaling to manage the EC2 instances and use EC2 instance and capture the metrics in the CloudWatch.

**Description:**

Let’s take the case of Hotstar — a platform that provides on-demand video streaming services. The more the users join the streaming service platform, the more the resources in terms of servers (EC2 in AWS) Hotstar needs to invest in. This way, the load is distributed across different servers and leads to jitter-free experience for the customers while watching the videos.   
  
Another example is Amazon Prime Day, where a bevy of customers access the amazon.com site. **Depending on the number of customers loggin**g into the amazon.com site, Amazon would like to add more servers for better customer experience.

Both the above actions lead to increased customer satisfaction, which will eventually boost profits for the companies. **This feature of adding and removing servers is called Dynamic Scaling and is a unique feature of the Cloud.** Simply put, the users of the Cloud can scale to thousands of servers and scale down when appropriate and pay for what they use. However, that flexibility to add/subtract servers does not come with the on-premises servers, which is why the cost is always fixed. Also, during the slack time, many resources remain under- utilized which is a wastage of CAPEX.  One way of adding and deleting the EC2 instances is to do it manually which may lead to extra manual effort, increase in costs, and inaccurate results.

Another approach is to use **Auto Scaling to manage the EC2 instances automatically**. As Auto Scaling adds more EC2 instances, the software/application installation and configuration can be automated using the AMI (Amazon Machine Images).  In the previous use case, we have seen **how to capture custom metrics (number of users logged in) in the CloudWatch**. Here, we would need the same metric to manage (add/delete) the EC2 instances depending on the number of users logged into the website.

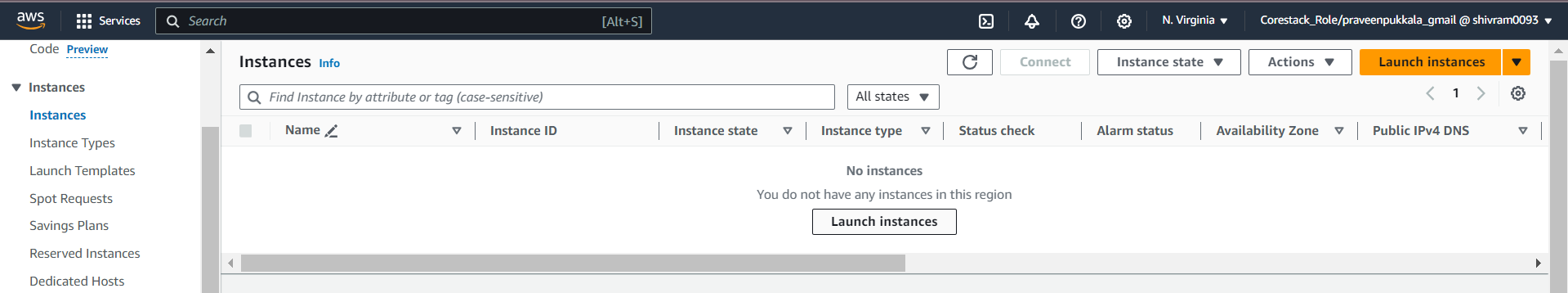
**Tools required:**AWS Services - CloudWatch, Auto Scaling, EC2

**Expected Deliverables:**

* Use Auto Scaling to manage the EC2 instances.
* Use EC2 instance and capture the metrics in the CloudWatch.

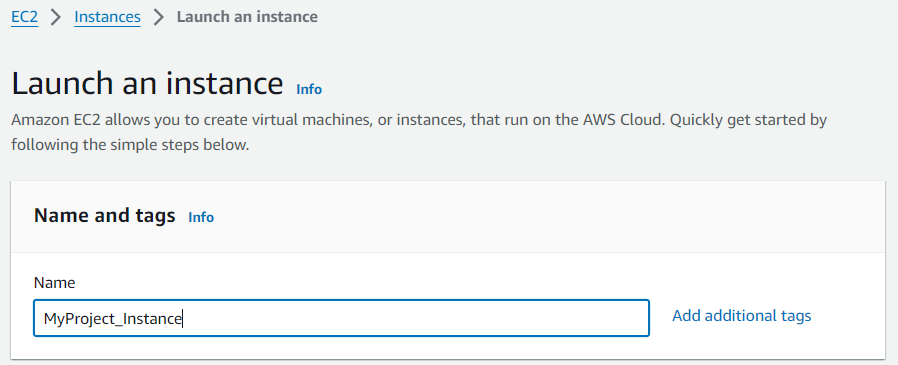
Approach:  
Here are the steps you might want to follow:

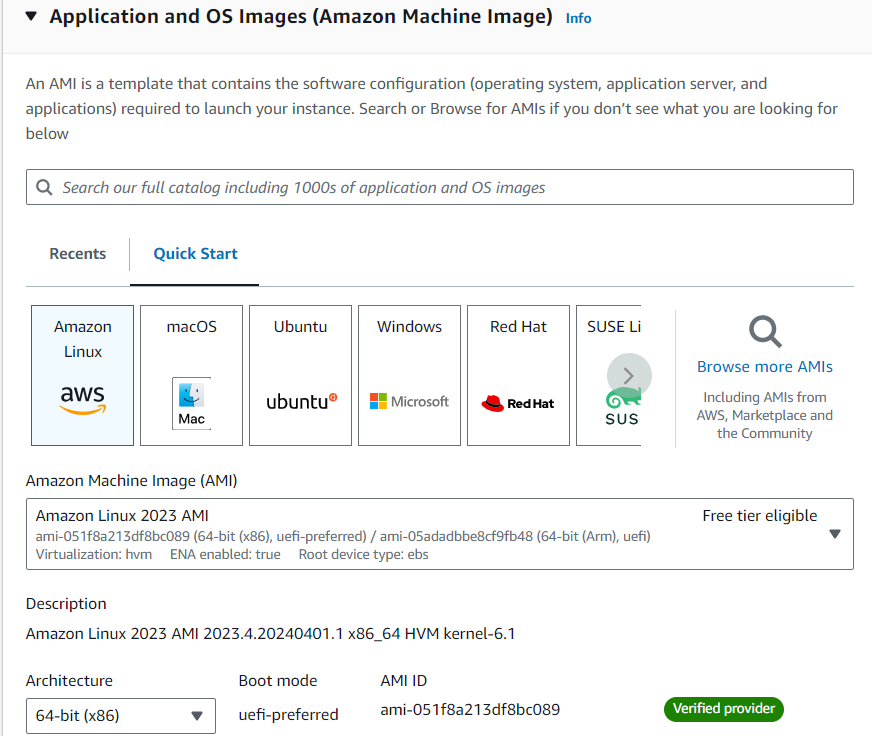
1. **Set up your EC2 instances**: Launch an EC2 instance with the necessary software and configurations for your application.



1. Once you have the instance set up as needed, create an Amazon Machine Image (AMI) from it. This AMI will be used by the Auto Scaling group to launch new instances.
2. Instance Name: MyProject\_Instance

Amazon linux 2023 AMI





A screenshot of a computer program

Description automatically generated

Create a new Keypair

A screenshot of a computer

Description automatically generatedScreens screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Rest is default and click create.  
A screenshot of a computer

Description automatically generated

**: Execute the below commands to install apache2 webserver and create a web page.**

#become a root

sudo su

# Get the list of available software packages

sudo yum update

# Install the Apache HTTP server

sudo yum install httpd

# Start the Apache HTTP server

sudo service httpd start

# Move to the default HTML folder

cd /var/www/html

# Delete the existing index.html file

sudo rm index.html

# Create a new index.html file with the specified content

echo "World's No. 1 Online Bootcamp" > index.html

: Access the webpage by typing the IP address of the EC2 in the browser.

It should display Welcome to App, which is the content of the webpage.

A screenshot of a computer

Description automatically generated

Step 5: Now, let’s create an AMI out of it.

Make sure to select the EC2 and go to Actions -> Image -> Create Image.

A screenshot of a computer

Description automatically generated

Step 6: Enter the name of the image and some description and click on Create.

image.

A screenshot of a computer

Description automatically generated

Step 7: Go to the AMI tab in the EC2 Management Console. Initially, the image will be in a pending status, within a few minutes, changes to available for the AMI to be used.

A screenshot of a computer

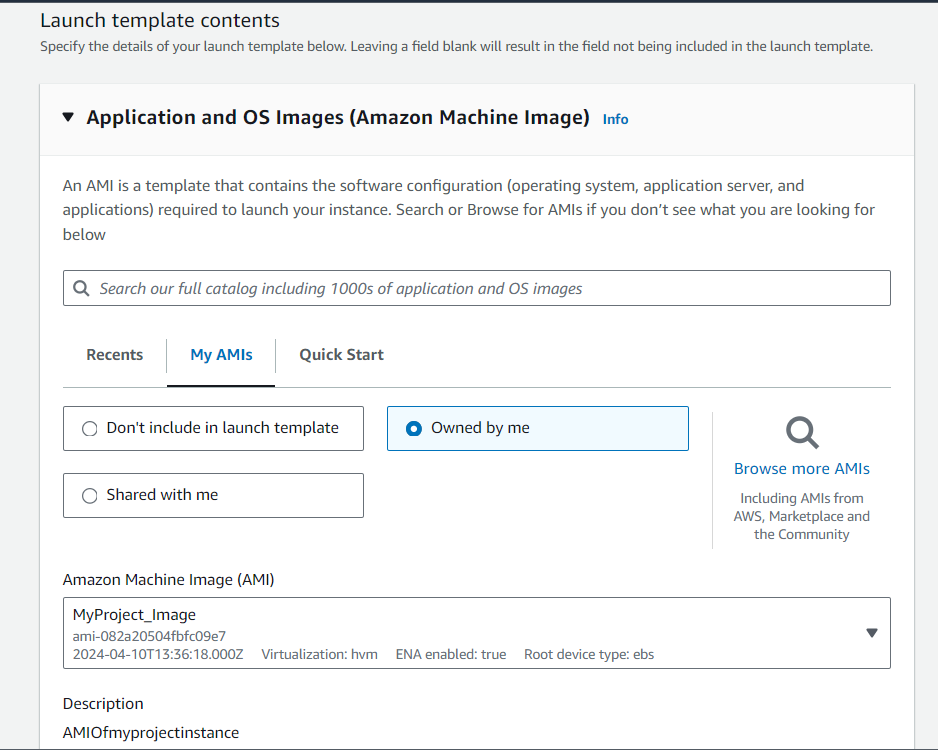
Description automatically generated  
The EC2 that was launched to create the AMI, is no more required and so can be terminated.  
A screenshot of a computer

Description automatically generated

Step 9: Go to the Launch Templates tab and click on the Create launch template.  
A screenshot of a computer

Description automatically generated

Select the AMI created in the previous step and select the instance type as t2. micro and select keypair to be used along with the EC2 instance.



Under the Security groups, select the one which allows the access to the SSH/22 and HTTP/80 traffic inbound. If there are no such Security groups, create one.

A screenshot of a computer

Description automatically generated

Go to the Advanced details and enable Detailed CloudWatch monitoring. This will allow the EC2 to send the metrics to the CloudWatch every 5 minutes instead of the default 1 minute  
A screenshot of a computer

Description automatically generated

Create

Go to the Auto Scaling Groups tab in the EC2 Management Console and click on Create an Auto Scaling group.

A screenshot of a computer

Description automatically generated

Select Subnet where the ASG can launch the EC2 instances. At least one subnet needs to be selected. Click on Next twice.

A screenshot of a computer

Description automatically generated

Specify the Desired and Minimum capacity as 1, the Maximum capacity as 4. This is the range of the EC2 instances that the ASG has to stick to. Click on Next thrice.

A screenshot of a computer

Description automatically generated

In the ASG, we have specified the Desired capacity as 1. So, the ASG with launch as EC2 with the details we have specified in the Launch Template

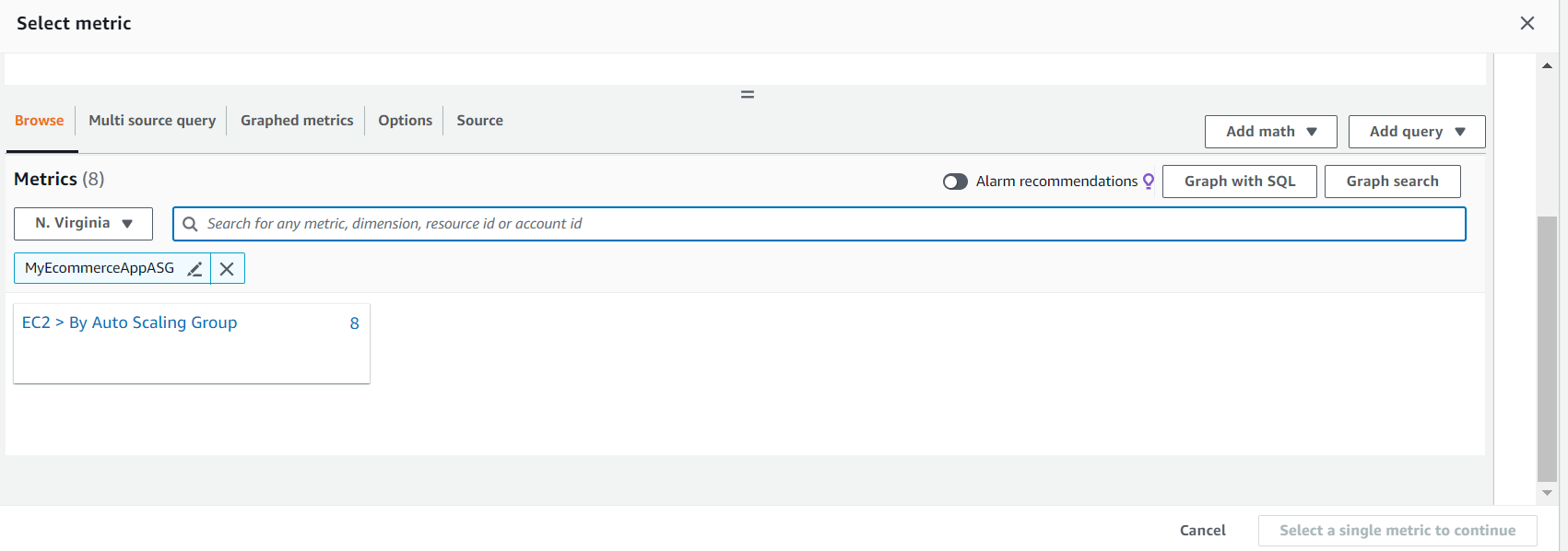
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

create an alarm for the ASG. When the alarm is breached, we can specify what action to take. Go to the CloudWatch Management Console, navigate to the Alarms tab, and finally click on Create alarm.



A screenshot of a computer

Description automatically generated

Network in (for user login count)  
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Initially, the alarm status will say insufficient data as it doesn’t have enough data to be in an OK or ALARM state. We can ignore this for now.

: Go back to the ASG in the EC2 Management Console. Expand the ASG created earlier!

Go to the Automatic scaling tab and click on Add policy.

: Select the Simple scaling as the policy type with ScaleUpPolicy as the name. in the Take the action, specify the number of units to add as 1. Click on Create to update the ASG.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Ensure the date in the name/description makes it clear when the image was created.

1. **Create a Launch Configuration**: A launch configuration is a template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances such as the ID of the Amazon Machine Image (AMI), the instance type, key pairs, and security groups.
2. **Set up CloudWatch Metrics**: AWS CloudWatch allows you to collect and track metrics for your AWS resources. In your case, you would want to monitor the number of users logged into your website. You can create a custom metric in CloudWatch to track this.
3. **Create an Auto Scaling Group**: An Auto Scaling group contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management. When you create an Auto Scaling group, you’ll need to specify the launch configuration to use, and the maximum, minimum, and desired number of EC2 instances in the group.
4. **Configure Auto Scaling Policies**: Auto Scaling policies determine when your Auto Scaling group should scale out (add more instances) or scale in (remove instances). You can create a policy to scale out your Auto Scaling group when the CloudWatch metric (number of logged-in users) exceeds a certain threshold, and another policy to scale in when the metric falls below a different threshold.

## **Test Your Setup**: Once everything is set up, you should test your setup by simulating high load on your application and verifying that new instances are launched, and vice versa when load is reduced. Approach ( Modified): **Corrected Project: Scaling EC2 Instance Based on CloudWatch Metrics**

This project outlines how to automatically scale an EC2 instance based on user load monitored by CloudWatch metrics. Here's an update using the latest AWS interface and best practices:

**Prerequisites:**

* An AWS account with appropriate permissions.
* Familiarity with the AWS Management Console and basic Linux commands.

**Steps:**

1. **Launch a Base EC2 Instance:**
   * Go to the EC2 service in the AWS Management Console.
   * Click on "Launch Instance".
   * Choose your desired AMI (e.g., Ubuntu 20.04 LTS).
   * Select an appropriate instance type (e.g., t2.micro).
   * Configure a security group that allows SSH (port 22) and HTTP (port 80) traffic inbound.
   * Launch the instance.
2. **Create a Launch Template:**
   * Once the instance is running, navigate to the EC2 service and select "Launch Templates".
   * Click on "Create launch template".
   * Provide a name and description for the template (e.g., "MyUbuntuTemplateForASG").
   * Under "AMI", select the ID of the running EC2 instance.
   * Choose the same instance type (t2.micro).
   * Select the key pair used for the launched instance.
   * Associate the security group created in step 1.
   * Under "Advanced Details", enable "Detailed Monitoring" to send metrics to CloudWatch every minute.
   * Create the launch template.
3. **Install Apache and Create a Webpage (Optional):**
   * Connect to the launched EC2 instance using SSH.
   * Update package lists and install Apache using the command line:

Bash

sudo apt update && sudo apt install apache2

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

* + Start the Apache service:

Bash

sudo systemctl start apache2

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

* + Create a simple index.html file with your desired content in the default web directory:

Bash

sudo echo "World's No. 1 Online Bootcamp" > /var/www/html/index.html

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

1. **Create an Auto Scaling Group (ASG):**
   * In the EC2 service, navigate to "Auto Scaling Groups" and click "Create Auto Scaling group".
   * Provide a name for the ASG (e.g., "MyEcommerceAppASG").
   * Under "Launch Template", select the template created in step 2.
   * Choose a VPC subnet where the instances will be launched.
   * Set the "Desired capacity", "Minimum capacity", and "Maximum capacity" to 1, 1, and 4 respectively. This allows the ASG to scale between 1 and 4 instances as needed.
   * Configure health checks to ensure launched instances are healthy.
   * Create the Auto Scaling group.
2. **Create a CloudWatch Alarm:**
   * Go to the CloudWatch service and navigate to "Alarms".
   * Click "Create alarm".
   * In the "Metric" section, select "EC2" as the namespace and choose "CPUUtilization" as the metric. You can also use a custom metric sent by your application (refer to AWS documentation for details).
   * For the dimension, select the Auto Scaling group name.
   * Set the "Statistic" to "Average" and the "Period" to 5 minutes (matches detailed monitoring).
   * Under "Alarm condition", choose "Greater ThanThreshold" and set the threshold value to a level that triggers scaling (e.g., 80%).
   * Configure the alarm to trigger scaling actions for the ASG in the next step.
   * Name the alarm appropriately (e.g., "AppHighCPUAlarm").
   * Create the alarm.
3. **Configure Auto Scaling Policies:**
   * Go back to the Auto Scaling group in the EC2 service.
   * Under the "Automatic scaling" tab, click "Add policy".
   * Choose "Scale-out policy" and provide a name (e.g., "ScaleUpPolicy").
   * Under "Add scaling action", select "Increase the desired capacity by" and set the value to 1. This adds one instance when the alarm is triggered.
   * Configure a cooldown period to prevent rapid scaling (optional).
   * Create the policy.
   * Repeat for a "Scale-in policy" that reduces capacity when CPU utilization falls below a desired threshold (e.g., 50%).
4. **Testing and Monitoring:**
   * You can manually test the scaling by simulating increased load on your application (if applicable). The alarm should trigger, and the ASG should launch a

## Improved Project: Scaling EC2 Instance Based on CloudWatch Metrics

This improved project addresses the provided feedback and utilizes the latest AWS interface for a smoother experience.

**Prerequisites:**

* An AWS account with appropriate permissions.
* Familiarity with the AWS Management Console.
* Basic understanding of EC2, Auto Scaling Groups (ASG), and CloudWatch.

**Steps:**

1. **Launch an Ubuntu EC2 Instance:**
   * Go to the EC2 service in the AWS Management Console.
   * Click on "Launch Instance."
   * Choose "Ubuntu Server 20.04 LTS (Focal Fossa)" as the AMI.
   * Select an appropriate instance type like "t2.micro" for demonstration purposes (consider cost and performance for real-world scenarios).
   * Configure a security group that allows SSH (port 22) and HTTP (port 80) traffic inbound.
   * Launch the instance.
2. **Install Apache and Create a Webpage (on the launched EC2):**
   * Connect to the instance via SSH.
   * Update package lists: sudo apt update
   * Install Apache: sudo apt install apache2
   * Start the Apache service: sudo systemctl start apache2
   * Move to the default web folder: cd /var/www/html
   * Create a new index.html file with content: echo "World's No. 1 Online Bootcamp" > index.html
   * Verify the webpage by accessing the instance's public IP address in a web browser.
3. **Create an AMI from the Instance:**
   * In the EC2 console, navigate to the instance you launched.
   * Under "Actions," select "Image" and then "Create Image."
   * Provide a descriptive name for the AMI that includes the date (e.g., "UbuntuWebServer-2024-04-10").
   * Click "Create image" to initiate the AMI creation process.
4. **Create a Launch Template:**
   * Go to the "Launch Templates" section in the EC2 service.
   * Click "Create launch template."
   * Specify a name like "MyUbuntuTemplateForASG" with a description.
   * Choose the AMI you created in step 3.
   * Select the same instance type (t2.micro) and key pair used for the initial launch.
   * Associate the security group created in step 1.
   * Under "Advanced details," enable "Detailed Monitoring" to send metrics to CloudWatch every minute.
   * Review and create the launch template.
5. **Create an Auto Scaling Group (ASG):**
   * In the EC2 service, navigate to "Auto Scaling Groups" and click "Create Auto Scaling group."
   * Provide a name like "MyEcommerceAppASG."
   * Select the launch template created in step 4.
   * Choose a VPC and subnet where the instances will be launched.
   * Set the desired capacity, minimum capacity, and maximum capacity to 1, 1, and 4, respectively. This allows the ASG to scale between 1 and 4 instances based on demand.
   * Configure health checks to ensure launched instances are healthy.
   * Review and create the ASG.
6. **Create a CloudWatch Alarm for Scaling:**
   * Go to the CloudWatch service and navigate to "Alarms."
   * Click "Create alarm."
   * Select "EC2 Auto Scaling" as the namespace and choose your ASG name.
   * Choose the metric "CPUUtilization" and set the statistic to "Average."
   * Set the comparison operator to ">" (greater than) and the threshold value to a suitable level based on your application's needs (e.g., 70%).
   * Configure the alarm period (e.g., 5 minutes) for when the threshold is breached.
   * Under "Actions," choose "Auto Scaling group" and select your ASG name.
   * In the "Action" section, specify "Scale Out" to add one instance when the alarm is triggered.
   * Configure additional notification settings if desired.
   * Review and create the alarm.
7. **Test Scaling:**
   * Use a load testing tool or simulate user traffic to increase the CPU utilization on the instance.
   * If the CPU utilization exceeds the threshold for a sustained period (as defined in the alarm), the ASG should automatically launch a new instance to handle the load.