**EX: 06 NAME: M. NIVASHINI**

**DATE: 06.08.2024 ROLL.NO.: 23MCA020**

**Auto Scaling Implementation**

**Aim:**

To configure an Auto Scaling Group with defined capacity parameters and observe its ability to scale up instances in response to increased CPU utilization on a Linux server, ensuring dynamic load handling.

**Theoretical Background**

**1. Create a Launch Template or Launch Configuration**

1. **Open the AWS Management Console.**
2. **Navigate to the EC2 Dashboard.**
3. **Click on “Launch Templates” (or “Launch Configurations” if you're using an older version).**
4. **Create a New Launch Template:**
   * Click on “Create launch template.”
   * Provide a name for the launch template.
   * Select an AMI (Amazon Machine Image) that you want to use.
   * Choose an instance type (e.g., t2.micro).
   * Configure other settings such as key pairs, security groups, and storage as needed.
   * Click “Create launch template” when done.

**2. Create an Auto Scaling Group**

1. **Navigate to the EC2 Dashboard and select “Auto Scaling Groups” from the menu.**
2. **Click on “Create Auto Scaling group.”**
3. **Select the Launch Template or Launch Configuration you created earlier.**
4. **Specify the Auto Scaling Group settings:**
   * **Auto Scaling group name:** Provide a name for your ASG.
   * **Network:** Choose the VPC and subnets where your instances will be launched.
   * **Desired capacity:** Set the initial number of instances.
   * **Minimum capacity:** Set the minimum number of instances.
   * **Maximum capacity:** Set the maximum number of instances.
5. **Configure Scaling Policies:**
   * **Add Scaling Policies:** Click “Add Policy” to define how the ASG should scale in and out.
   * **Policy Type:** Choose “Simple” or “Target Tracking” scaling policy. Target Tracking is often preferred for automatic adjustments.
   * **Scaling Metric:** For example, use “Average CPU utilization” as the metric.
   * **Target Value:** Set the desired CPU utilization percentage (e.g., 60%).
6. **Review and Create:**
   * Review all the configurations.
   * Click “Create Auto Scaling group.”

**3. Create a Cloud Watch Alarm for Scaling**

1. **Navigate to the Cloud Watch Dashboard.**
2. **Select “Alarms” from the left-hand menu and click “Create Alarm.”**
3. **Choose a Metric:**
   * Select “EC2” and then “Per-Instance Metrics.”
   * Choose “CPU Utilization” for the metric you want to monitor.
4. **Configure the Alarm:**
   * Set the conditions for the alarm (e.g., CPU utilization greater than 90%).
   * Define the actions for the alarm (e.g., trigger scaling up or down).
5. **Create Alarm:**
   * Provide a name for the alarm.
   * Click “Create Alarm.”

**4. Increase CPU Utilization Manually**

1. **SSH into one of the EC2 instances in your ASG.**
2. **Run the yes command to stress the CPU:**

sudo su

top

q

yes > /dev/null &

top

* + This will create high CPU utilization. You can stop this by pressing Ctrl + C when you want to stop the stress test.

**5. Observe Auto Scaling Behavior**

1. **Go back to the AWS Management Console.**
2. **Navigate to the Auto Scaling Groups section.**
3. **Select your Auto Scaling Group and view the “Activity” tab to monitor the scaling activities.**
4. **Observe the Scaling Policies in action:**
   * Watch for new instances being launched as the CPU utilization increases.
   * Verify that the ASG scales up according to the policies you set.

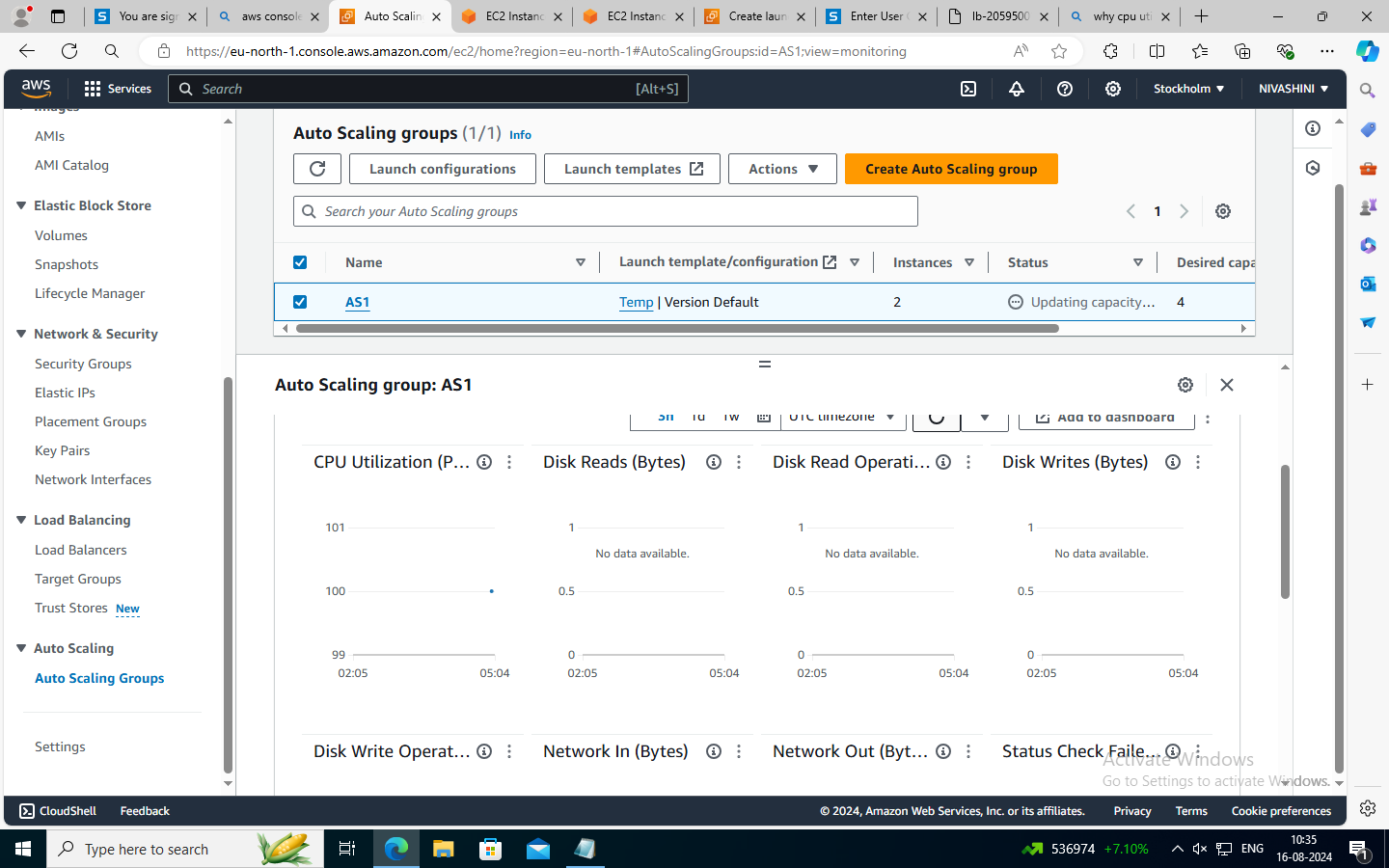
**6. Test Auto Scaling Down**

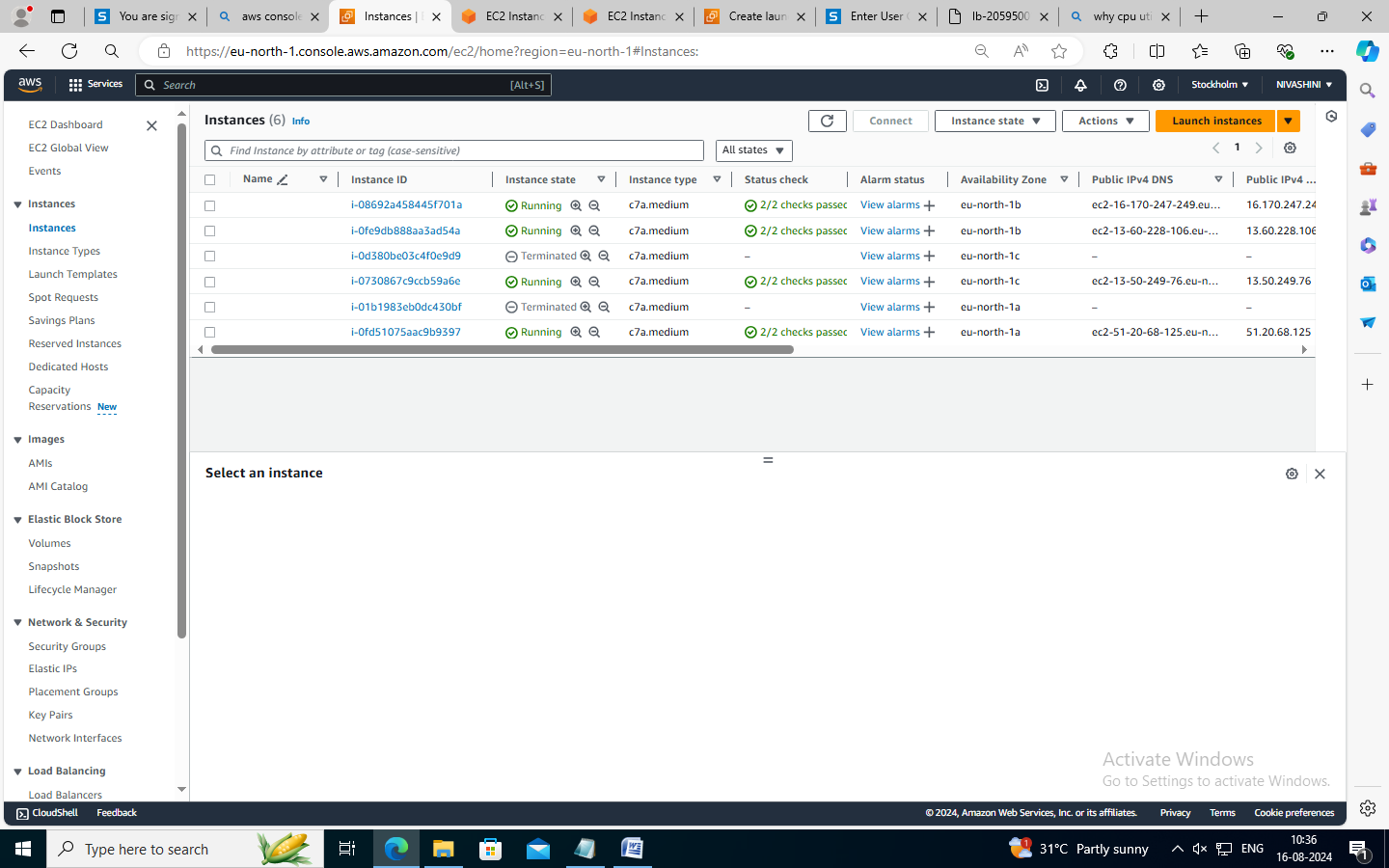
1. **Stop the stress test on the EC2 instance.**
2. **Monitor the Cloud Watch alarms and Auto Scaling activity to see if the ASG scales down as CPU utilization decreases.**

kill process id

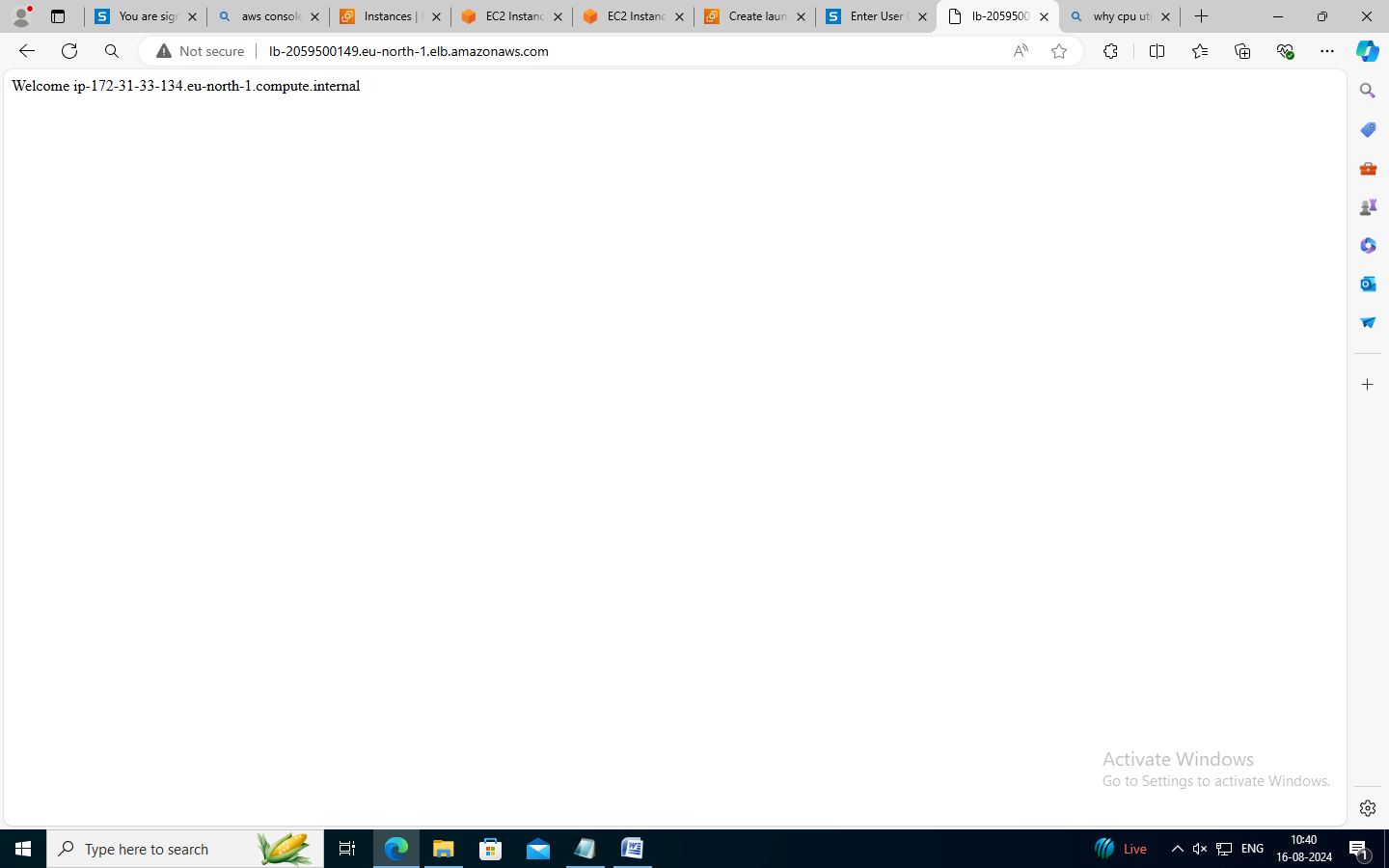
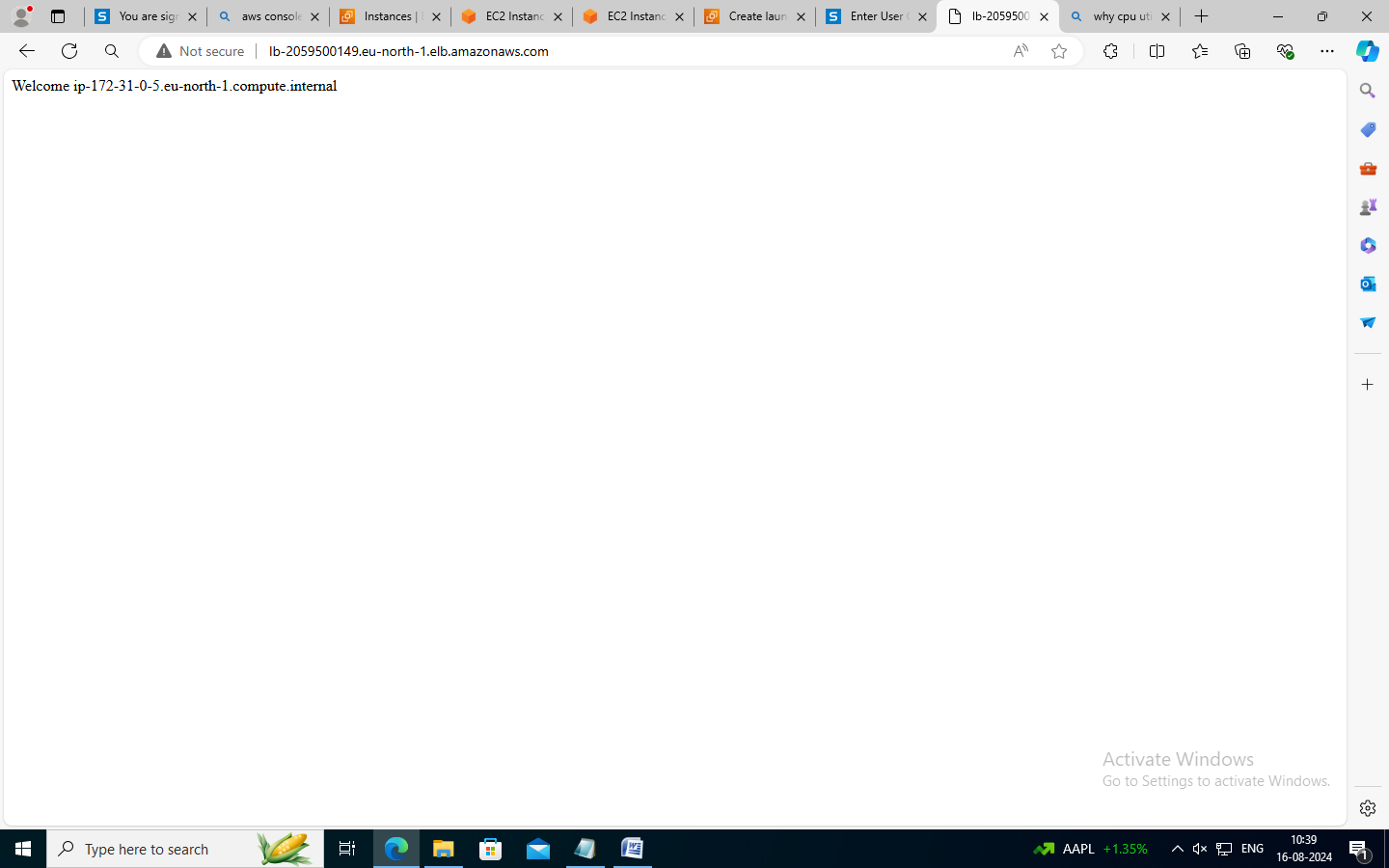
**Output**

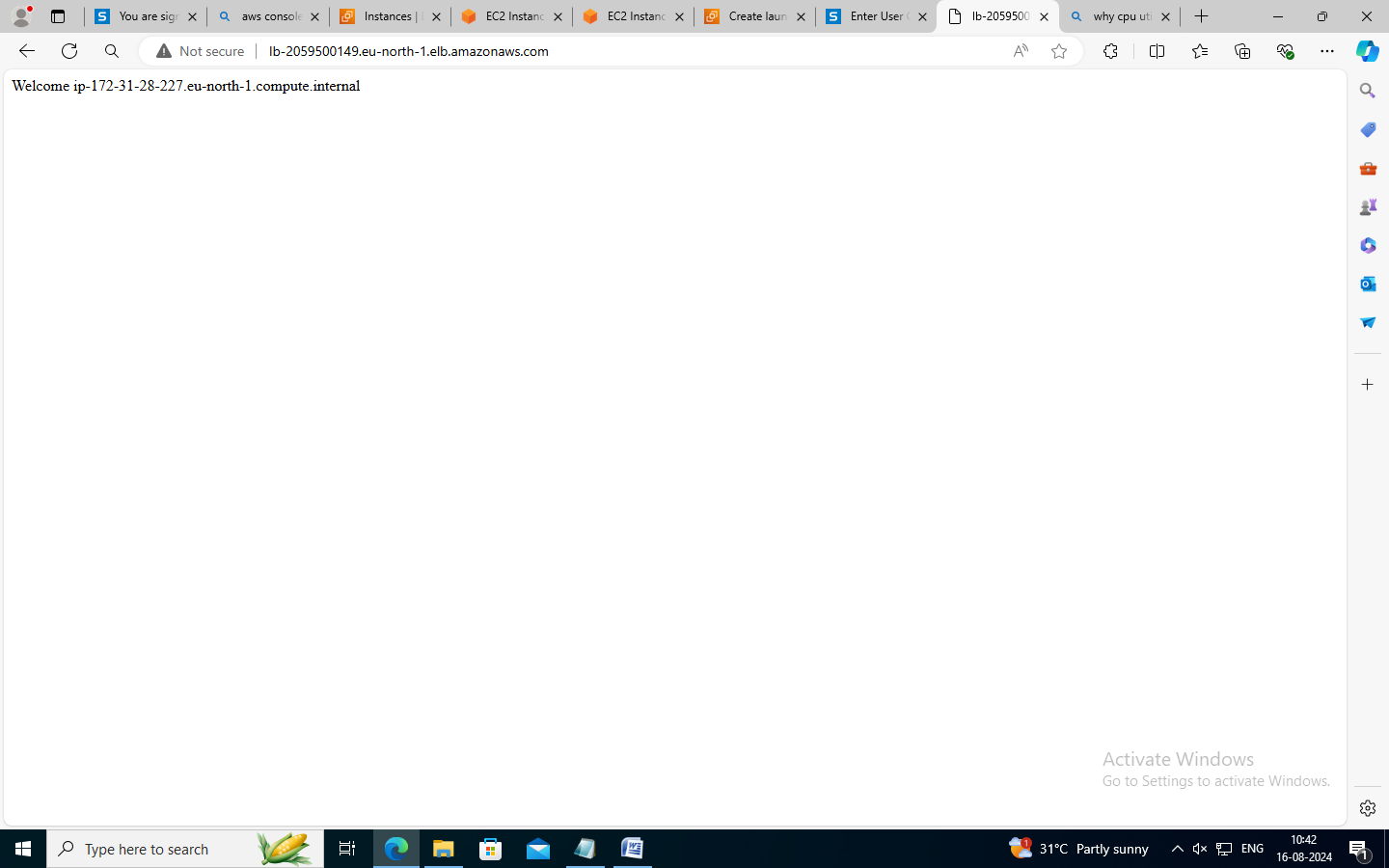
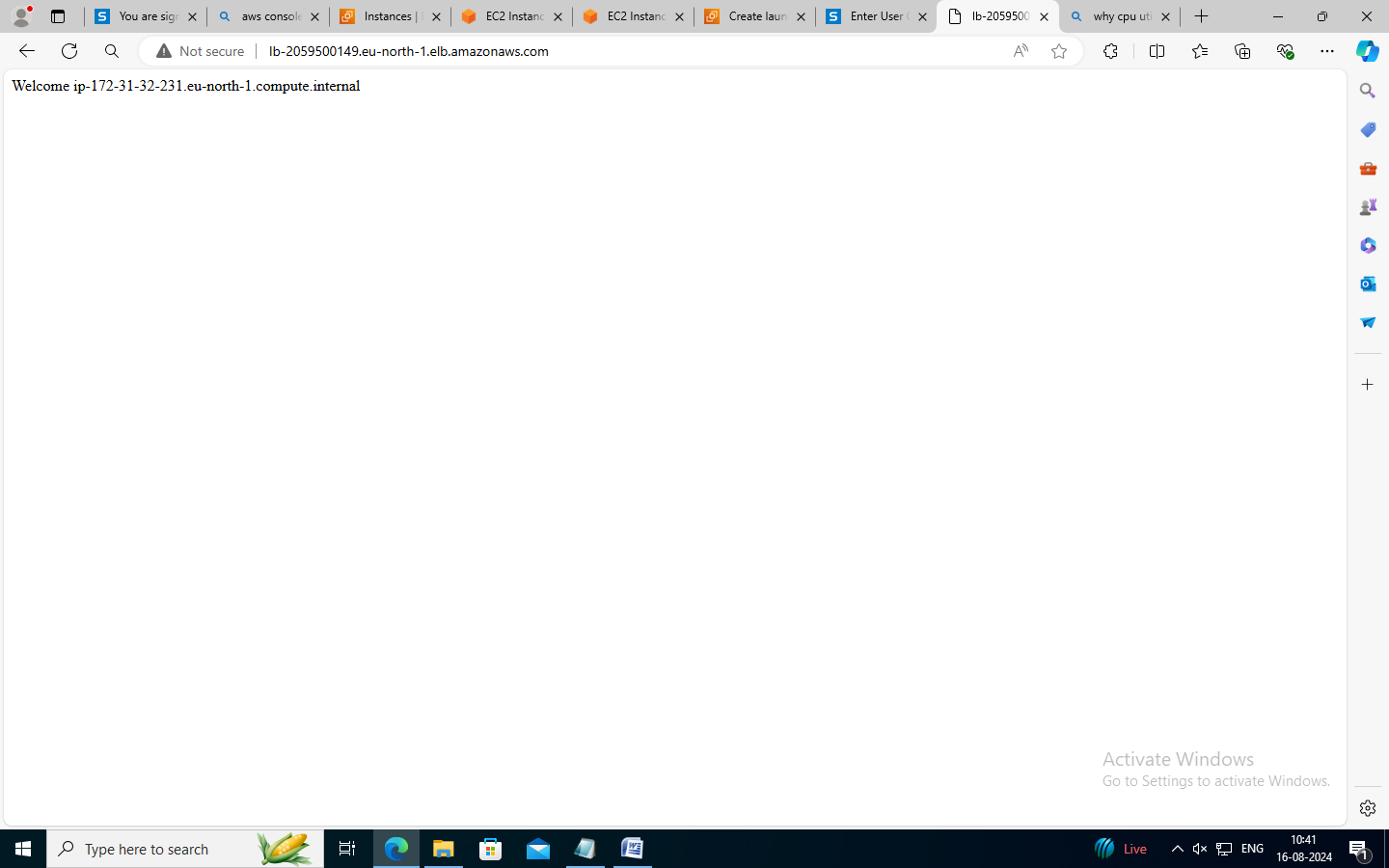
**CPU Utilization Increased**

****

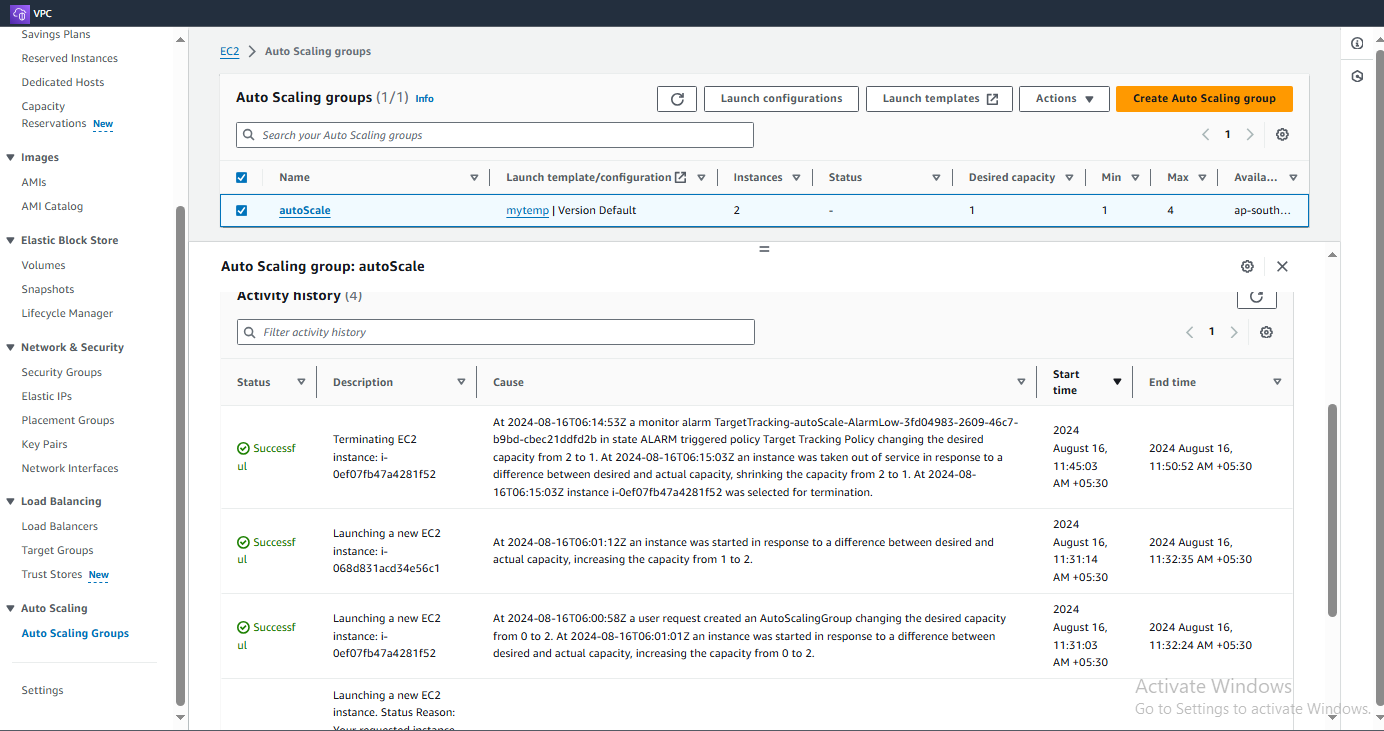
****

**4 Instances**

****

****

**After CPU Utilization is decreased**

****

**Conclusion**

Thus the Configuration of an Auto Scaling Group effectively demonstrates its ability to dynamically adjust to varying loads by scaling instances based on CPU utilization. Observing the scaling process under increased load highlights the system’s responsiveness and efficiency in maintaining application performance and availability has been developed and executed successfully.