**Project3:**

**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 logging 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-premise 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 CloudW

I have performed following steps to complete the project and the screen shots are added separately on screenshots tab and submitted.

**Step 1: Logged into AWS account and created a Auto Scaling Group**

**Step 2**: Launched a Template of Amazon Linux free tier eligible

**Step 3**: Launched Template with instance type t2.micro free tier eligible

**Step 4**: Created a key pair for the Template

**Step 5**: Created a new security group for the Template and added ssh and http in inbound firewall setting.

**Step 6**: Successfully created template

**Step 7**: In this step I have configured Network settings with subnets

**Step 8**: Attached a application load balancer to the template with Health check of target group

**Step 9**: Configured group size and scaling option with desired capacity of 3 units, minimum desired capacity of 2 and maximum capacity of 5

**Step 10**: Enabled auto scaling group and 3 instances are running now.

**Step 11**: Load balancer target group health check.

**Step 12**: Auto scaling is working properly as I deliberately terminated a instance auto scaling group has replaced it with new instance

**Step 13**: Have created dynamic scaling policy under ASG and given the criteria as avg cpu utilization of 60

**Step 14**: The dynamic scaling policy is captured under cloud watch it has created 2 alarms one is high alarm another is low alarm. Since there are no applications running in the instances so alarm low is triggered.

**Step 15**: Since we have given criteria as minimum 2 instances and maximum 5 instances during building auto scaling policy one instance have been terminated because of low usage of cpu and low usage alarm is triggered.