Auto Scaling Group Setup for Node.js Application

This document provides step-by-step instructions for setting up an Auto Scaling Group (ASG) for a Node.js application using Amazon Web Services (AWS). The configuration includes automatic scaling based on CPU utilization metrics, with a threshold of 70% CPU usage.

# 1. Set Up an EC2 Launch Template

Before creating the Auto Scaling Group, you need a Launch Template that defines the EC2 instance configuration that will be launched by the ASG.

## Steps:

1. Go to the EC2 Dashboard in the AWS Management Console.  
2. In the left navigation pane, under 'Instances', choose 'Launch Templates'.  
3. Click on 'Create Launch Template'.  
4. Fill out the following details:  
 - Launch template name: manhwa-app-template  
 - Version description: Initial version  
 - AMI ID: Choose the AMI ID of your Node.js application.  
 - Instance Type: Select the instance type (e.g., t2.micro, t3.medium).  
 - Key Pair: Select the existing key pair for SSH access.  
 - Security Groups: Select or create a security group that allows HTTP (port 80) and SSH access (port 22).  
 - Network settings: Choose the appropriate VPC and subnet.  
5. Click 'Create Launch Template'.

# 2. Create an Auto Scaling Group

Now, you’ll create the Auto Scaling Group, which uses the Launch Template to scale instances based on the CPU utilization metric.

## Steps:

1. Go to the EC2 Dashboard.  
2. In the left navigation pane, under 'Auto Scaling', click on 'Auto Scaling Groups'.  
3. Click on 'Create Auto Scaling group'.  
4. Auto Scaling group name: manhwa-app-asg  
5. Launch Template: Choose the template created earlier (manhwa-app-template).  
6. VPC and Subnets: Select the VPC and subnets in which you want to launch the instances.  
7. Load Balancing: (Optional) If you want to set up a Load Balancer, you can choose one here.

# 3. Configure Desired Capacity and Scaling Policies

You can now configure the minimum and maximum number of instances and create a policy for scaling based on CPU utilization.

## Steps:

1. Set the Desired Capacity (the number of instances to start with), e.g., 1.  
2. Set the Minimum Capacity to 1 (so at least 1 instance is always running).  
3. Set the Maximum Capacity to 3 (for example).  
4. In Scaling policies, select Target tracking scaling policy and configure it based on CPU utilization:  
 - Policy type: Target tracking scaling policy.  
 - Target value: Set this to 70 to target 70% CPU utilization.  
 - Predefined metric type: Choose Average CPU Utilization.  
 - Instance warm-up time: Set this to 300 seconds (5 minutes).  
5. Click Next and proceed with reviewing the settings.

# 4. Add Monitoring

To monitor CPU utilization effectively, you can use Amazon CloudWatch to track the performance of the Auto Scaling Group.

## Steps:

1. Go to the CloudWatch Dashboard.  
2. In the left navigation pane, under 'Alarms', click 'Create Alarm'.  
3. Click 'Select metric', and then choose EC2 → Per-Instance Metrics → CPUUtilization.  
4. Set up an alarm that will trigger whenever the CPU utilization exceeds 70%. You can use this to notify you when the scaling occurs.

# 5. Enable Auto Scaling Group Monitoring

Ensure that monitoring is enabled for the Auto Scaling Group to trigger the scaling based on CPU utilization.

## Steps:

1. In the Auto Scaling Group settings, under Monitoring, enable Group metrics collection.  
 This will allow metrics such as GroupMinSize, GroupMaxSize, and GroupInServiceInstances to be sent to CloudWatch for tracking and scaling decisions.

# 6. Test Auto Scaling Group

To verify that the Auto Scaling Group scales based on CPU load:  
1. Simulate high CPU load by running CPU-intensive tasks on the EC2 instance.  
2. Monitor the instances in CloudWatch and the Auto Scaling Group dashboard.  
3. As the CPU utilization reaches 70%, a new instance should be automatically launched to distribute the load.

# Summary of Auto Scaling Configuration:

- Launch Template: manhwa-app-template  
- Auto Scaling Group: manhwa-app-asg  
- Desired Capacity: 1  
- Minimum Capacity: 1  
- Maximum Capacity: 3  
- Scaling Policy: Target tracking for CPU utilization at 70%.  
- Monitoring: CloudWatch alarm for CPUUtilization > 70%.