CAPSTONE PROJECT

Infra Optimization.

**Course-end Project 1**

DESCRIPTION

Create a DevOps infrastructure for an e-commerce application to run on high-availability mode.

**Background of the problem statement:**  
A popular payment application, **EasyPay** where users add money to their wallet accounts, faces an issue in its payment success rate. The timeout that occurs with  
the connectivity of the database has been the reason for the issue.  
While troubleshooting, it is found that the database server has several downtime instances at irregular intervals. This situation compels the company to create their own infrastructure that runs in high-availability mode.  
Given that online shopping experiences continue to evolve as per customer expectations, the developers are driven to make their app more reliable, fast, and secure for improving the performance of the current system.

**Implementation requirements:**

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
5. Create a new user with permissions to create, list, get, update, and delete pods
6. Configure application on the pod
7. Take snapshot of ETCD database
8. Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

**The following tools must be used:**

1. EC2
2. Kubernetes
3. Docker
4. Ansible or Chef or Puppet

**The following things to be kept in check:**

1. You need to document the steps and write the algorithms in them.
2. The submission of your GitHub repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, then executing them, and recording the results.
4. You need to submit the final specification document, which includes:

* Project and tester details
* Concepts used in the project
* Links to the GitHub repository to verify the project completion
* Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  |  |  |  |  |  |  |
|  |  |  | |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

**I.Create the cluster (EC2 instances with load balancer and elastic IP)**

In this project we will try to implement the automation distribution of incoming web traffic across EC2 instances on the AWS cloud.

We will use ELB (Amazon Elastic Load Balancing).

**1-Let’s start by see what Elastic Load balancing (ELB) is.**

EBL automatically distributes incoming application traffic across multiple targets and virtual appliances in one or more Availability Zones (AZs).

ELB has advantages and disadvantages.

**a-Advantages of ELB**

i-ELB automatically distributes incoming application traffic across multiples targets, such as EC2 instances, containers, and IP addresses to achieve high availability.

ii-It can automatically scale to handle change in traffic demand, allowing to maintain consistent application performance.

iii-It can monitor the health of its registered targets and route traffic only to the healthy targets.

iv-It evenly distributes traffic across all availability zones in a region, improving fault tolerance.

**b-Disadvantages of ELB**

i-ELB can add latency to the application, as traffic must pass through the load balancer before being route to the targets

ii-It has limited customization options, so we may need to use additional tools and services to fully meet our application’s requirements.

iii-It can introduce additional complexity to our application architecture, requiring to manage and maintain additional resources.

iv-It can increase our overall AWS costs, especially if we have high traffic volumes or require multiples load balancers.

**2-Let’s now see the different types of Load Balancer.**

They usually list 4 types of ELB

a-Classic Load Balancer.

It’s the traditional form of load balancer. It distributes the traffic among the instances and is not intelligent enough to support host-based routing or path-based routing. It ends up reducing efficiency and performance in certain situations. It is operated on connection level as well as requesting level. Classic Load Balancer is in the Transport Layer (TCP/SSL) or the Application Layer (HTTP/HTTPS).

b- Application Load Balancer.

This type of Load Balancer is used when decisions are to be made related to HTTP/HTTPS/ Nginx traffic routing. It supports path-based routing and host-based routing. This Load Balancing works at the Application Layer of the OSI Model. It also supports dynamic host port mapping.

c-Network Load Balancer.

It works at the Transport Layer (TCP/SSL) of the OSI Model. It’s capable of handling millions of requests per second. It’s mainly used for load balancing TCP traffic.

d-Gateway Load Balancer.

Gateway Load Balancer provides the facility to deploy, scale, and manage virtual appliances like firewall. Gateway Load Balancer combines a transparent network gateway and then distributes the traffic.

**3-Step to configure our Application Load Balancer on AWS**

In this project we will build 4 EC2 in and deploy 4 AZ. We will make sure we have a load balancer that will distribute the traffic.

ELB can manage traffic across single or multiple AZ.

31- Schema

32- Create Security Group

33-Create 4 instances

34-Create Target group

35-Create Load Balancer

36-Test Load Balancer

37-Create Rules

38-Test Rules

**31.Below is the schema of our application.**

Application Load

Balancer

Listener

Listener

**Electronics All Items**

**Target Group**

**Target Group**

Instance3

Instance1

Instance2

Instance4

**32-Security group**

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

33-Create 4 instances

Launch 4 servers on AWS Instance1, instance2, instance3 and instance4

**Below is the script to create the websites.**

*#!/bin/bash*

*yum update -y*

*yum install httpd -y*

*systemctl start httpd*

*systemctl enable httpd*

*cd /var/www/html*

*echo "<p>Welcome to Jean.in</p><h1>This is Instence1</h1>" >index.html*

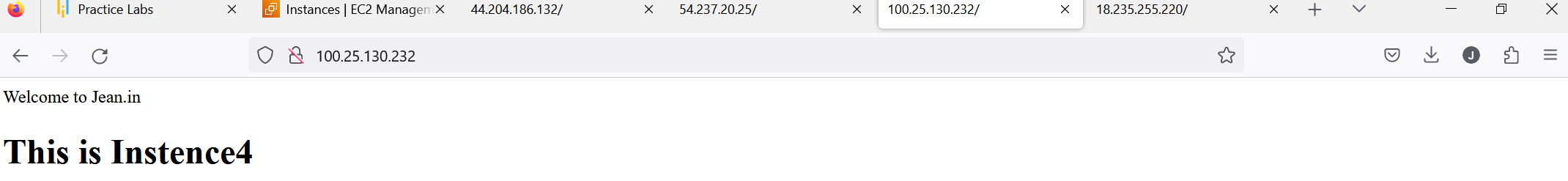
**Evidence of Websites successfully working**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generatedGraphical user interface, application, Word

Description automatically generated**

**Graphical user interface, application, table

Description automatically generated**

**34-Let’s create Targets Group**

**Graphical user interface, text

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, email

Description automatically generated**

**Graphical user interface, application, Word

Description automatically generated**

**35-Let’s create Load Balancer**

**Graphical user interface, text

Description automatically generated**

**Diagram

Description automatically generated**

**We are creating Application Load Balancer**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**36-Test of Load Balancer**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, application, email, website

Description automatically generated**

**37-Rules**

**Let’s connect to Instence3.**

**A computer screen capture

Description automatically generated with medium confidence**

**Let’s connect to Instence4.**

**Text

Description automatically generated**

**Let’s go back to Load Balancer**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email, website

Description automatically generated**

**Graphical user interface, text, website

Description automatically generated**

**Graphical user interface, text

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**38-Test for Rules**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**ELASTIC IP ADDRESS**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer

Description automatically generated



Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

Test: nginx is running successfully.

Graphical user interface, text, application

Description automatically generated