AWS mini-project

**Resource Sentinel:**

**Your Vigilant Guardian for Optimized**

**Resource Management**

**By**

**Aravind U R**

Resource Sentinel is an advanced resource management system built on AWS that automates the monitoring, scaling, and optimization of cloud infrastructure. Designed to enhance efficiency and reduce costs, Resource Sentinel ensures that your cloud resources are always right-sized to meet your application's needs without overspending.

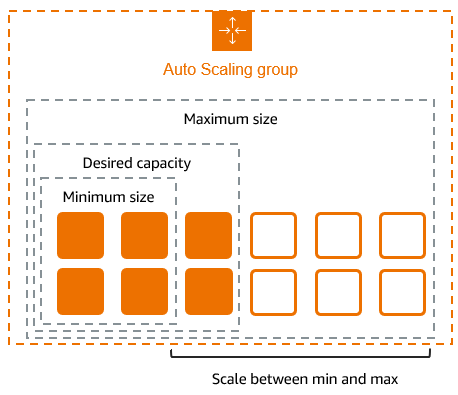
**Pre requisites Services**

1. **Application Load Balancer (ALB):**
   * **Role:** The ALB acts as the entry point for incoming HTTP traffic. It distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, within the Auto Scaling Group (ASG).
   * **Security Group (ALB SG):** This component listens for incoming HTTP traffic and ensures secure communication by controlling access to the ALB.
2. **Auto Scaling Group (ASG):**
   * **Role:** The ASG dynamically adjusts the number of EC2 instances based on real-time demand. It scales in or out to maintain the desired performance levels while optimizing resource usage.
   * **Security Group (ASG SG):** This controls the traffic flow between the ALB and the EC2 instances within the ASG, listening to all TCP traffic coming from the ALB SG.
3. **Launch Template:**
   * **Role:** Defines the configuration for the EC2 instances launched within the ASG, including instance type, AMI, key pairs, security groups, and other instance parameters.
4. **Amazon CloudWatch:**
   * **Role:** CloudWatch continuously monitors the performance and health of resources, collecting metrics and triggering alarms based on predefined thresholds.
   * **Functionality:** It enables Resource Sentinel to monitor critical metrics (e.g., CPU utilization, network traffic) and trigger scaling actions based on the resource demand.
5. **Scaling Policies:**
   * **Role:** These are predefined rules that determine when to scale in or out the EC2 instances within the ASG. Based on the CloudWatch metrics, these policies automatically adjust the number of instances to handle the traffic load.
6. **CloudWatch Alarms:**
   * **Role:** These alarms are set up to trigger specific actions when metrics reach certain thresholds. For example, if CPU usage is too high, an alarm will trigger the ASG to scale out and add more instances.
   * **Notifications:** Upon triggering an alarm, an email notification is sent via the SNS (Simple Notification Service) topic, keeping administrators informed about the scaling activities.
7. **SNS (Simple Notification Service) Topic:**
   * **Role:** Acts as a messaging service that sends notifications to subscribed endpoints, such as email addresses, when certain events occur (e.g., scaling events triggered by CloudWatch alarms).

Before going to the Architectural section, we need to know more about Auto scaling Policies. Because, it is a critical component of the Resource Sentinel project, playing a vital role in the management and optimization of cloud resources.

**AWS Auto scaling group**

**AWS Auto Scaling Group (ASG) is a service provided by AWS that automatically adjusts the number of Amazon EC2 instances in a specified group based on demand. It helps ensure that you have the right number of EC2 instances available to handle the load for your applications, which improves fault tolerance, scalability, and cost efficiency.**

**Desired capacity:**

The desired number of EC2 instances that the ASG tries to maintain at any given time. This is the target number of instances that you want running under normal conditions

**Minimum size:**

The minimum number of EC2 instances that the ASG should have running at all times. This ensures that even during low demand or after scaling in, the ASG will not go below this number of instances.

**Maximum size:**

The maximum number of EC2 instances that the ASG is allowed to scale up to. This sets an upper limit to prevent the ASG from launching too many instances during high demand.

**Common Use Cases:**

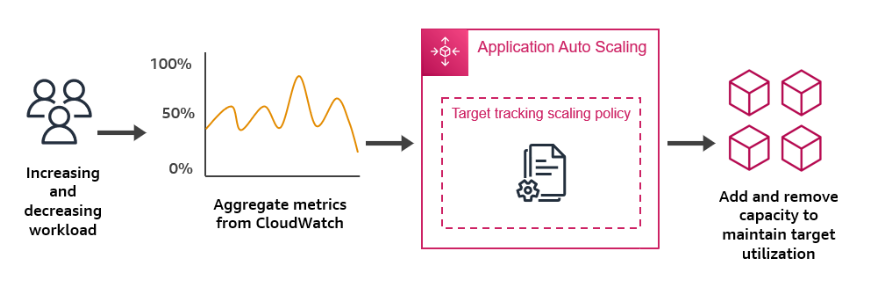
* **Web Applications**: ASG can scale web servers up or down based on the traffic to your website.
* **Batch Processing**: Automatically add instances during high-load processing and scale down when tasks are completed.
* **Disaster Recovery**: Ensure your application remains available by automatically replacing failed instances.

**Auto Scaling Policies**

* [Dynamic Scaling](https://jayendrapatil.com/aws-auto-scaling-policies/#Dynamic_Scaling)
* Target tracking
* Simple scaling
* Step scaling
* [Scheduled Scaling](https://jayendrapatil.com/aws-auto-scaling-policies/#Scheduled_Scaling)
* [Predictive Scaling](https://jayendrapatil.com/aws-auto-scaling-policies/#Predictive_Scaling)

**Dynamic Scaling policy:**

Automatically adjusts the number of Amazon EC2 instances in your Auto Scaling Group (ASG) based on real-time demand responds to CloudWatch Metrics.

****

**Target Tracking Scaling Policy**

* Description: Automatically adjusts the number of instances to maintain a specific target metric (like CPU utilization) at a desired value.
* Example: Keep the average CPU utilization at 50%.

**Step Scaling Policy**

* Description: Adjusts the number of instances in response to changes in a monitored metric, with scaling actions defined in steps.
* Example: Add one instance if CPU utilization goes above 60%; add two instances if it goes above 80%.

**Simple Scaling Policy (Less commonly used)**

* Description: A basic form of scaling where you define a single adjustment for when a metric reaches a specific threshold.
* Example: Add one instance when CPU utilization exceeds 70%.

**Scheduled scaling policy:**

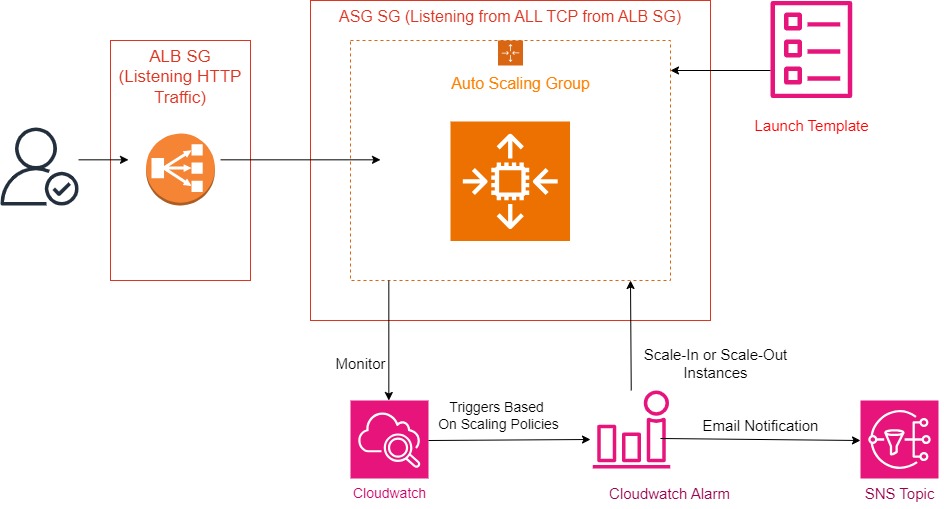
**Scheduled Scaling** is a feature of AWS Auto Scaling that allows you to automatically adjust the number of Amazon EC2 instances in your Auto Scaling Group (ASG) based on a pre-defined schedule. This is particularly useful when you know that your application will experience predictable changes in demand, such as increased traffic during business hours or a spike in usage during a specific event.

**Predictive scaling policy:**

Predictive Scaling in AWS Auto Scaling Group (ASG) is an advanced feature that uses machine learning to forecast the future load on your application and automatically adjusts the number of EC2 instances in your ASG to meet that anticipated demand. This helps you ensure that your application has the right amount of capacity at the right time, improving performance and cost-efficiency.

Let’s Begin!

**Architectural Diagram**



**Solution Configuration**

**1. Create Security Groups for ASG and ALB**

* **SGForALB**: Allow HTTP from Internet

Allow http – 0.0.0.0/0

* **SGForASG:** AllowSSH from internet and http from SGForALB

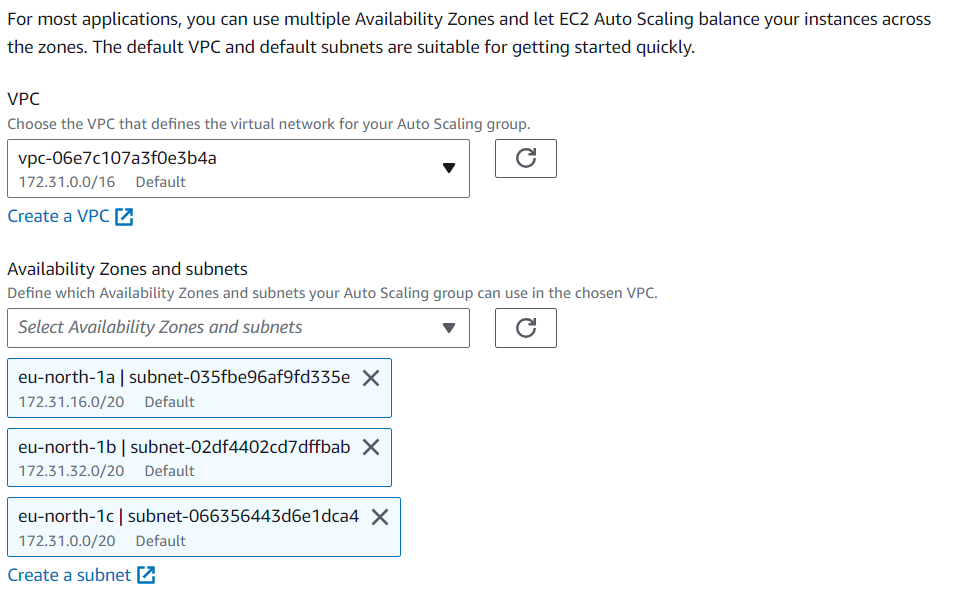
Allow SSH – 0.0.0.0/0

Allow HTTP – SGForALB

**2. ASG, Launch Template and ALB Configuration**

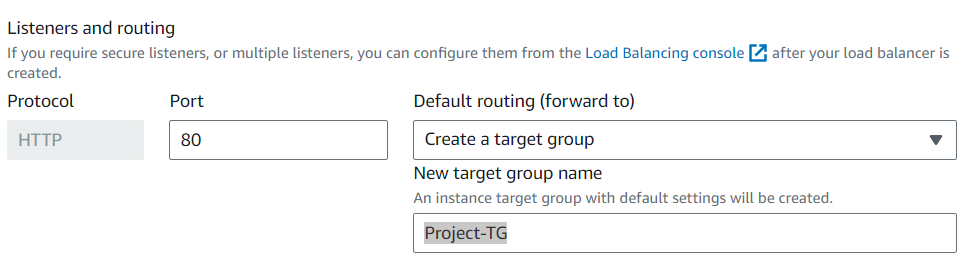
* **Navigate to Auto Scaling Groups**:
* In the EC2 dashboard, under **Auto Scaling**, select **Auto Scaling Groups**.
* Click **Create Auto Scaling group**.
* Name for ASG: Product-ASG
* Click on **Create launch template**.
* Name: Project-LT
* Choose the AMI (Amazon Machine Image) that your instances will use.
* Select the Instance type (e.g., t3. micro).
* Configure Key pair, Security groups, and Storage options.
* Add user data for install an Apache web server and “hello world” simple html web page.
* Review the settings and click Create launch template.
* In next step (Choose instance launch option):

Choose all 3 availability zones under networking section;



* In Configure advanced options; select Attach to a new load balancer.
* Load balancer type: Application load balancer
* Name: Project-ALB
* Internet facing
* Under Listeners and Routing option;

http – Port 80 – Create a Target Group - **Project-TG** (Automatically creates a target group for instances added in ASG)



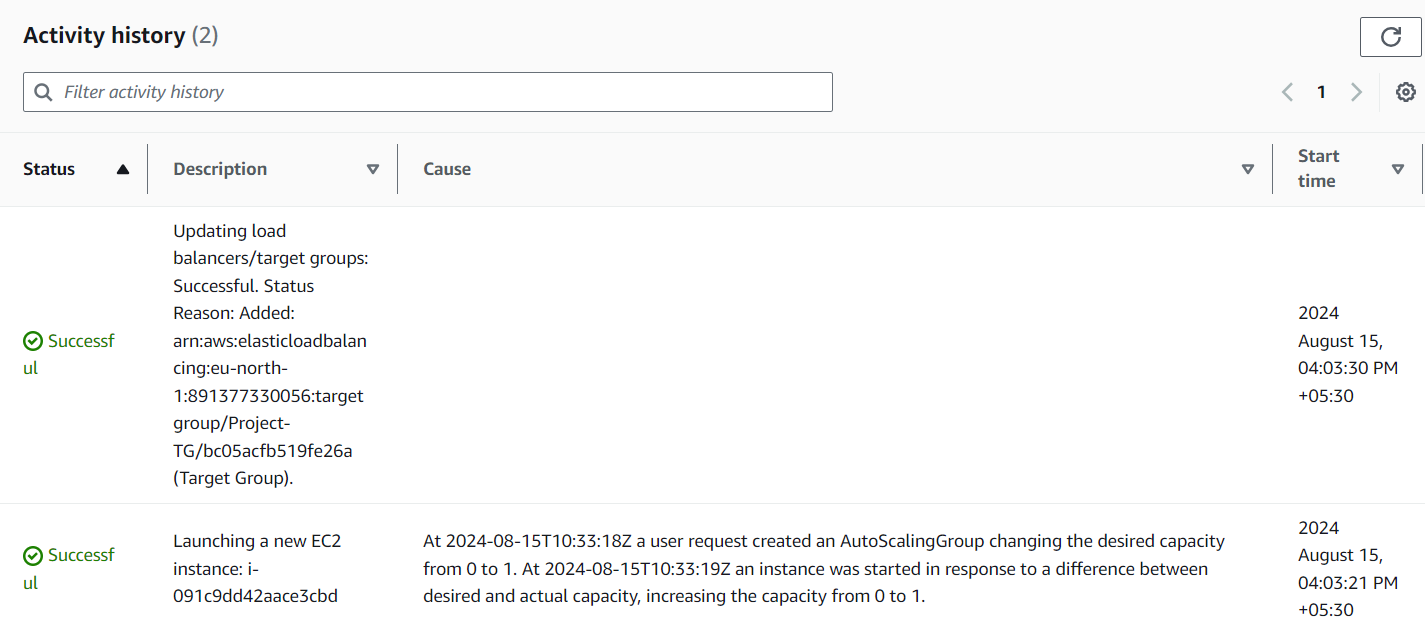
* Under Health check option EC2 health check is enabled in default. Also Turn on Elastic Load Balancing health checks.
* Configure Group sizing and scaling

Desired capacity – 1

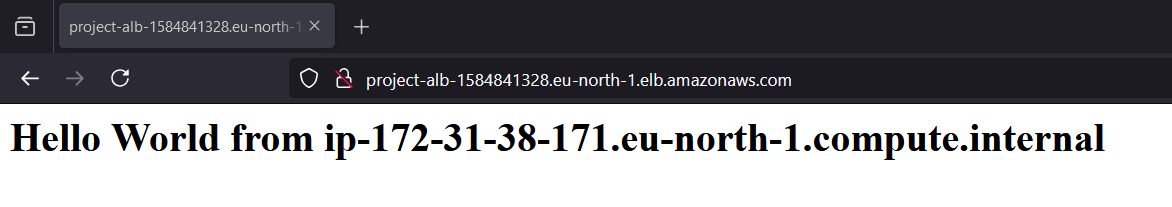
Minimum – 1

Maximum – 2

* We do not specify and Scaling policies now.
* Review all the configuration we have done and click on create Auto scaling group.
* Under Activity option in ASG we can observe the Instances are creating as per the group sizing- 1 Instance added as Desired capacity.



* Security group for the created ALB is now configured with SGForASG, we need to manually change it to SGForALB; Then only we can use ALB DNS for view the web page hosted in instances in ASG.
* Now we can see an Instance Launched under Instances Tab in Running Status.
* We can check the web server status by copying the DNS name of ALB to a browser – it will show the web page.



Our First part was completed, now we going to the main section!!

I have planned here to configure Dynamic scaling policies for this project.

*Skip the predictive scaling policy because we need to run the ASG for at least 14 days for observing the traffic for done this. Also skipped the step scaling, and use simple scaling policy for SNS notification creation.*

**3. Dynamic Scaling Policy Configuration**

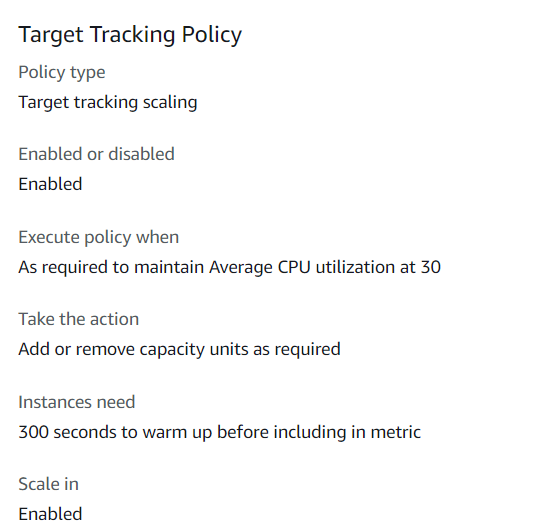
In this step we are configuring the main part of our Resource Sentinel project.

**Configuring Target Tracking scaling policy based on CPU Utilization CloudWatch Metrics:**

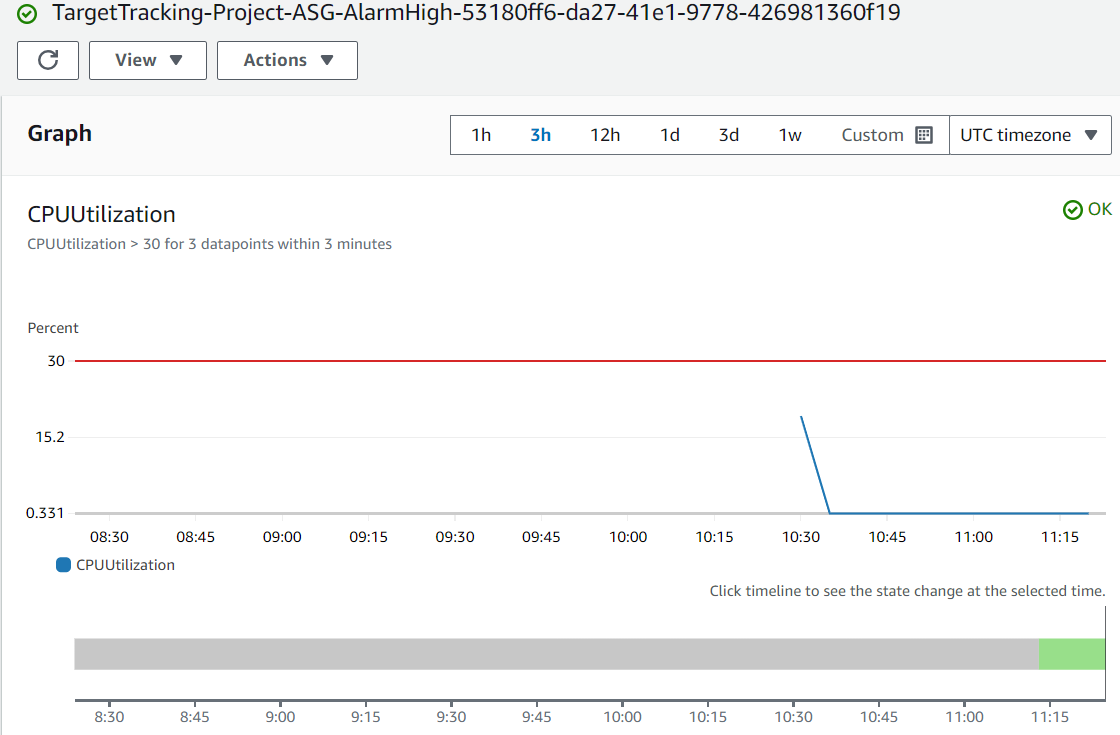
Here we have configuring a CloudWatch metrics for CPU Utilization of instances in ASG. If the utilization increases above 30% the alarm status changed to In-alarm and add an additional ec2 instance for balancing the load.

* Create a Target Tracking Scaling Policy
* In the **Details** section of the ASG, choose the **Automatic Scaling** tab.
* Click on **Add policy** and select **Target tracking scaling policy**.
* Configure the Target Tracking Scaling Policy

**Warm-Up Period:** After an instance is launched, the Auto Scaling group waits for the warm-up period to complete before including the instance's metrics in the average CPU utilization calculation. This prevents premature scaling decisions based on unstable or partial metrics.



Now we can observe an Alarm (In Ok state) was automatically created in CloudWatch based CPU Utilization of our instance.



A metric alarm has the following possible states:

**OK** – The metric or expression is within the defined threshold.

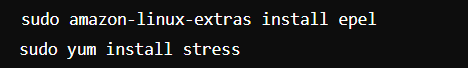
**In - ALARM** – The metric or expression is outside of the defined threshold.

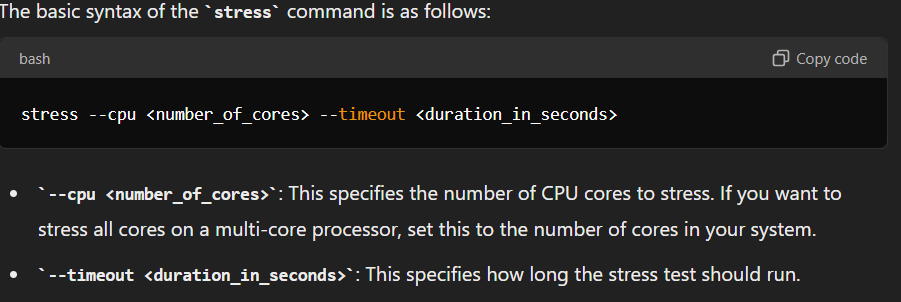
**INSUFFICIENT\_DATA** – The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

Now we are going to Increase the CPU Utilization using Stress Utility.

**The stress utility is a simple tool that can be used to artificially increase CPU utilization, which is useful for testing purposes, such as validating an Auto Scaling policy or monitoring system performance under load.**

* For creating load on Instance, we have connected the instance using Instance connect.
* Then run the below commands to increase the CPU utilization above 30%.
* Install Stress Utility

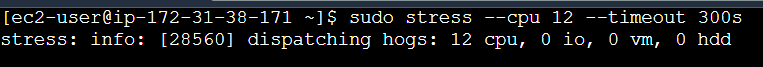




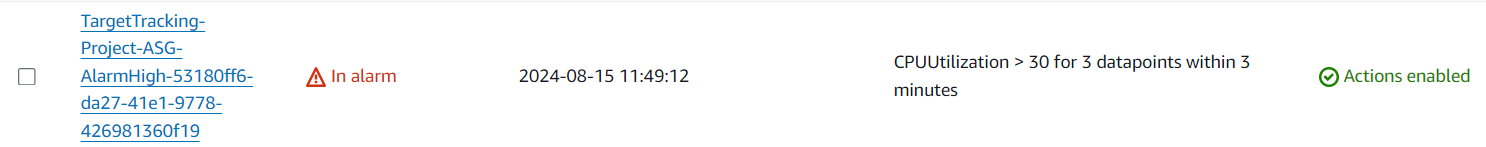
Here;

stress --cpu 12 --timeout 300s

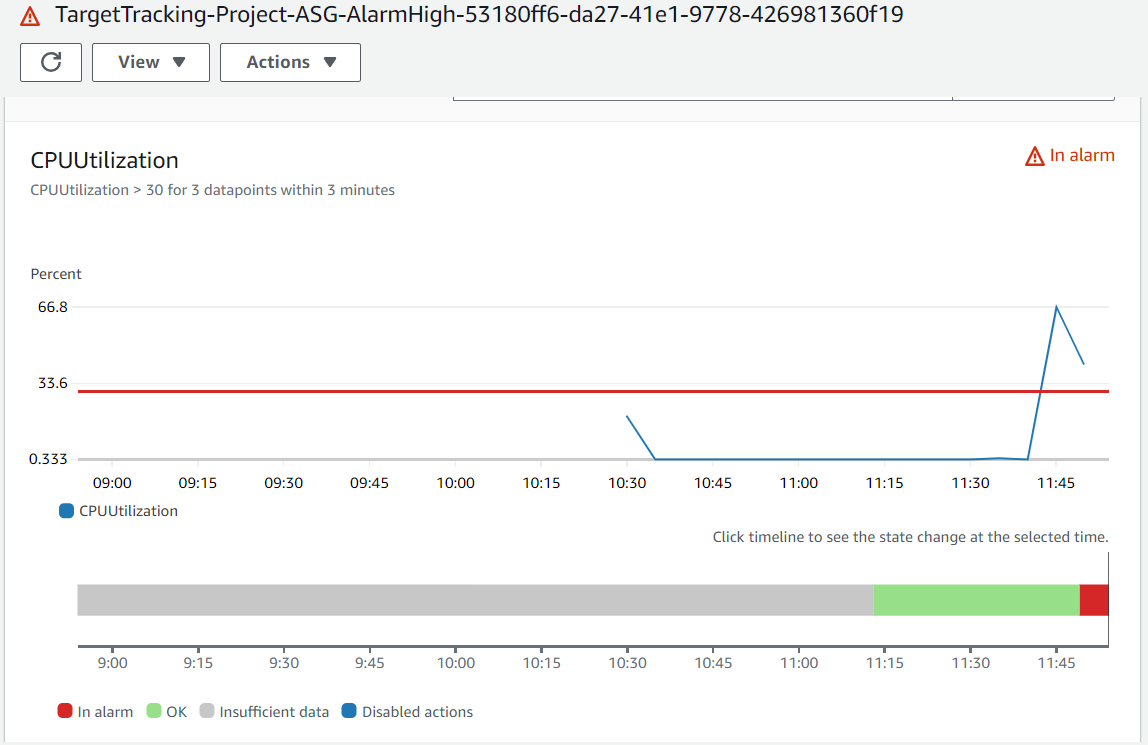
Create the stress for 5min.



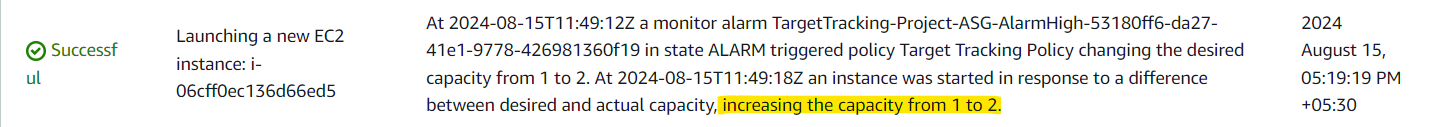
Now we can go to the alarm section; there we can observe our Alarm is in In-alarm state.



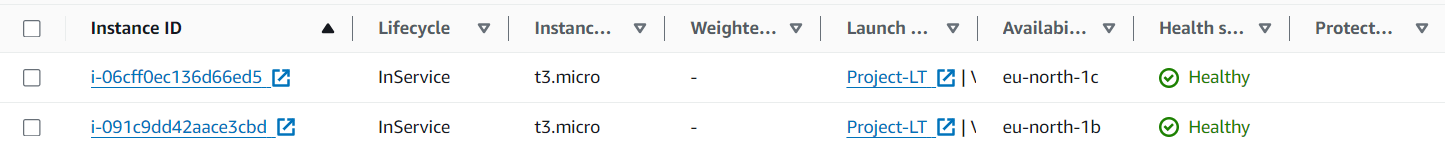
And inside the alarm the Graph shows the CPU Utilization above the threshold of 30%.



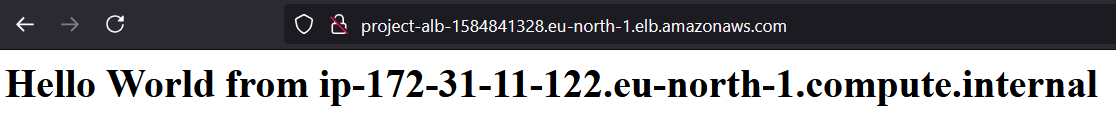
And now at ASG Activity section, the scaling groups automatically adds an additional instance now.

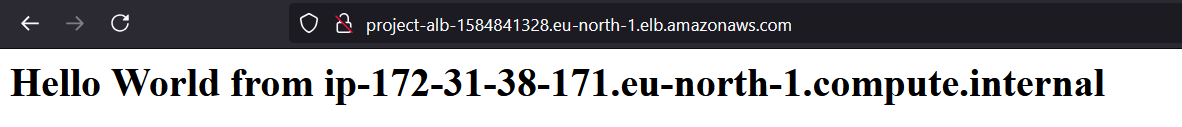


Under Managed instances tab, now its two EC2 are launched.



Also, we can observe the ALB balancing the load on each instance by running the ALB DNS on a browser.





Its also scale out to one instance once the CPU Utilization decreases to below threshold value.

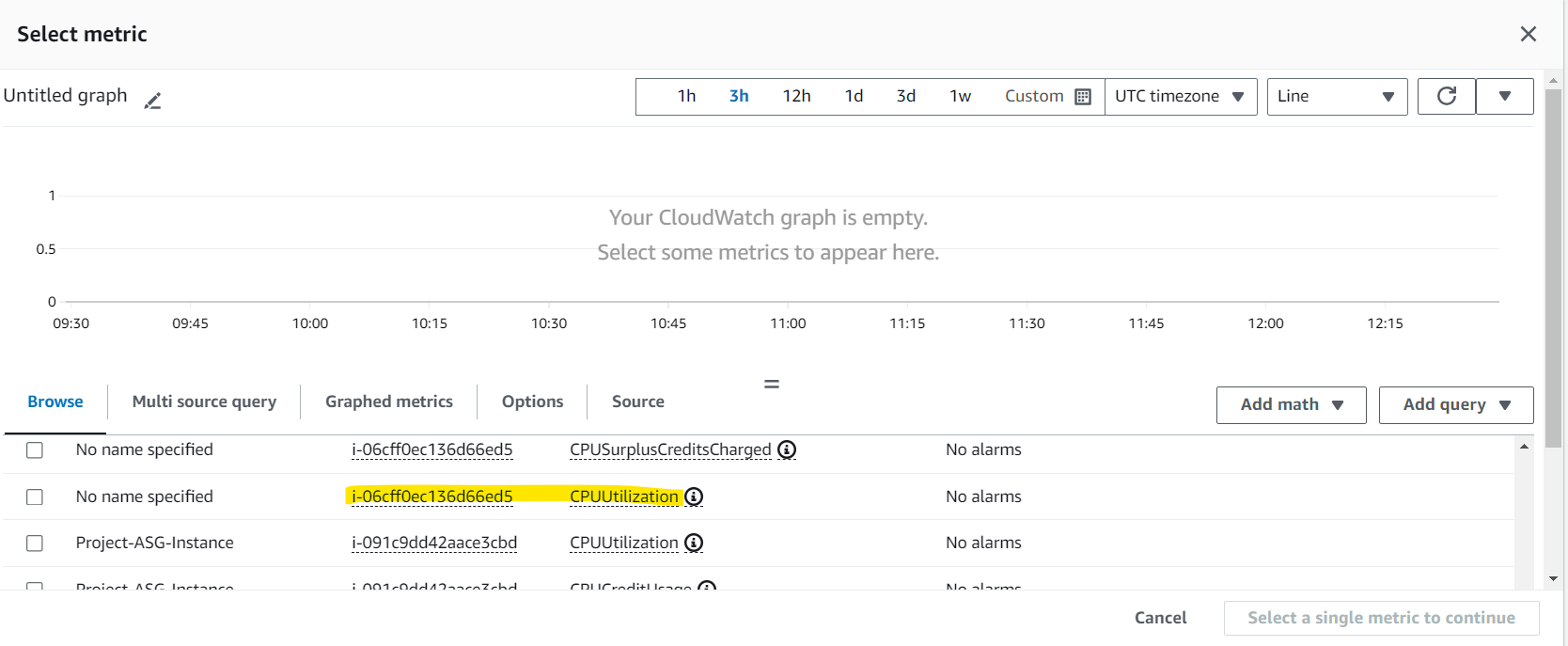
So, we can conclude that our Target tracking scaling policy was successfully configured.

Then Adding an additional policy for creating an Email alerting system for enhancing efficiency of our Architecture.

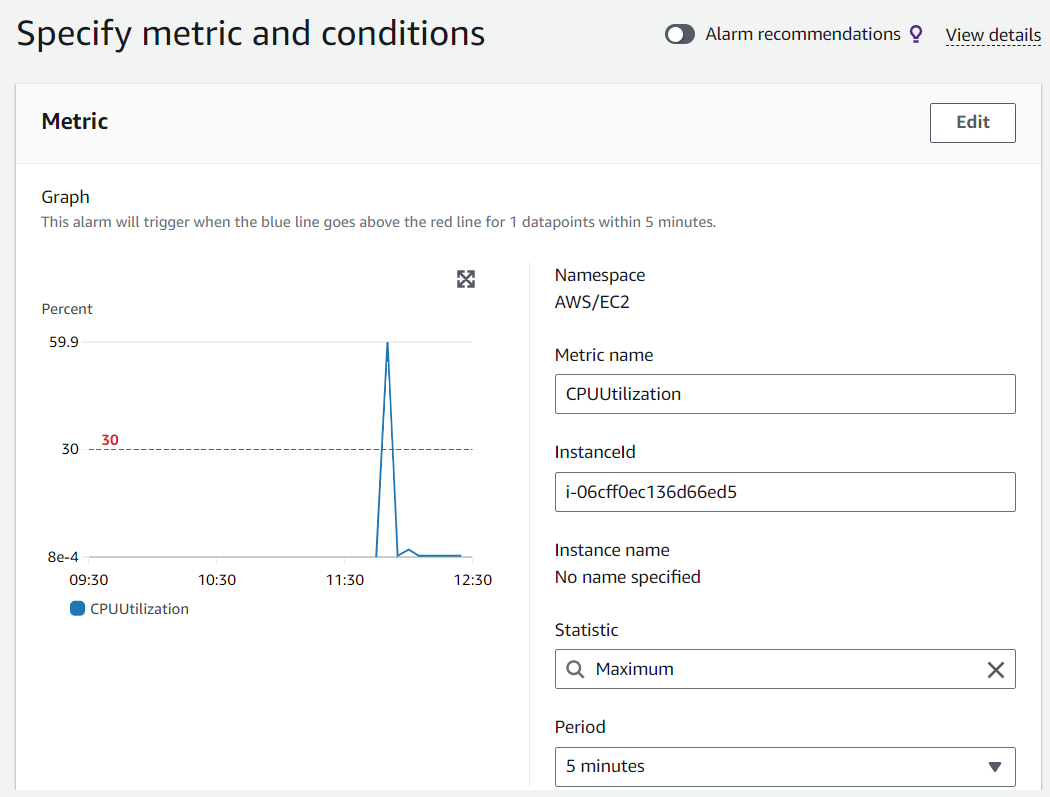
**Configuring Simple scaling policy based on CPU Utilization CloudWatch Metrics for Enabling Email Alerting:**

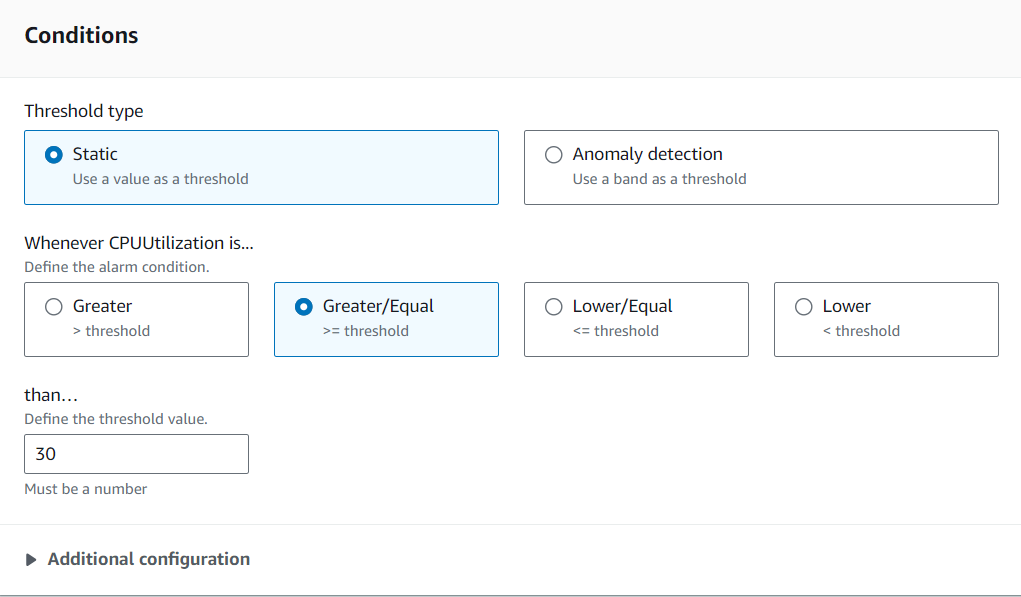
Here we are using AWS SNS (Simple notification service) enabling email alerts.

* Going to Dynamic scaling policy
* Select policy type as Simple scaling.
* Scaling Policy name: CPU-utilization-email-alert
* Click on create a CloudWatch Alarm
* Specify metric and conditions – Select Metrics
* Browse CPU Utilization Metrics for our Instance ID;

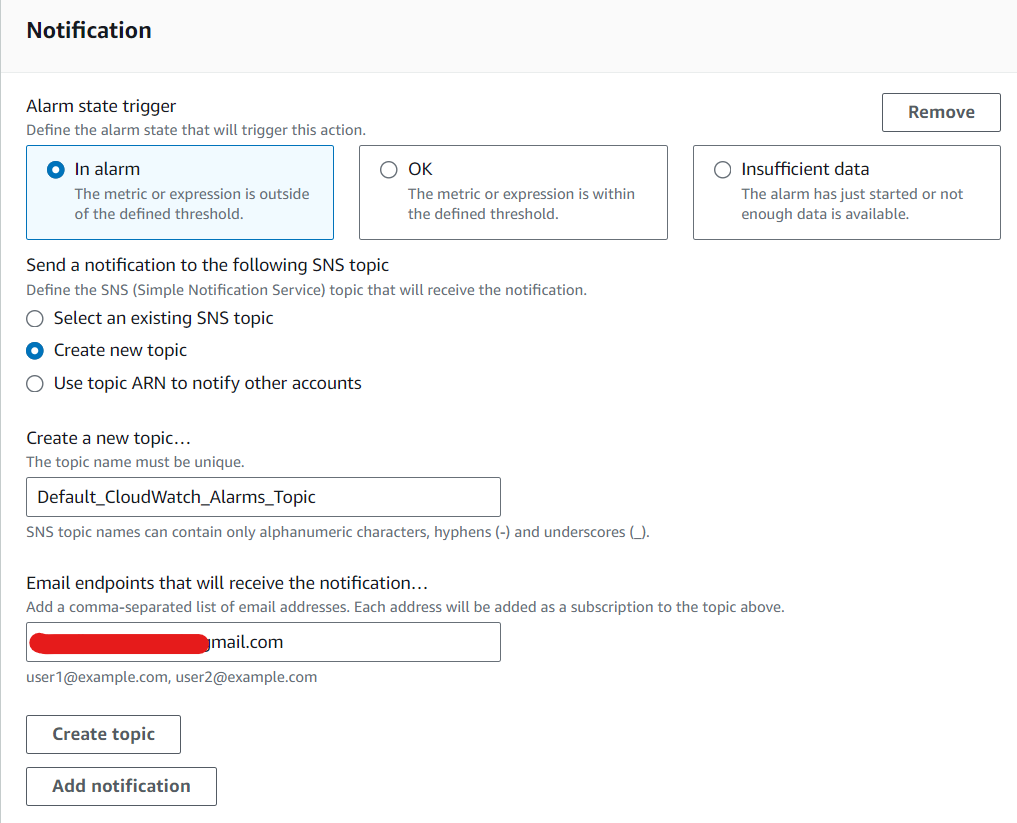


* Specific Metrics conditions as below;

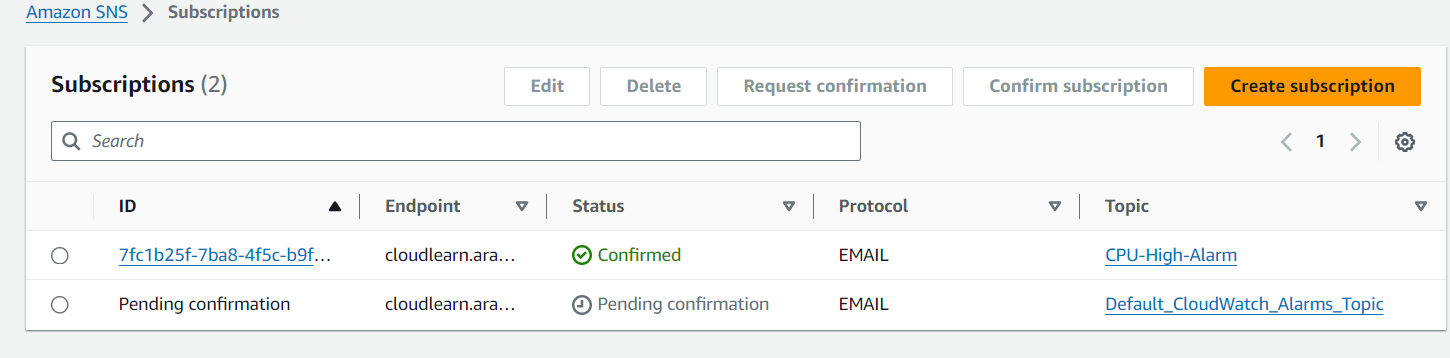




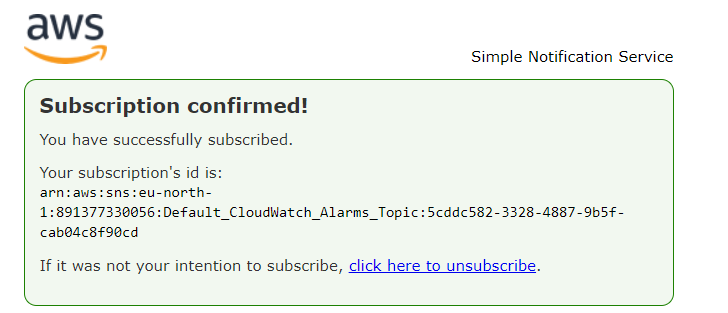
* Choose Alarm state trigger as In-Alarm
* Send notification section – Choose create a new topic.
* Give your email id here for send notifications.



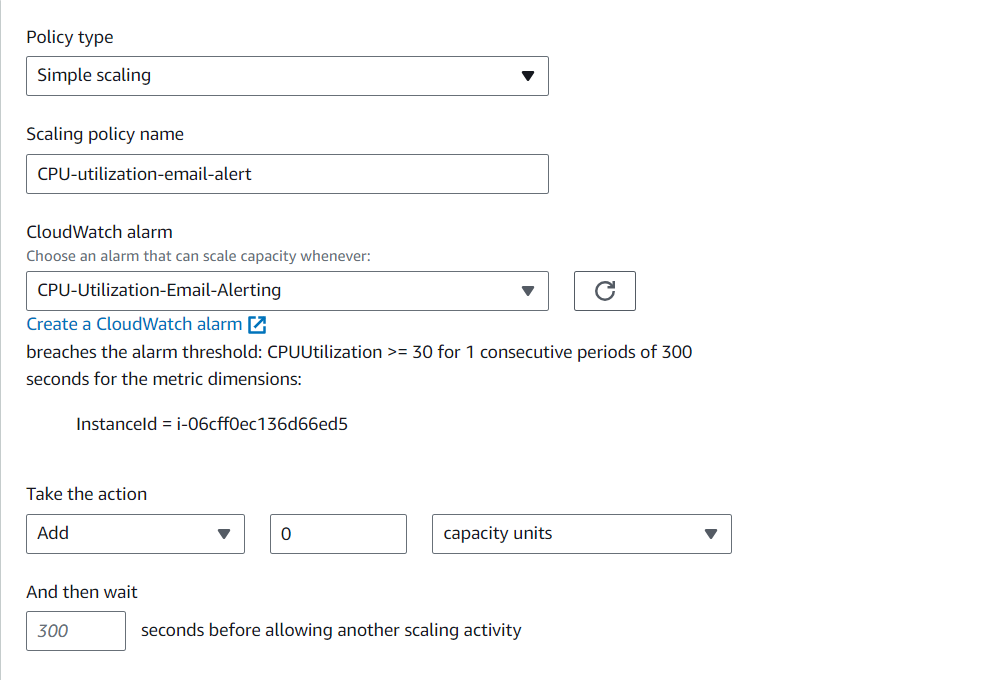
* Click on create a topic
* Give Alarm name: CPU-Utilization-Email-Alerting
* Review and click on Create Alarm.
* We can see the SNC Subscription details there;



* Provide the confirmation from the mail received on your mentioned mail id.



* Go back to policy creation window
* Select the CloudWatch Alarm which we have created.
* Increase the count of capacity units to 1.



* Then click on create.
* Then we have to set the alarm state to 30%
* So, we use an alarm set command for that;

Browse the below URL;

https://docs.aws.amazon.com/cli/latest/reference/cloudwatch/set-alarm-state.html



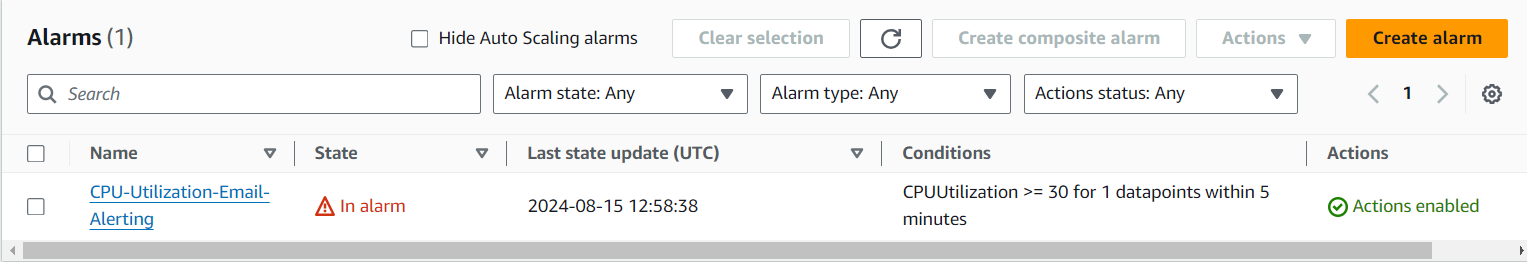
Edit the command as per our need;

aws cloudwatch set-alarm-state --alarm-name "CPU-Utilization-Email-Alerting" --state-value ALARM --state-reason "testing purposes"

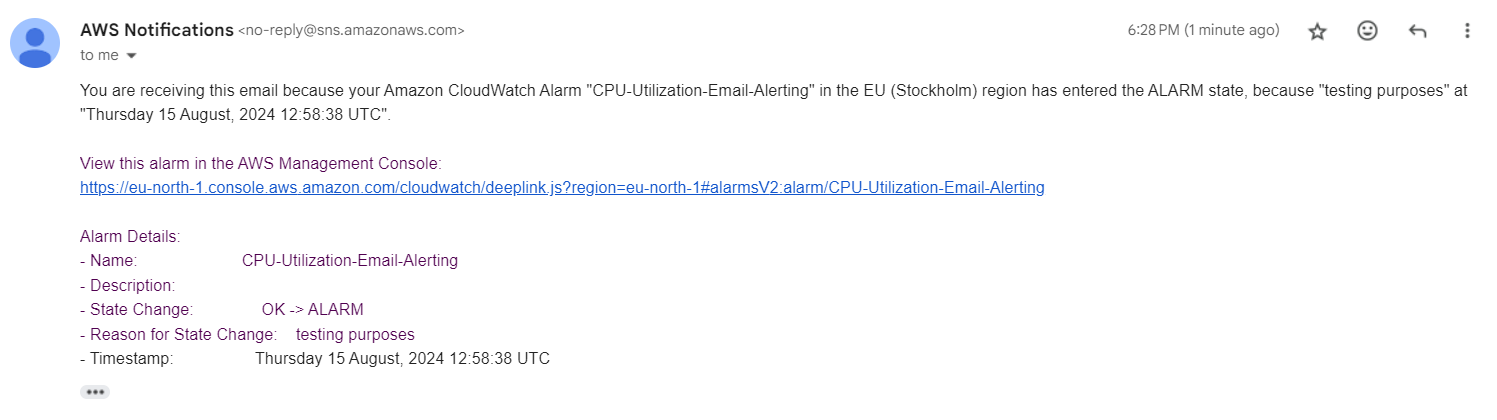
* Run the command in AWS Cloud shell;



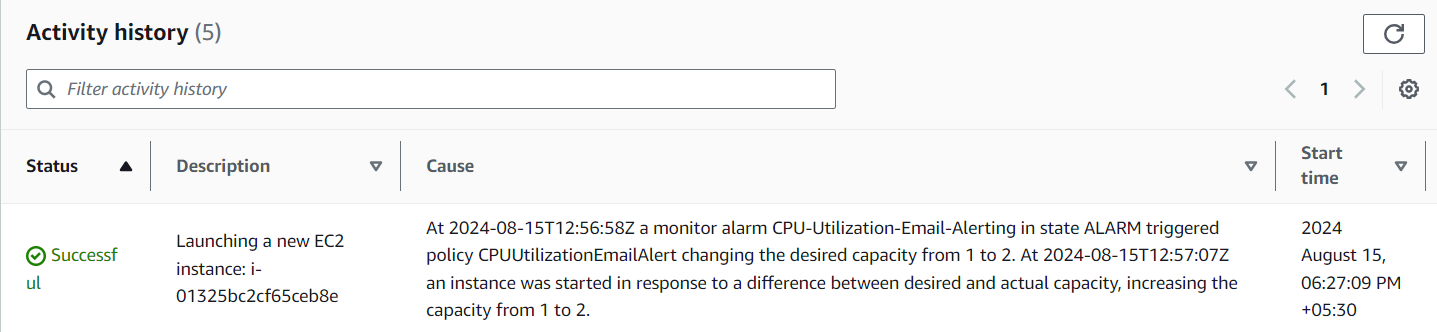
* Now we can see the alarm state going to In-alarm in CloudWatch alarm section.



* And we can observe the email notification was received on our mail.



* Also, an EC2 Instance was added on our ASG Activity tab.



Then our all the configurations are Working successfully.

**Conclusion**

Auto Scaling policies are foundational to the effectiveness of Resource Sentinel. They provide the mechanism for dynamic, responsive, and cost-effective resource management, ensuring that your cloud infrastructure is always optimized for performance, availability, and cost. By automating the scaling process, these policies enable Resource Sentinel to fulfil its role as a vigilant guardian over your resources, maximizing efficiency and minimizing waste.

Resource Sentinel is a versatile solution that can be applied across various industries to enhance resource management, optimize operations, and reduce costs. Here’s how Resource Sentinel can be utilized in different industrial scenarios:

**1**. **In Manufacturing Industry:** Resource Sentinel can dynamically scale the compute resources required for managing production line automation and monitoring systems based on real-time demand.

**2. Retail and E-commerce:** During flash sales or major promotions, Resource Sentinel automatically scales up the web servers and backend systems to handle the surge in traffic, ensuring that the website remains responsive and customers have a smooth shopping experience.

**3. Healthcare Industry:** Resource Sentinel ensures that the cloud-based electronic health record (HER) system can handle peak usage times, such as during a public health crisis or flu season, by automatically scaling up resources

**4. Financial Services:** A financial institution runs a suite of trading platforms and risk management systems that require high availability and low latency, especially during market hours.