Hands-On with AWS Auto Scaling

Lab Steps

Step 1: Launch Template Creation

In AWS Auto Scaling, a Launch Template is a configuration that defines how Amazon EC2 instances should be launched in your Auto Scaling group.

A launch template contains all the settings needed to launch an EC2 instance, such as: AMI, Instance type, key pair, SG, EBS, Network settings and User data – for bootstrapping (e.g., install apps on startup).

.

Create a Launch Template:

- Create an inbound security Group that allows traffic from SSH(Port 20) and HTTP(Port 80)
    - Navigate to EC2 Dashboard→Under "Instances," select **Launch Templates**.-->Click **Create Launch Template**.-->Provide a **Template Name** (e.g., AutoScaling-LT).-->Select an AMI (e.g., Amazon Linux 2).-->Choose an instance type (e.g., t2.micro).
    - Configure a key pair for SSH access.-->Add a security group that allows HTTP (port 80) and SSH (port 22) access. You can use an existing SG→Leave the rest as default and **Create Template**.

Step 2: Create an Auto Scaling Group (ASG)

- Go to the **Auto Scaling** section under "EC2."-->**Create an Auto Scaling Group**:→Click **Create Auto Scaling group**.
    - Provide a name (e.g., My-ASG).-->Select the Launch Template you created earlier.-->next→Select the default VPC and Subnets for your ASG.
- **Configure Instance Settings**:
    - Set the desired capacity (e.g., 2).-->Minimum instances: 1, Maximum instances: 4.
    - Scaling options→Target scaling(CPU Utilization, 50%)
- **Configure Health Checks**: Use EC2 health checks for simplicity.
- **Attach a Load Balancer**:
    - (Optional) If you have a Load Balancer, attach it here. Skip for this basic lab.

2. **Review and Create**:
    - Confirm the settings and click **Create Auto Scaling Group**.
    - Once created, click on Autoscaling group name→ click on activity tab to see 2 instances(desired) created.

Step 3: Simulate and Test Auto Scaling

- **Increase Load**:

    - Navigate to EC2→click on the instance ID of one of the instances→ click on the Monitoring tab. Check the CPU Utilization on CloudWatch.
    - Connect to one of the EC2 instances. Use a stress tool to generate CPU load by running **sudo yum install -y stress**→simulates a CPU-intensive workload on 2 CPU cores for 1 minute by running**: stress --cpu 2 --timeout 60**
- **Monitor Scaling**:
- click on the instance ID of the instance→ click on the Monitoring tab. Check the CPU Utilization on CloudWatch
    - Go to the ASG dashboard and observe new instances being launched once CPU utilization go over the maximum threshold.
    - Check CloudWatch metrics to see scaling triggers.
- **Decrease Load**:
    - Stop the stress test and observe the ASG scaling down instances after a few minutes.

Step 5: Clean Up Resources

- Delete the Auto Scaling Group.=→Delete the Launch Template.-->Terminate any remaining EC2 instances.--> Delete related CloudWatch alarms.

Expected Outcomes

1. Instances scale out when load increases (e.g., high CPU utilization).
2. Instances scale in when the load decreases.

Hands-On with Application Load Balancer

Objective

To create and configure an **Application Load Balancer (ALB)** to route traffic to multiple EC2 instances based on path-based routing.

Steps

1. Launch EC2 Instances

- Navigate to **EC2** and launch **two EC2 instances** in the same VPC and Availability Zone:
  Security Group should allow traffic from port 22(SSH) and port80(HTTP)
  Install a web server on both instances:using the following command:
  sudo yum update -y
  sudo yum install -y httpd
  sudo systemctl start httpd

sudo systemctl enable httpd

Connect to each instance and **Create the /var/www/html/ Directory using :**

sudo mkdir -p /var/www/html
(The path of the directory you want to create. This is commonly the **default root directory for web servers** like Apache or NGINX.)

- Run :
- echo "Welcome to Server 1" | sudo tee /var/www/html/index.html
  On the first instance: and
  echo "Welcome to Server 2" | sudo tee /var/www/html/index.html on the second instance

  NB: tee writes the input to the file (index.html) with elevated permissions (thanks to sudo).

- **Test in Browser;** Copy the public IP addresses of the first instance and paste it into a new browser tab. You should see the text "Welcome to Server 1"
  Repeat same on the second instance by copying its public IP into a new browser. You should see the text "Welcome to Server 2"

2. Create a Target Group

- Navigate to **Target Groups** under **Load Balancing**.-->Click **Create target group** and choose:
  - **Target type**: Instances.
  - **Protocol**: HTTP.
  - **Port:** 80.
  - **Health check protocol**: HTTP.
  - **Health check path**: /.
  - Register the two EC2 instances by Selecting both instances and click **Include as pending**.--> Create Target Group.

3. Create the Application Load Balancer

- Navigate to **Load Balancers** under **Load Balancing**.-->Click **Create Load Balancer** and choose **Application Load Balancer**.-->Configure the ALB:
  - **Name**: Eg., My-ALB.
  - Scheme: Internet-facing.
  - IP Address Type: IPv4.
  - **Listeners**: Add a listener for HTTP on port 80.
  - **Availability Zones**: Select the default VPC and the subnets.

- Configure **Security Groups**: delete the default Security group and select the Security Group you created for the instances (security group allowing inbound traffic on port 80 and 22.)
- Configure **Routing**: Choose the target group you created earlier→Review and create the ALB.

4. Test the Load Balancer

- Once the ALB is active, copy its **DNS name** from the ALB dashboard. -->Open a browser and paste the DNS name:
- You will see "Welcome to Server 1" or "Welcome to Server 2" as the ALB distributes traffic.

5. Clean Up

- Delete the ALB, target groups, and EC2 instances to avoid unnecessary charges.