Assignment 4 – ALB and ASG

Updated demo recording: ELB_ASG_Midterm.mp4

Load balancers are powerful services that can handle millions of transactions per second and distribute traffic across servers. They also play a big role in reliability by sending requests only to healthy servers. Furthermore, we can utilize auto-scaling groups with a load balancer (ALB & NLB) and virtual machines (EC2) to automatically scale the application.

ACCEPTANCE CRITERIA – Include the following in the PDF:

- Screenshot of healthy instances in both TGs of ALB and NLB.
- Screenshot of the Activity tab in the ASG.

Task 1. Run web servers behind ALB

- Just like lab 1, create a web app with a custom HTML "I am a server one" in "us-east-1a" AZ.
- Create the second web app with a custom HTML "I am a server two" in "us-east-1b" AZ.
- Create an ALB and register those 2 instances. When creating the ALB, select "us-east-1a" and "us-east-1b" for nodes.
- The reason I specified the AZs in the spec is, that the load balancer can route traffic only to the instances where its node is available. Let's say you created 2 instances in AZ 1a and 1b. But your load balancer nodes are created in AZ 1d and 1f, then the load balancer cannot route the requests to the servers and you will see an "**unused**" state in the target group.

For this task, you can open up port 80 to all (0.0.0.0/0). That makes debugging and development much easier. Many students get unhealthy in the target group of ALB. That is mainly because apps are not properly running on the EC2 instance. When you have port 80 open to all (0.0.0.0/0), you can directly test if the web app is working properly with the browser.

You can tighten up the security posture by allowing access only from the ALB. The step-by-step instructions do that. AWS engineers upgrade the services every day. The step-by-step instructions tend to be outdated. Just refer them but don't follow them blindly.

Your app running on servers across two data centers (AZs) means, your app is highly available.

Task 2. Run web servers behind NLB

This task is very similar to task 1, ALB. The only difference is, to change the Target Group protocol to **TCP** (Layer 4)**,** not HTTP (Layer 7). Remember, ALB works at layer 7 whereas NLB works at layer 4.

Task 3. Run the web server in ASG

Instead of manually adding & removing servers, let's scale the number of servers based on the demand or CPU utilization metric.

- Create a launch template. It allows you to select AMI whereas the launch configuration requires you to enter AMI ID.
    - Give it a name
    - Select the Amazon Linux AMI in the Quickstart.
    - Select an instance type, t2.micro.

- Select an SG that allows access from the internet on port 80.
- Expand advanced. Enter the following in the user data.

It starts web apps automatically when the server starts. So, you don't have to do anything manually when scaling out.

```
#!/bin/bash
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello YourName from $(hostname -f)</h1>" > /var/www/html/index.html
```

If the webservers are unhealthy, just remember the lab1 and previous tasks. Make sure the web app is running and port 80 is open in the servers' security groups.

- Create the Auto Scaling Group.
  - Select launch template/configuration.
  - Select AZs (Subnets). That is where your instances launched.
  - Click on attach to an existing load balancer and select the default TG of the ALB.
  - Select ELB in the health checks panel.
  - Set desired, min, and max capacity. Set a target tracking scale policy.

You can mimic the high CPU utilization with the "stress" library to test scaling out behavior.

Task 4. Clean up ALB, NLB, EC2 instances. They cost huge amounts.

Troubleshooting instructions unhealthy status in the Target Group

- Change the EC2 instances' Security Groups to allow all transactions from the internet (0.0.0.0) on the HTTP (80) port just like the Lab 1.
- Grab the EC2 instance's public IP address and put it in the browser to check if the web server is up or not. If it is not returning anything, that is the reason why it is unhealthy.
- You must make sure that there is a custom HTML in the /var/www/html directory. Otherwise, the default Apache page returns 403 status even if it seems like it is working in the browser. It could be the cause because the target group is expecting 200 but the web servers are returning 403. Therefore, it marks that as unhealthy.
- If the web page is not responding, simply just do what you did in Lab 1. SSH into the server. Open up the port 22 (SSH) to all (0.0.0.0) and use the EC2 connect in the AWS Console. "SSM agent is not online" is most likely caused by not associating the IAM role with the SSM policy with the EC2 instance or AWS Academy hiccup.
- Once you are in the instance, install the HTTPD if it has not been installed. Define a custom HTML in the /var/www/html. Start the Apache web server. Then reregister the instance in the Target Group.