Yash Mutatkar **Date : 29 October 2024**

Task : Set up Auto Scaling with Load Balancer

- Deploy an EC2-based application with an **Application Load Balancer (ALB)**.
- Configure **Auto Scaling** to launch instances based on CPU usage.
- Test scaling by generating artificial load.

---

1. Creating EC2 Instances

First things first, I launched EC2 instances using the AWS Management Console. I carefully selected an appropriate Amazon Machine Image (AMI) and instance type that would meet my application's needs. After setting up my key pair for SSH access, I configured the security groups to allow inbound HTTP traffic on port 80 from anywhere. This was crucial because I wanted to ensure that my instances could receive requests without any hiccups.

2. Setting Up the Application Load Balancer (ALB)

Next, I created an Application Load Balancer (ALB) to distribute incoming traffic across my EC2 instances. This would enhance the availability and fault tolerance of my application. I configured the ALB to listen for HTTP requests on port 80 and set up health checks to monitor the registered EC2 instances. This way, I could ensure that only healthy instances would serve traffic.

3. Configuring Target Groups

To make the ALB work effectively, I needed to create a target group. This involved grouping my EC2 instances so the ALB could route traffic to them. Initially, my target group had no instances registered, so I quickly registered the running EC2 instances. I also configured health checks to monitor their status, checking for successful responses to ensure they were ready to handle traffic.

4. Deploying the Application

Now came the exciting part: deploying my application! I SSH-ed into one of my EC2 instances using the key pair I had created earlier. After navigating to the Apache document root directory at `/var/www/html`, I replaced the default `index.html` file provided by Apache with my custom HTML content—a simple "Hello, World!, this is yash mutatkar" page. Once I updated the HTML file, I restarted the Apache server to ensure it served my new content.

5. Setting Up Auto Scaling

To prepare for varying traffic loads, I set up Auto Scaling. I created a launch template that specified the configuration for my EC2 instances, including the AMI, instance type, and security group. Then, I created an Auto Scaling group using this launch template and associated it with my ALB. I set the desired and maximum instance counts based on expected load, ensuring that I could handle increases in traffic. I configured scaling policies based on CPU utilization, which would automatically adjust the number of instances up or down depending on traffic demands. I also made sure the Auto Scaling group performed health checks to manage the instances effectively.

6. Verifying and Testing

With everything in place, I accessed my ALB using its DNS name provided by AWS. It was such a relief to see my custom HTML page displayed instead of the default Apache welcome page! To put my setup to the test, I simulated traffic to observe the Auto Scaling group in action. I watched as it increased the number of instances under heavy load and scaled back down when the demand decreased.

Step 1: Create a VPC

1. **Go to the VPC Console**:
   - Open the AWS Management Console.
   - Navigate to the **VPC** service.
2. **Create a VPC**:
   - Click on **Your VPCs** > **Create VPC**.
   - Choose **VPC only**.
   - Enter a **Name tag** (e.g., `yash-vpc`).
   - Set the **IPv4 CIDR block** to `10.0.0.0/16`.
   - Click **Create VPC**.

Step 2: Create Subnets

1. **Create Subnet A**:
   - Click on **Subnets** > **Create Subnet**.
   - Choose your VPC from the dropdown.
   - Enter a **Name tag** (e.g., `yash-subnet-a`).
   - Set the **Availability Zone** to `us-east-1a`.
   - Set the **IPv4 CIDR block** to `10.0.1.0/24`.
   - Click **Create**.
2. **Create Subnet B**:
   - Repeat the above steps, but set:
     - **Name tag** to `yash-subnet-b`.
     - **Availability Zone** to `us-east-1b`.
     - **IPv4 CIDR block** to `10.0.2.0/24`.

Step 3: Create a Security Group

1. **Go to Security Groups**:
   - In the VPC Console, click on **Security Groups** > **Create Security Group**.
   - Enter a **Name** (e.g., `yash-allow-http`).
   - Select your VPC.
2. **Configure Ingress Rule**:
   - Under the **Inbound rules** tab, click **Add rule**.
     - **Type**: HTTP

        - **Protocol**: TCP
        - **Port Range**: 80
        - **Source**: Anywhere (0.0.0.0/0)
    - Click **Save rules**.

Step 4: Create a Launch Template

1. **Go to the EC2 Console**:
    - Navigate to the **EC2** service.
2. **Create a Launch Template**:
    - Click on **Launch Templates** > **Create launch template**.
    - Enter a **Name** (e.g., `yash-launch-template`).
    - Choose an **AMI**: Search for the Ubuntu 24.04 AMI (e.g., `ami-0866a3c8686eaeeba`).
    - Select **Instance type**: `t2.small`.
    - Set **Key pair**: Select your existing key pair (`yashlinux`).
    - Scroll down to the **Advanced details** section, and in **User data**, add your script (you may need to encode it in Base64 if required).
    - Click **Create launch template**.

Step 5: Create a Target Group

1. **Go to the EC2 Console**:
    - Click on **Target Groups** under the **Load Balancing** section.
2. **Create Target Group**:
    - Click **Create target group**.
    - Choose **Target type**: `Instances`.
    - Set **Target group name** (e.g., `yash-target-group`).
    - Set **Protocol**: HTTP.
    - Set **Port**: 80.
    - Select your VPC.
    - Configure the **Health checks** (default settings are usually fine).
    - Click **Create**.

Step 6: Create an Application Load Balancer

1. **Go to Load Balancers**:
    - In the EC2 Console, click on **Load Balancers** > **Create Load Balancer**.
2. **Select Application Load Balancer**:
    - Choose **Application Load Balancer**.
    - Click **Create**.
3. **Configure Load Balancer**:
    - Name it (e.g., `yash-application-load-balancer`).
    - Choose **Scheme**: Internet-facing.

    - Select **IP address type**: IPv4.
    - Add your subnets (`yash-subnet-a` and `yash-subnet-b`).
4. **Configure Security Groups**:
    - Choose the security group you created earlier (`yash-allow-http`).
5. **Configure Listeners**:
    - For the **Listener**: Set **Protocol** to HTTP and **Port** to 80.
    - Click **Next**.
6. **Configure Target Group**:
    - Select the target group you created earlier (`yash-target-group`).
    - Click **Next**.
7. **Review and Create**:
    - Review your settings and click **Create**.

Step 7: Create an Auto Scaling Group

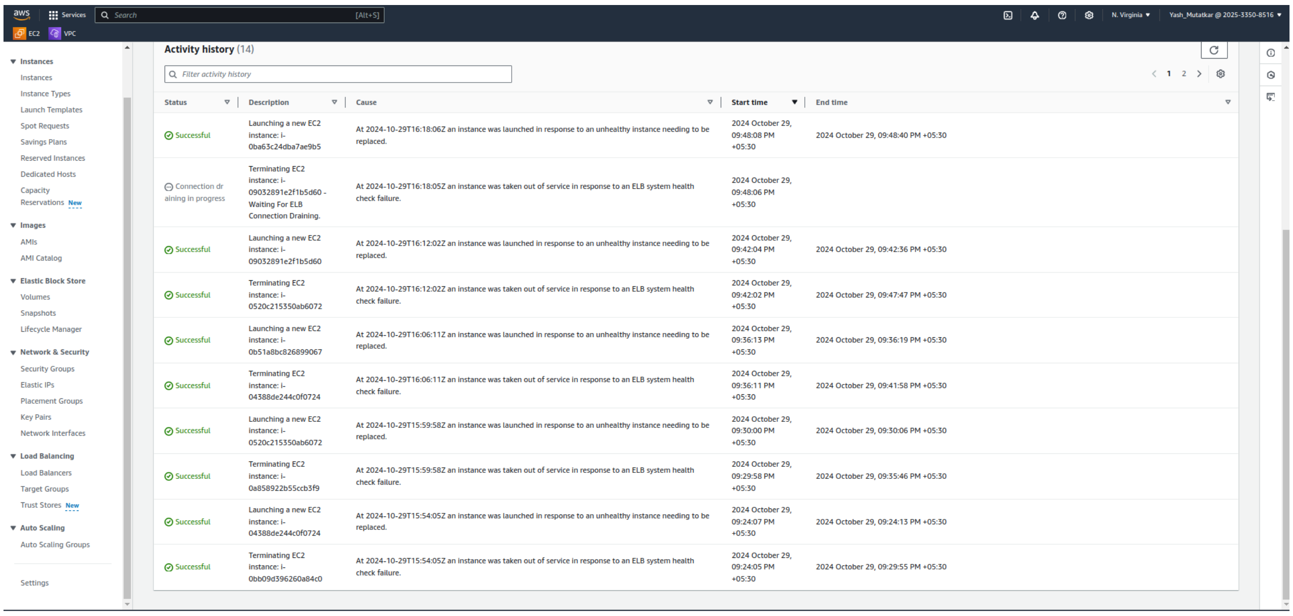
1. **Go to Auto Scaling Groups**:
    - In the EC2 Console, click on **Auto Scaling Groups**.
2. **Create Auto Scaling Group**:
    - Click **Create Auto Scaling group**.
    - Enter a name (e.g., `yash-asg`).
    - Choose the **Launch Template** you created earlier.
    - Click **Next**.
3. **Configure Settings**:
    - Set **Desired capacity**: 2, **Minimum capacity**: 1, and **Maximum capacity**: 5.
    - Choose the subnets you created earlier.
4. **Configure Load Balancer**:
    - In the **Load Balancing** section, select **Attach to a load balancer**.
    - Select the Application Load Balancer and the target group you created.
    - Click **Next**.
5. **Configure Scaling Policies** (optional):
    - Set scaling policies based on CPU utilization or other metrics if needed.
6. **Review and Create**:
    - Review the settings and click **Create Auto Scaling group**.

Step 8: Test the Load Balancer

1. **Get Load Balancer URL**:
    - Go back to the **Load Balancers** section.
    - Copy the DNS name of your load balancer.
2. **Access the Load Balancer**:

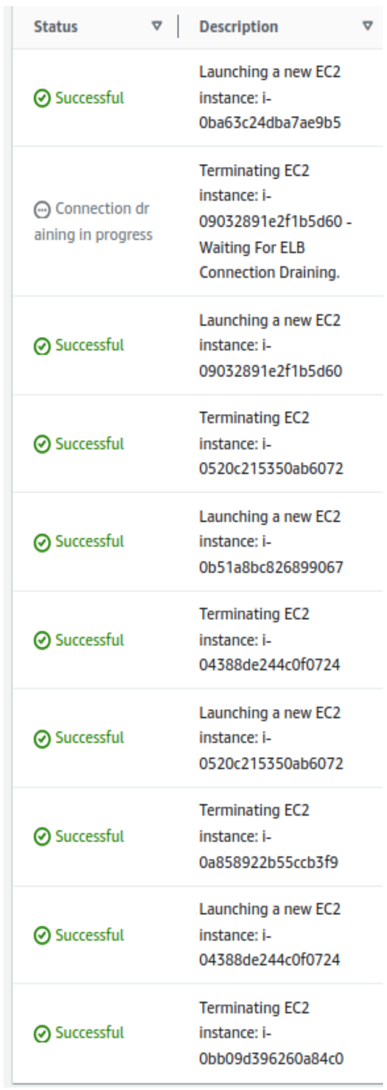
- Open a web browser and paste the DNS name. You should see your application responding.

Autoscalling in Action



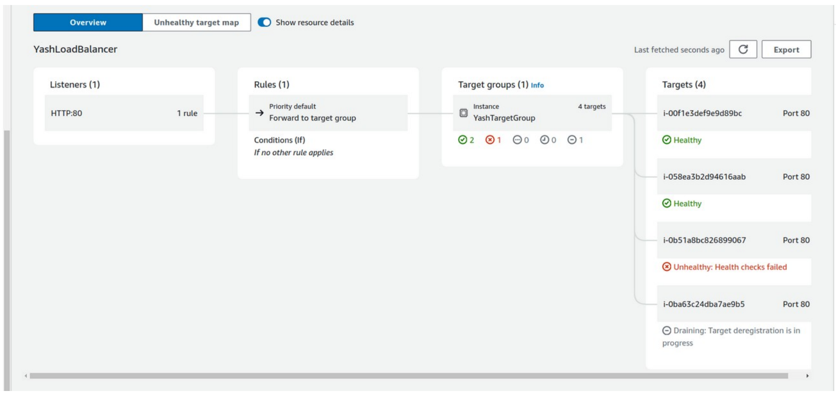
Activity history (14)

Status	Description	Cause	Start time	End time
Successful	Launching a new EC2 instance: i-0ba63c24dba7ae9b5	At 2024-10-29T16:18:06Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 October 29, 09:48:08 PM +05:30	2024 October 29, 09:48:40 PM +05:30
Connection draining in progress	Terminating EC2 instance: i-09032891e2f1b5d60 - Waiting For ELB Connection Draining.	At 2024-10-29T16:18:05Z an instance was taken out of service in response to an ELB system health check failure.	2024 October 29, 09:48:06 PM +05:30	
Successful	Launching a new EC2 instance: i-09032891e2f1b5d60	At 2024-10-29T16:12:02Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 October 29, 09:42:04 PM +05:30	2024 October 29, 09:42:36 PM +05:30
Successful	Terminating EC2 instance: i-0520c215350ab6072	At 2024-10-29T16:12:02Z an instance was taken out of service in response to an ELB system health check failure.	2024 October 29, 09:42:02 PM +05:30	2024 October 29, 09:47:47 PM +05:30
Successful	Launching a new EC2 instance: i-0b51a8bc826899067	At 2024-10-29T16:06:11Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 October 29, 09:36:13 PM +05:30	2024 October 29, 09:36:19 PM +05:30
Successful	Terminating EC2 instance: i-04388de244c0f0724	At 2024-10-29T16:06:11Z an instance was taken out of service in response to an ELB system health check failure.	2024 October 29, 09:36:11 PM +05:30	2024 October 29, 09:41:58 PM +05:30
Successful	Launching a new EC2 instance: i-0520c215350ab6072	At 2024-10-29T15:59:58Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 October 29, 09:30:00 PM +05:30	2024 October 29, 09:30:06 PM +05:30
Successful	Terminating EC2 instance: i-0a858922b55ccb3f9	At 2024-10-29T15:59:58Z an instance was taken out of service in response to an ELB system health check failure.	2024 October 29, 09:29:58 PM +05:30	2024 October 29, 09:35:46 PM +05:30
Successful	Launching a new EC2 instance: i-04388de244c0f0724	At 2024-10-29T15:54:05Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 October 29, 09:24:07 PM +05:30	2024 October 29, 09:24:13 PM +05:30
Successful	Terminating EC2 instance: i-0bb09d396260a84c0	At 2024-10-29T15:54:05Z an instance was taken out of service in response to an ELB system health check failure.	2024 October 29, 09:24:05 PM +05:30	2024 October 29, 09:29:55 PM +05:30

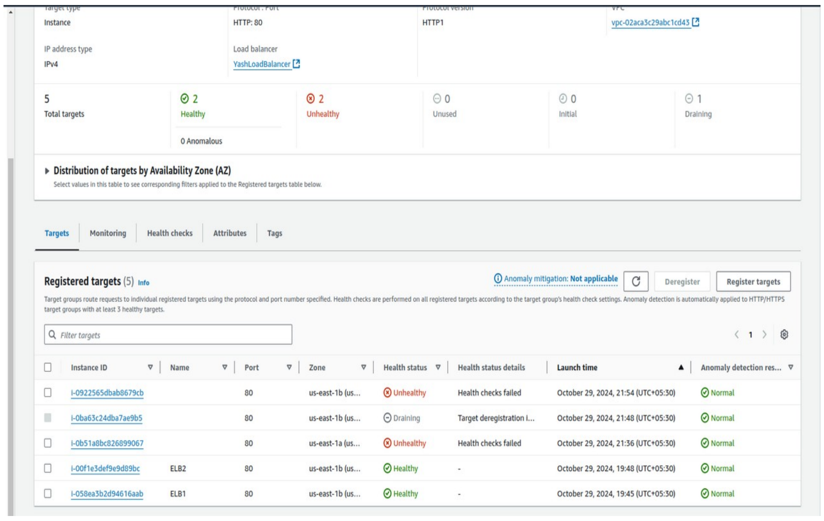


Status	Description
Successful	Launching a new EC2 instance: i-0ba63c24dba7ae9b5
Connection draining in progress	Terminating EC2 instance: i-09032891e2f1b5d60 - Waiting For ELB Connection Draining.
Successful	Launching a new EC2 instance: i-09032891e2f1b5d60
Successful	Terminating EC2 instance: i-0520c215350ab6072
Successful	Launching a new EC2 instance: i-0b51a8bc826899067
Successful	Terminating EC2 instance: i-04388de244c0f0724
Successful	Launching a new EC2 instance: i-0520c215350ab6072
Successful	Terminating EC2 instance: i-0a858922b55ccb3f9
Successful	Launching a new EC2 instance: i-04388de244c0f0724
Successful	Terminating EC2 instance: i-0bb09d396260a84c0

Resource map



Target Group



Target type: Instance
Protocol : Port: HTTP: 80
Protocol version: HTTP1
VPC: vpc-02aca3c29abc1cd43
IP address type: IPv4
Load balancer: YashLoadBalancer

5 Total targets
2 Healthy
0 Anomalous
2 Unhealthy
0 Unused
0 Initial
1 Draining

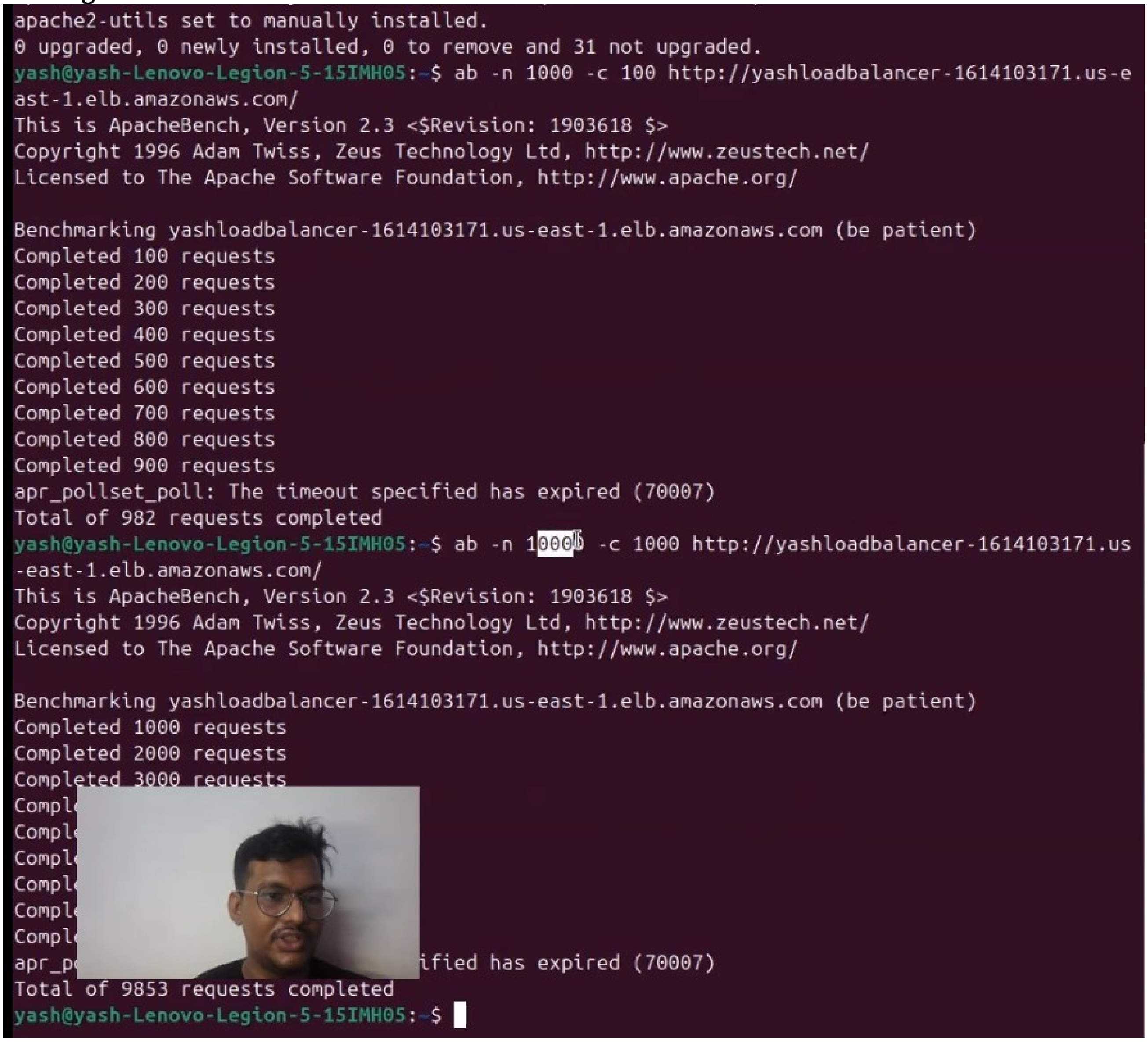
Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.

Registered targets (5) Info

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Instance ID	Name	Port	Zone	Health status	Health status details	Launch time	Anomaly detection res...
i-0922565dbab8679cb		80	us-east-1b (us...	Unhealthy	Health checks failed	October 29, 2024, 21:54 (UTC+05:30)	Normal
i-0ba63c24dba7ae9b5		80	us-east-1b (us...	Draining	Target deregistration i...	October 29, 2024, 21:48 (UTC+05:30)	Normal
i-0b51a8bc826899067		80	us-east-1a (us...	Unhealthy	Health checks failed	October 29, 2024, 21:36 (UTC+05:30)	Normal
i-00f1e3def9e9d89bc	ELB2	80	us-east-1b (us...	Healthy	-	October 29, 2024, 19:48 (UTC+05:30)	Normal
i-058ea3b2d94616aab	ELB1	80	us-east-1b (us...	Healthy	-	October 29, 2024, 19:45 (UTC+05:30)	Normal

Testing The LoadBalancers & Auto scalers



```
apache2-utils set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
yash@yash-Lenovo-Legion-5-15IMH05:~$ ab -n 1000 -c 100 http://yashloadbalancer-1614103171.us-e
ast-1.elb.amazonaws.com/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking yashloadbalancer-1614103171.us-east-1.elb.amazonaws.com (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
apr_pollset_poll: The timeout specified has expired (70007)
Total of 982 requests completed
yash@yash-Lenovo-Legion-5-15IMH05:~$ ab -n 10000 -c 1000 http://yashloadbalancer-1614103171.us
-east-1.elb.amazonaws.com/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking yashloadbalancer-1614103171.us-east-1.elb.amazonaws.com (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Compl
Compl
Compl
Compl
Compl
Compl
apr_p                          ified has expired (70007)
Total of 9853 requests completed
yash@yash-Lenovo-Legion-5-15IMH05:~$
```

apache2-utils

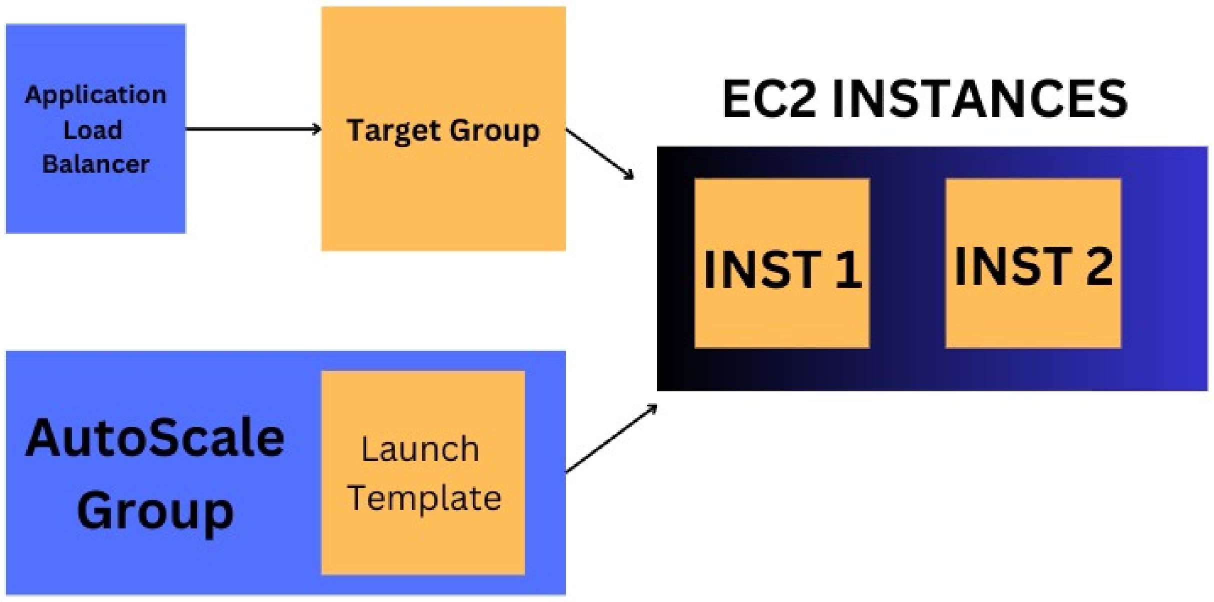
- **What It Is**: A package that includes various utility programs for the Apache HTTP server.
- **Key Tool**: Contains `ab` (Apache Benchmark), a command-line tool for testing the performance of HTTP servers.
- **Installation**:
    - Installed using the command: `sudo apt-get install apache2-utils`
    - Ensures you have tools for performance testing and other server-related utilities.

ab Command (`ab -n`)

- **Purpose**: Used to benchmark the performance of your web server by simulating multiple requests.
- **Syntax**: `ab -n <total_requests> -c <concurrent_requests> <URL>`
    - `-n <total_requests>`: Specifies the total number of requests to perform (e.g., 1000 or 10000).
    - `-c <concurrent_requests>`: Sets the number of multiple requests to perform at a time (e.g., 100 or 1000).

- `<URL>`: The URL of the web server you are testing (e.g., your ALB DNS).

AWS EC2 Deployement Architecture


Application Load Balancer
Target Group
EC2 INSTANCES
INST 1
INST 2
AutoScale Group
Launch Template