Production level Load Balancing using AWS ALB with Auto Scaling

Application Load Balancer (ALB) in AWS

The **Application Load Balancer** is part of AWS Elastic Load Balancing (ELB) and operates at the **Application Layer (Layer 7)** of the OSI model. It's designed to handle advanced routing for HTTP and HTTPS traffic, making it ideal for modern web applications, microservices, and containerized workloads.

Key Features:

1. **Content-Based Routing**
   ALB can route traffic based on the content of the request. For example, you can route requests to /api/* to one set of targets and /images/* to another. This is called **path-based routing**.

2. **Host-Based Routing**
   You can route requests based on the domain name (host header). For instance, requests for app.example.com go to one target group, and blog.example.com go to another.

3. **Target Groups**
   ALB sends traffic to target groups. Each target group can have multiple EC2 instances, ECS containers, IP addresses, or Lambda functions. Targets within a group are registered with a specific protocol and port.

4. **Health Checks**
   ALB performs periodic health checks on targets. If a target fails the health check, ALB stops routing traffic to it until it becomes healthy again.

5. **SSL Termination**
   ALB supports SSL/TLS termination. You can upload your SSL certificate to AWS Certificate Manager (ACM) and let ALB handle HTTPS decryption, reducing the burden on backend servers.

6. **Integration with Auto Scaling**
   ALB automatically updates target groups as Auto Scaling launches or terminates instances. This ensures only healthy and available instances receive traffic.

7. **Sticky Sessions (Session Affinity)**
   You can enable sticky sessions to ensure that a user's session stays on the same backend instance during its lifecycle.

8. **WebSocket Support**
   ALB supports WebSockets, which is useful for real-time applications like chat apps.

9. **Logging and Monitoring**
   ALB provides access logs, CloudWatch metrics, and AWS X-Ray integration for request tracing.

Auto Scaling in AWS

Auto Scaling is a service that automatically adjusts the number of Amazon EC2 instances in a group based on traffic or demand. It ensures that your application has the right amount of compute resources at any time, which helps with performance and cost optimization.

How Auto Scaling Works:

You start by creating an **Auto Scaling Group (ASG)**. This group manages a fleet of EC2 instances based on rules and policies you define. You also create a **Launch Template** or **Launch Configuration** that tells AWS how to launch these instances (what AMI to use, instance type, security groups, etc.).

Types of Scaling:

1. **Dynamic Scaling**
   This is triggered automatically based on real-time metrics. For example, if CPU usage exceeds 70% for more than 5 minutes, AWS can add a new EC2 instance.
2. **Scheduled Scaling**
   This allows you to scale based on predictable usage patterns. For example, if your application receives more traffic every weekday from 9 AM to 5 PM, you can schedule the scaling accordingly.
3. **Predictive Scaling**
   This uses machine learning to analyze historical usage patterns and automatically provision resources in advance. It helps in anticipating demand spikes and avoiding latency.

Scaling Policies:

Scaling decisions are governed by policies. Common types are:

- **Target Tracking Scaling**: Maintains a metric at a specific target value (e.g., keep average CPU at 50%).
- **Step Scaling**: Responds to larger metric changes with more aggressive scaling.
- **Simple Scaling**: Adds or removes a fixed number of instances when a metric crosses a threshold.

Health Checks:

Auto Scaling uses health checks (from EC2 or ELB) to monitor instances. If an instance becomes unhealthy, Auto Scaling automatically terminates and replaces it.

Lifecycle Hooks:

Lifecycle hooks allow you to perform custom actions during instance launch or termination. For example, you can run configuration scripts or deregister from external systems before termination.

ALB + Auto Scaling Together

A common AWS architecture pattern is to place an Application Load Balancer in front of an Auto Scaling Group. The ALB receives traffic and distributes it to healthy EC2 instances managed by the Auto Scaling Group. This combination ensures high availability, fault tolerance, and cost efficiency.

When traffic increases, the Auto Scaling Group adds more EC2 instances. The ALB automatically registers these new instances and starts routing traffic to them. When demand decreases, instances are terminated, and ALB deregisters them automatically. This setup is widely used for web applications, APIs, microservices, and container-based apps.

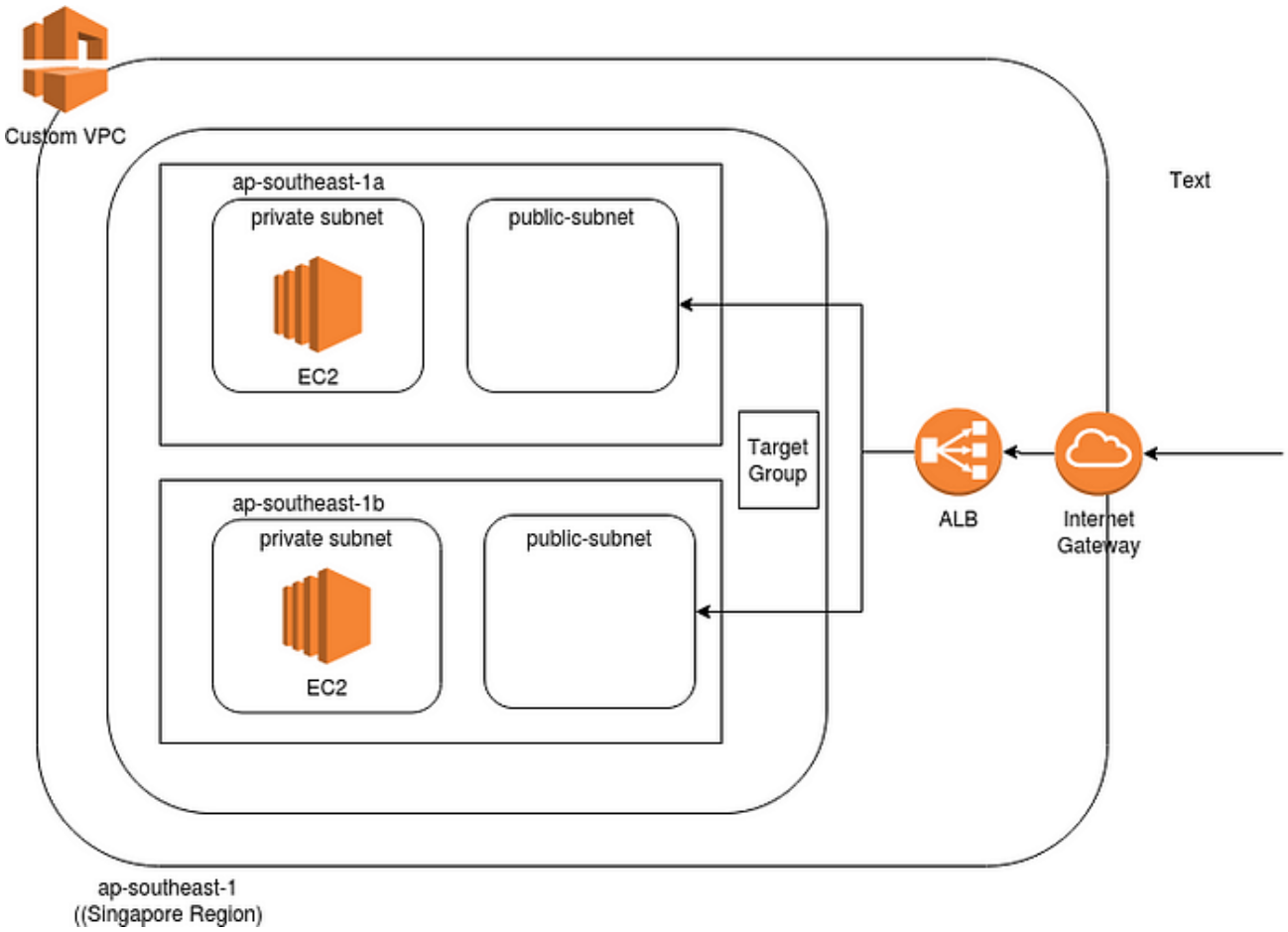

Figure 1 — Load Balancing with Auto Scaling using an AWS ALB

Figure 1 explains the deployment architecture.

- A Custom VPC with two Availability Zones for High Availability
- Each Availability Zone has a public subnet and a private subnet.
- There are two EC2 instances, which are in private subnets and the Application Load Balancer is attached to two public subnets as expected.
- ALB is coupled with Auto Scaling
- There is one Target Group connecting both the EC2 instances
- EC2 Apache installation happens with AWS SSM

Steps

Step 1: Create a Custom VPC

Create the Custom VPC with two public subnets and two private subnets spanned in two Availability Zones (See Figure 1).

Step 2: Create a Target Group

(See Figure 2 and 3)

Target Group Name: <My_Target_Group>
Target Type: <Instance>
Protocol: <HTTP>
Port: <80>
VPC: <Select the Custom VPC>

Health Check Settings:
Protocol: <HTTP>
Path: </index.html>

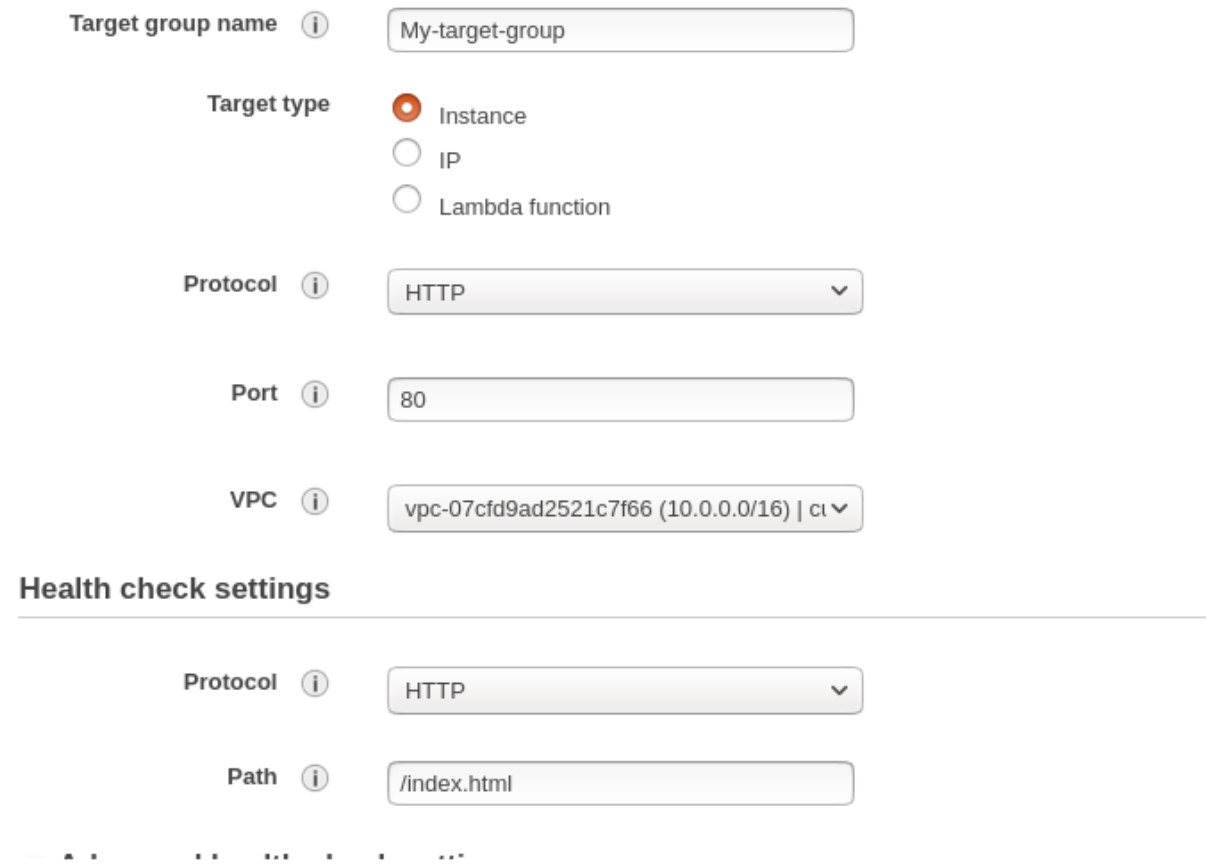

Figure 2 — Creating the Target Group

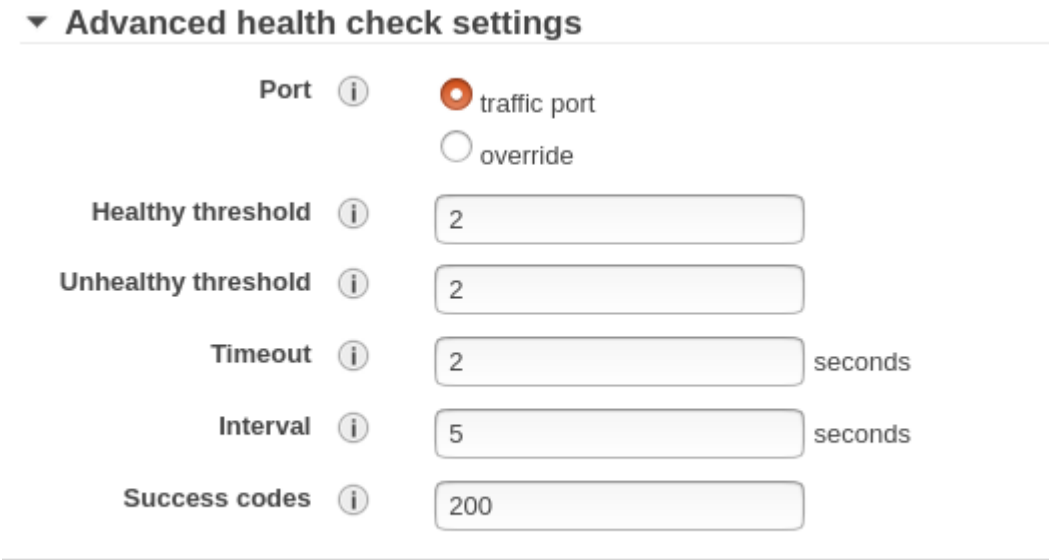

Figure 3 — Creating the Target Group (cont)

Step 3: Create an Application Load Balancer (ALB)

Name: <Enter a name> i.e. MyALB
Scheme: Select Internet Facing

IP address type : ipv4
Listener Configurations: Open port 80 (HTTP)

Now you are required to select the two public subnets in each Availability Zone. You are selecting public subnets here because by default Load Balancers are attached to public subnets (See Figure 4).

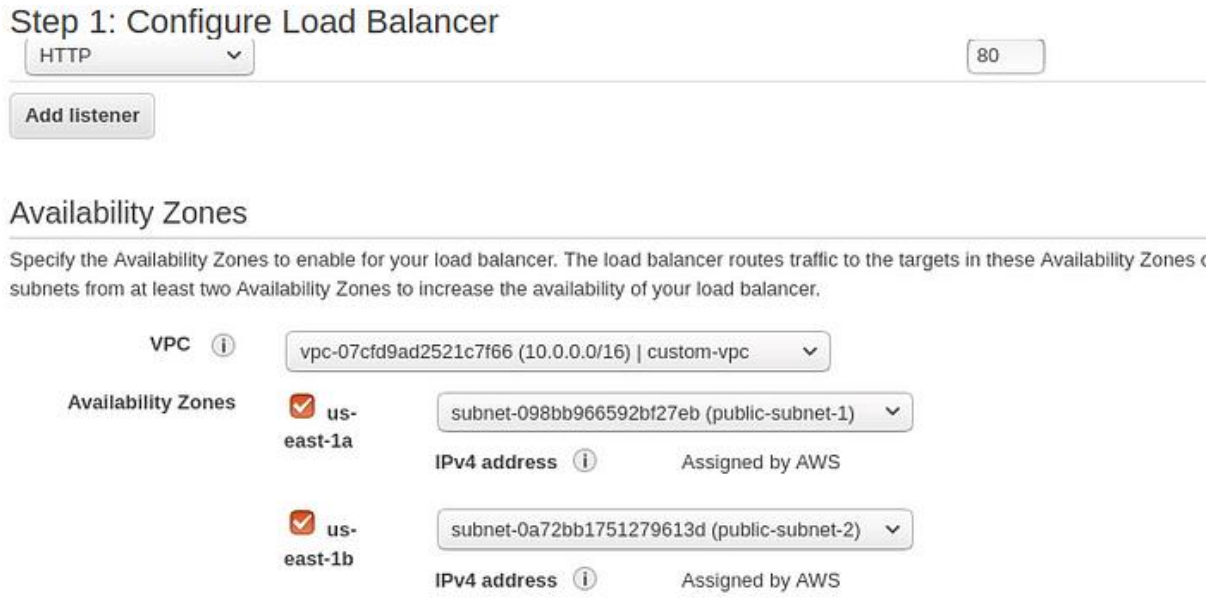

Figure 4 : Attaching public subnets to ALB

Now, Create a New Security Group for the Load Balancer (i.e. myalb-sg)
Add HTTP Port 80.

[P.Note: You are required to attach the Default Custom VPC Security Group to ALB once it is created. And the Default Custom VPC Security Group also need to have port 80(HTTP) opened] (See Figure 5 and 6).

Zoom image will be displayed

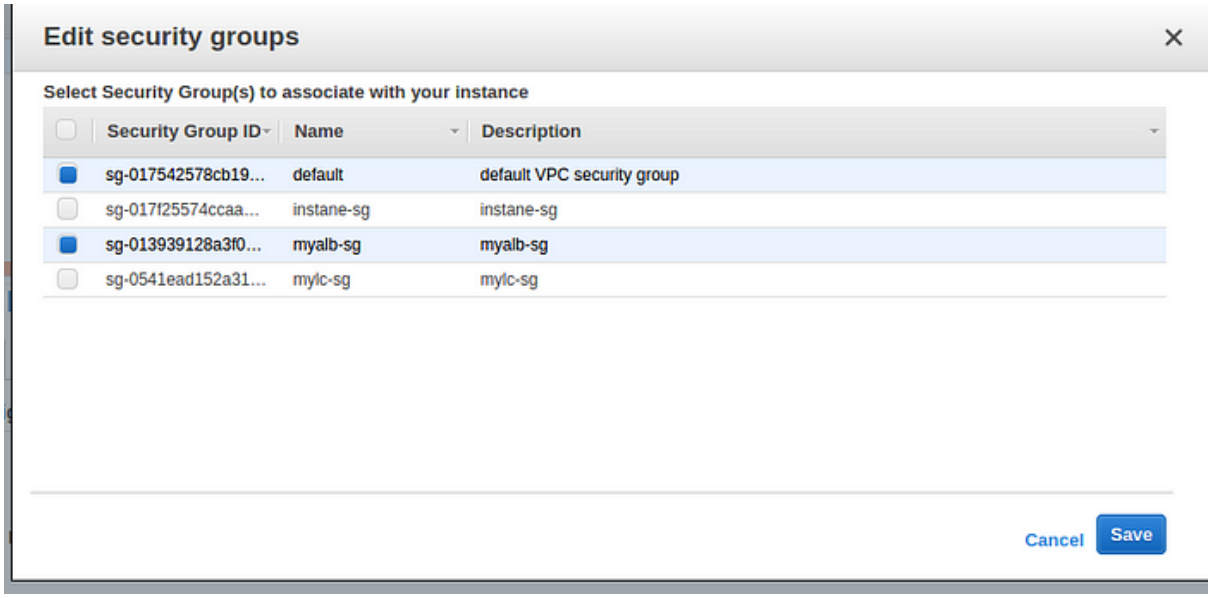

Figure 5 — ALB Security Groups

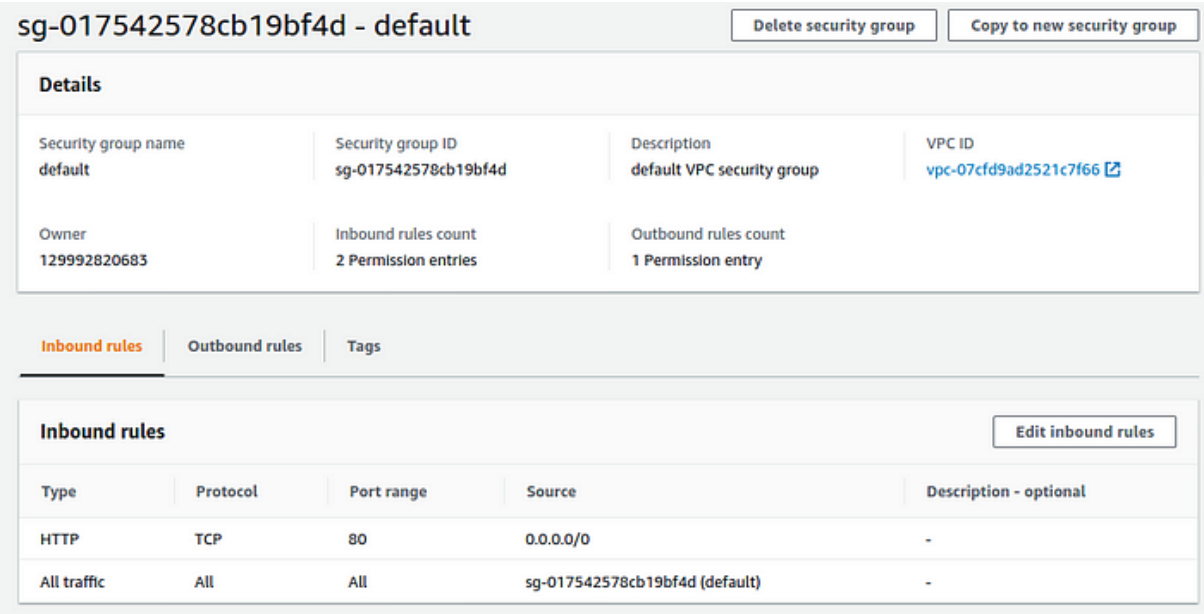

Figure 6 — ALB Default VPC Security Group port settings

Click Next and select the Target Group that you have already created in step 2. That will populate the target values to the rest of page.

Click Next to create the Load Balancer.

Now it is the time to enable Auto Scaling to the ALB. In order to create Auto Scaling, you are required to create a Launch Configuration / Launch Template and then create a Auto Scaling Group (ASG). Here we stick to Launch Configuration setup for brevity.

Step 4: Create a Launch Configuration

[P.Note: Before creating the Launch Configuration, it is recommended to create the Security Group since it has a bug if you create while creating the Launch Configurations. This happens only when you are dealing with a Custom VPC. If you create the Security Group while you are creating the LC, it assigns the SG to the Default VPC SG not to the Custom VPC SG. Therefore, create a SG and add port (80)]

Once you are done creating the SG, you may proceed creating the LC.

Go to EC2 -> Launch Configurations -> Click *Create Launch Configuration*

Select the instance type (Amazon Linux t2.micro) and fill the following parameters

Name: <Enter a Launch Configuration name> e.g: MyLC

Purchasing options: <deselect the option>

IAM Role: <Select SSM Full Access IAM Role> e.g. EC2SSMAccess

[This is required since we are using SSM for Apache installations and EC2 instances are created with the Launch Configuration]

Monitoring: <deselect the option>

Click Next and select the SG that you have created before to this.

Then click *Confirm* to create the LC.

Step 5: Create the Auto Scaling Group (ASG)

Go to EC2 -> Select *Auto Scaling* -> Click *Create Auto Scaling Group* button

Select *Launch Configuration* and select the LC that you have created under Step 4

Click Next

Enter the ASG name — e.g. MyASG

Select the Group Size — Key in 2 here (This is the desired ASG capacity)

Select the VPC (Custom VPC)

Select Subnets –Select two private subnets in this example, where you have your targets (EC2)

Click Next and Select "Use Scaling Policies to adjust the capacity of this group" — Select a range that can scale to. (i.e. between 2 and 4)

Key in the Scale Group Size information (Metric Type = CPU Utilization, Target Value = 80)

Warm up time after an instance creation — Probably 100 seconds.

Add a Notification — i.e. SNS Notification (optional , good to have)

If all okay, now you should be able to see the ASG created with two EC2 instances.

Step 5: Install Apache in EC2 instances using AWS SSM

Go to AWS SSM -> Select Documents -> Select Document *Owned by me* -> Select *installApache* Document (See Figure 7) and click to select it -> Use *Run Command* to select the two EC2 instances -> Click *Run* button to execute

P.Note: I will be writing a separate blog on AWS SSM Documents in the future and until such time please have a look at the AWS SSM Document Run Command for more information [1].

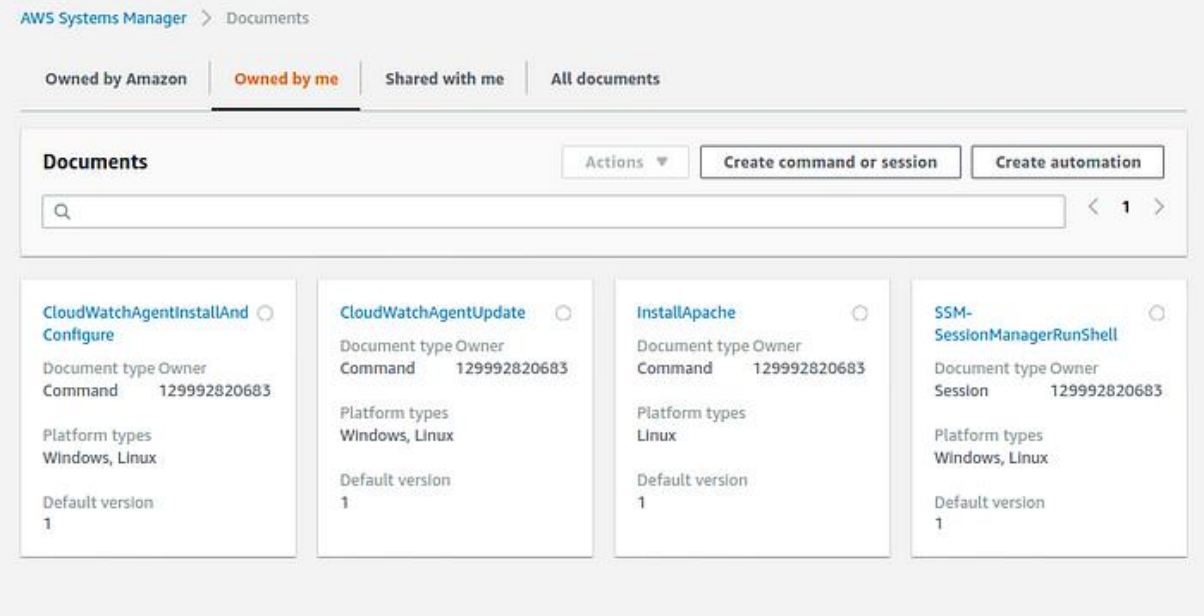

Figure 7 — installApache SSM Document

If all fine, Apache would be installed successfully on each EC2 instance.

Step 6: Attach EC2 instances to the ALB

Now after all above steps, time to add EC2 instances to ALB.

You are required to add EC2 instances to Target Group of ALB.

Go to EC2 -> Load Balancers -> Select the Load Balancer (MyALB) -> Select *Targets* tab -> Click *Edit* tab -> Add two EC2 instances to the Load Balancer

Zoom image will be displayed

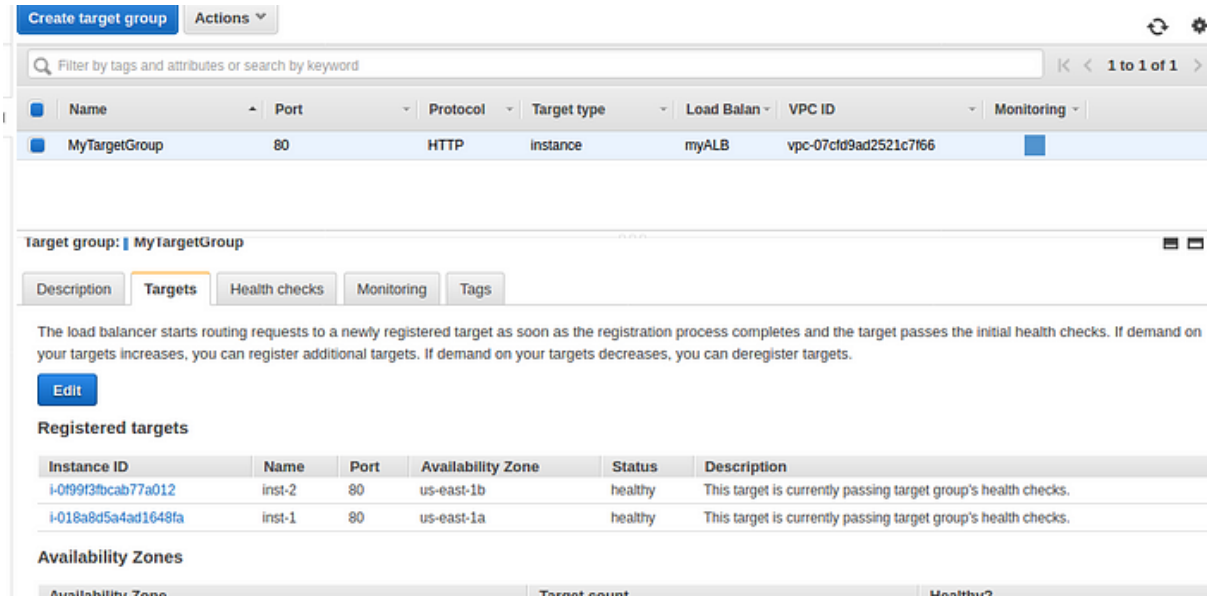

Figure 8 — Adding EC2 instances to the ALB Target Group

Stay for few seconds. You will see the *Status* as *healthy*.

Step 7: Test the Load Balancer

You can do a quick check to see whether the example works now.

Select ALB -> Click Description Tab -> Click the *Description* tab -> Copy (Ctrl +C) the DNS name (e.g. myALB-1767835550.us-east-1.elb.amazonaws.com)

Paste the DNS name on a web browser and see all ok. If you see a similar page like below (See Figure 9)

Zoom image will be displayed