**Creating a High-Availability Web Application Infrastructure on AWS with VPC, Auto Scaling, and Application Load Balancer**

**A screenshot of a computer

Description automatically generated**

**AWS Architecture Overview**

* This project outlines the architecture for deploying a scalable web application on Amazon Web Services (AWS). Below are the key components and configurations:
* Virtual Private Cloud (VPC)

A VPC has been established with the address range 10.0.0.0/16 to isolate the environment and provide network control.

* Public Subnets

Three public subnets have been created within the VPC, each with its own CIDR block (10.0.0.0/20,

10.0.16.0/20, 10.0.32.0/20). These subnets are strategically placed to ensure redundancy and

availability.

* Auto Scaling Group

An Auto Scaling Group has been configured to manage the EC2 instances. Utilizing t2.micro instances,

the group is set to maintain a minimum of 2 instances and a maximum of 5 instances, ensuring

scalability based on demand. Apache is installed on each instance to serve web content.

* Application Load Balancer (ALB)

To evenly distribute incoming traffic among the EC2 instances, an Application Load Balancer has been

deployed. This load balancer enhances the availability and fault tolerance of the application.

* Security Groups

Web Server Security Group: This group allows inbound HTTP traffic only from the Application Load

Balancer, enhancing security by restricting access to authorized sources.

Load Balancer Security Group: Inbound HTTP traffic is permitted from any source (0.0.0.0/0), allowing

external access to the load balancer.

**Creating a VPC**

1. In the search bar type VPC -> Create VPC -> VPC and more -> Name your VPC -> Add CIDR block 10.0.0.0/16

A screenshot of a computer

Description automatically generated

1. Deploying subnets in different availability zones provides high availability and fault tolerance for your applications. This is because if one availability zone becomes unavailable, the other two can continue to handle traffic and requests, preventing any disruption to your application or service. Create three public subnets with 10.0.0.0/20 & 10.0.16.0/20 & 10.0.32.0/20.

A screenshot of a chat

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Create VPC

A screenshot of a computer program

Description automatically generated

**Creating A Launch Template**

1. In the search bar type EC2 -> Scroll down to Instances -> Launch templates -> Select Create launch template
2. Name launch template. For AMI select Amazon Linux. For instance type select t2.micro. Select your keypair name.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Under Network settings select Create security group -> Name your security group -> Allow SSH and HTTP -> Select your VPC (it’s automatically in default so you have to change it).

A screenshot of a computer

Description automatically generated

1. For inbound security rules add SSH and HTTP.

A screenshot of a computer

Description automatically generated

1. For Advanced network configuration select Enable for Auto-assign public IP. A public IP address is necessary for instances in a public subnet to communicate with the internet, receive incoming traffic, and respond to requests from external clients or users.

A screenshot of a computer

Description automatically generated

6. In Advanced details scroll to the bottom until you see User data and paste following the command.

#!/bin/bash

# Update the system and install Apache

yum update -y

yum install -y httpd

# Start Apache and enable it to start on boot

systemctl start httpd

systemctl enable httpd

# Fetch the availability zone of the EC2 instance

EC2AZ=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)

# Create a simple HTML page displaying the availability zone

echo "<center><h1>This Amazon EC2 instance is located in Availability Zone: $EC2AZ </h1></center>" > /var/www/html/index.html

1. Create Launch template.

A screenshot of a computer

Description automatically generated

**Creating auto scaling group**

1. Type EC2 in the search bar -> On the left-hand side locate Auto Scaling -> Auto Scaling Groups -> Select Create Auto Scaling Group -> Name group -> Select launch template we just created.

A screenshot of a computer

Description automatically generated

1. Select the VPC we created earlier -> Select all Availability Zones and subnets -> Select next.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. We attach a new load balancer in order to distribute incoming traffic evenly across all the instances in the group, ensuring that no single instance becomes overwhelmed or overloaded with requests. An internet-facing load balancer has a public IP address, which clients on the internet can use to connect to your application.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. When you create a listener and routing rule on your load balancer, you specify which target group(s) should receive the incoming traffic.

A screenshot of a computer program

Description automatically generated

1. Enabling group metrics collection with CloudWatch for your load balancer allows you to monitor the performance of your load balancer and its associated resources.

A screenshot of a computer

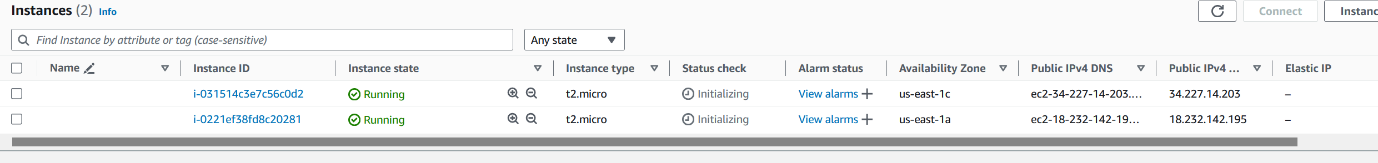
Description automatically generated

1. The Auto Scaling minimum should be 2 and the maximum.

A screenshot of a computer

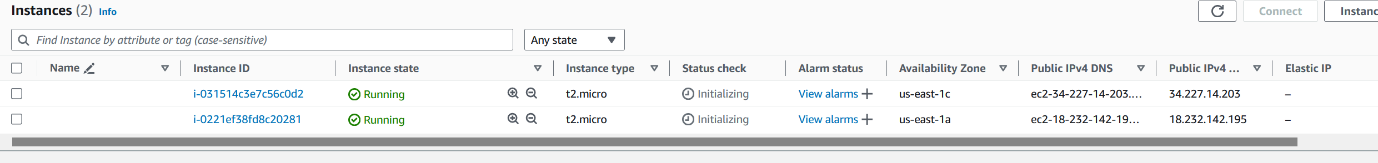
Description automatically generated

1. Click Next until you locate Create Auto Scaling Group.



**Let's check if our instances are up and running!**

1. In the search bar type EC2 -> Instances.



1. Locate the Public IPv4 address -> Open browser -> [http://34.227.14.203](http://34.201.53.174).

A screen shot of a computer

Description automatically generated

1. Everything looks good to go!.

**Advanced:**

Add a target policy for the ASG to scale after cpu utilization is above 50%. After the autoscaling group has been created, find a stress tool to be able to stress an instance above 50% to see if your scaling policy works! After the autoscaling group has been created, find a stress tool to be able to stress an instance above 50% to see if your scaling policy works!

1. In the search bar type EC2-> Scroll down to Auto Scaling groups > Select group > Go to Automatic scaling > Create dynamic scaling policy.
2. Enter 50 for the target value -> Create.

A screenshot of a computer

Description automatically generated

1. SSH into one of your instances and run the following commands to install a stress utility.

A computer screen shot of a computer program

Description automatically generated

1. Once installed, CPU load can be generated using Stress by running:

A screenshot of a computer

Description automatically generated

1. Instance surpassed 50%.

A screenshot of a computer

Description automatically generated

A screenshot of a video

Description automatically generated

1. Instance surpassed 50% and generated a new instance:

A screenshot of a computer

Description automatically generated

**Clean up:**

1. In the search bar type EC2 -> Detach Load balancer -> Delete Auto Scaling Group -> Delete Launch Template -> Terminate Instance

2. In the search bar type VPC -> Delete VPC