Name: Md Hasibul Haque Zahid
ID: 2302302

Assignment 3

A short description of the steps that I took to implement my elastic web service, and the used parameters.

1. **EC2 Instance**: First, I logged into the AWS Management Console and went to the EC2 service. From there, I selected "Launch Instance" to start a new virtual server. I selected the appropriate instance type, network configuration, and operating system (e.g., Amazon Linux 2).
2. **Connect to Access Instance**: Once the instance was up and running, I accessed it via SSH using the command line. For example, I used a command like **ssh -i my-key.pem ec2-user@<instance-public-ip >** to establish a secure connection to the instance. (**My Key was : Zahid11.pem**)
3. **Setting Up & Install PHP**: After logging into the instance, I installed PHP and any necessary dependencies using the package manager available on the operating system. For instance, on Amazon Linux, I might have used the command **sudo yum install php** to install PHP.
4. **Installing HTTPS Certificate**: To secure the web service with HTTPS, I obtained an SSL/TLS certificate from a Certificate Authority (CA) like Let's Encrypt. I installed the certificate on the instance and configured the web server (e.g., Apache or Nginx) to use HTTPS for encrypted communication.
5. **Configuring Web Service**: Finally, I moved my PHP application files to the appropriate directory on the instance and set up the web server to serve them. This entailed editing server configuration files (such as httpd.conf for Apache) to specify the document root and other options. **The used httperf command(s) to create load on the instances**

- httperf --server assign3lb-1-100238849.us-east-1.elb.amazonaws.com--uri=/index.php –wsess= 700,5,2 --rate 1 --timeout 5
- httperf --server assign3lb-1-100238849.us-east-1.elb.amazonaws.com--uri=/index.php --wsess=1500,10,2 --rate 1 --timeout 5
- httperf --server assign3lb-1-100238849.us-east-1.elb.amazonaws.com--uri=/index.php --wsess=2600,10,2 --rate 1 --timeout 5

Name: Md Hasibul Haque Zahid
ID: 2302302

Hello! My name is Md Zahid and my IP addres is: 172.31.85.145
random value was:728
Execution time:0.17448401451111 second
Done !

Fig 1: the PHP code run Successfully

Fig 2: Graph Execution

Name: Md Hasibul Haque Zahid
ID: 2302302

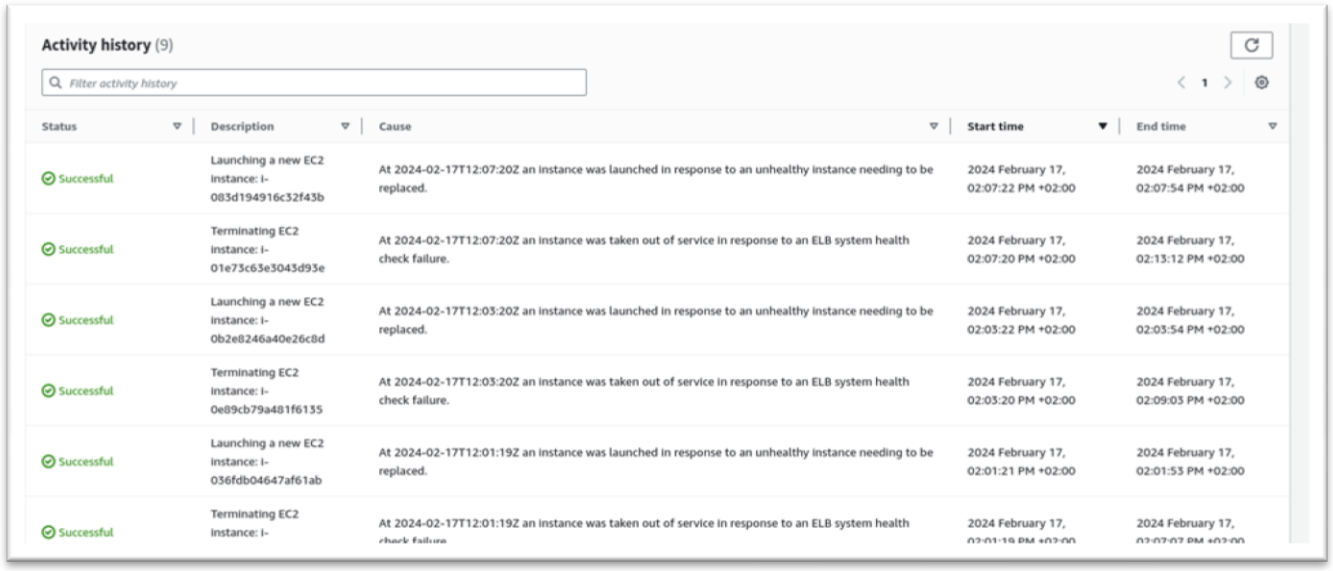

Fig 3: Launch Instance Due to Balance the Load

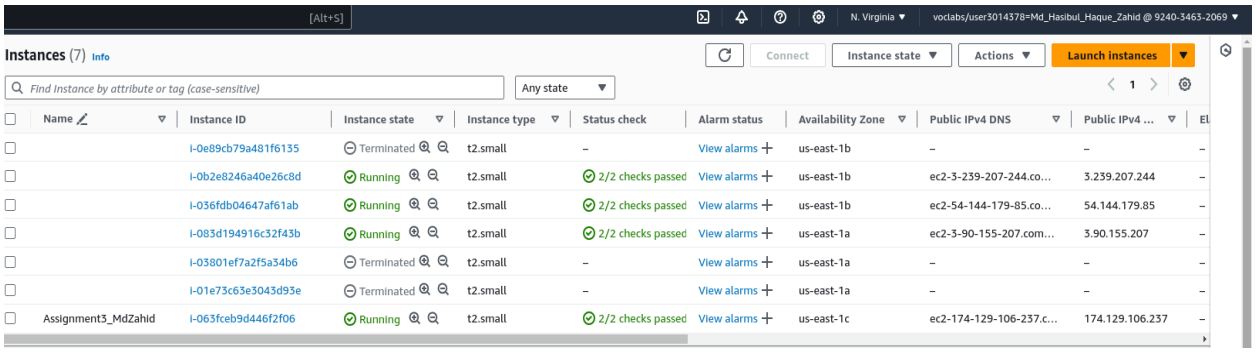

Fig 4: Dynamically Running & Terminate Instance

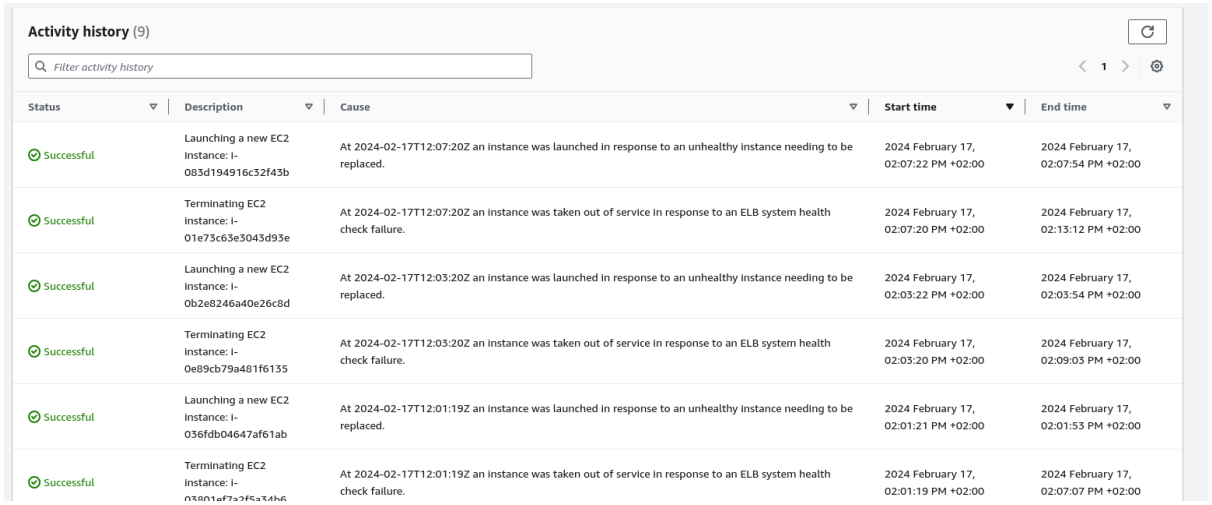

Fig: EC2 Instance Activity History

Name: Md Hasibul Haque Zahid
ID: 2302302

Video link: https://drive.google.com/file/d/18Mrj9U_AgYL7DTWjn51ZY1yLc0Yf-8md/view?usp=sharing

A detailed analysis on the obtained behavior and reactivity:

The auto-scaling mechanism for cloud resources in my web service allows for dynamic adjustment of computing capacity based on variations in load. Here's a detailed analysis of the obtained behavior and reactivity to load variations:

1. **Increased Load**: When there is an increase in traffic or workload, the auto-scaling mechanism detects the extra load using predefined metrics such as CPU utilization, network traffic, or request rate. When the threshold for increased load is exceeded, the auto-scaling mechanism creates additional EC2 instances to handle the excess load. These instances are launched quickly to ensure that the web service can continue to handle incoming requests without losing performance.
2. **Decreased Load**: When the workload decreases and the load on existing instances fall below a certain level, the auto-scaling mechanism reduces the number of instances. It terminates excess instances in order to maximize resource utilization and reduce costs. This ensures that resources are not wasted during times of low demand.

Behavior and Reaction Time:

- The behavior of the auto-scaling mechanism is reactive, meaning it responds dynamically to changes in load in near real-time.
- Reaction time depends on various factors such as the scaling policies configured, the metrics used for scaling decisions, and the time it takes to launch or terminate instances. Generally, auto-scaling reactions can occur within minutes, depending on the configuration and the cloud provider's infrastructure.

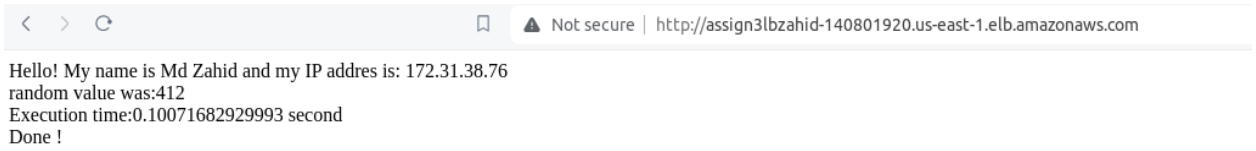

Fig: Load Balancer DNS

Name: Md Hasibul Haque Zahid
ID: 2302302

Possible Web Applications Benefitting from Auto-Scaling:

1. **E-commerce Platforms**: During peak shopping seasons or sales events, e-commerce websites experience sudden spikes in traffic. An auto-scaling mechanism ensures that the website can handle increased traffic without downtime or performance issues, ensuring a seamless shopping experience for customers.(Ex**: AliBaba)**
2. **Netflix/Movie Streaming Services**: Video streaming platforms often experience fluctuations in demand based on popular content releases or live events. Auto-scaling allows these platforms to dynamically adjust server capacity to meet demand, ensuring smooth playback and minimal buffering for users.
3. **Social Media Applications**: Social media platforms may experience viral content or events leading to a sudden surge in user activity. Auto-scaling ensures that the platform can accommodate the increased user engagement without slowdowns or errors. **(Ex: Facebook)**
4. **Online Gaming**: Multiplayer online games require scalable infrastructure to handle varying numbers of concurrent players. Auto-scaling allows gaming platforms to add or remove server capacity based on the number of active players, ensuring a seamless gaming experience for users. **(Ex: Steam Gaming Server)**

Reflection:

Before doing this assignment I always thought may be someone behind the scenario is doing this may be a human but During this exercise, I've gained valuable insights into the intricacies of implementing and managing a web service on AWS, particularly focusing on auto-scaling mechanisms using load balancer.

Have you learned anything completely new?

This is completely a new experience for me. Though I struggled a lot but fortunately I can complete this assignment. Load Balancer and Auto Scaling totally new things for me.

Did anything surprise you? The seamless integration of auto-scaling with AWS services was surprising. The ability to automatically provision and terminate instances based on predefined metrics is quite impressive and highlights the power of cloud computing.

Did you find anything challenging? Why? My challenges was in network setting group & load balancer at first it was but difficult for me in first time to complete this things without error but after carefully checking I somehow managed to overcome .

Did you find anything satisfying? Why? It was satisfying to see how auto-scaling can effectively handle fluctuations in workload without manual intervention. The ability to maintain optimal performance and resource utilization even during peak traffic periods is immensely gratifying, as it ensures a seamless user experience while optimizing costs.