Scalable Web Application Architecture on AWS

Table of Contents

Introduction

This document provides a comprehensive overview of the implementation of a scalable web application architecture on AWS. The project aims to leverage AWS services such as Auto Scaling, Load Balancing, and backend service integration to ensure high availability, scalability, and reliability of the web application.

Project Overview

The project involves designing and implementing a web application architecture on AWS to support a scalable and resilient application. Key objectives include:

- Implementing load balancing to distribute traffic across multiple instances.
- Setting up Auto Scaling to dynamically adjust the number of instances based on demand.
- Integrating backend services such as Memcached, RabbitMQ, and MySQL for caching, messaging, and data storage

Architecture Overview

The architecture consists of:

- Application Load Balancer to distribute incoming traffic.
- Auto Scaling group to manage instances dynamically.
- Backend services including Memcached, RabbitMQ, and MySQL
- Integration with Route ٥٣ for DNS management
- SSL certificate from AWS Certificate Manager for secure communication

AWS Services Used

- EC٢: Virtual servers for running the web application instances.
- ELB: Application Load Balancer for distributing traffic.
- Auto Scaling: Dynamically adjusting the number of instances based on demand.
- S٣: Storing deployment artifacts and other resources.
- Route ٥٣: DNS management for domain registration.
- AWS Certificate Manager: SSL certificate for secure communication.
- Memcached: In-memory caching for improved performance.
- RabbitMQ: Message broker for communication between components.
- MySQL: Backend database for storing application data.

Implementation Steps

١. Set up AWS account and configure IAM roles and permissions.
٢. Launch EC٢ instances for the web application and backend services.
٣. Configure security groups, IAM roles, and instance profiles.
٤. Create an Application Load Balancer and set up target groups.
٥. Configure Auto Scaling policies and scaling rules.
٦. Integrate backend services with the web application instances.
٧. Configure Route ٥٣ for domain registration and SSL certificate.
٨. Deploy application artifacts to S٣ bucket.
٩. Test and monitor the deployment.

Backend Services Integration

- Memcached: Used for caching frequently accessed data to improve performance.
- RabbitMQ: Used as a message broker for asynchronous communication between components.
- MySQL: Used as the backend database for storing application data.

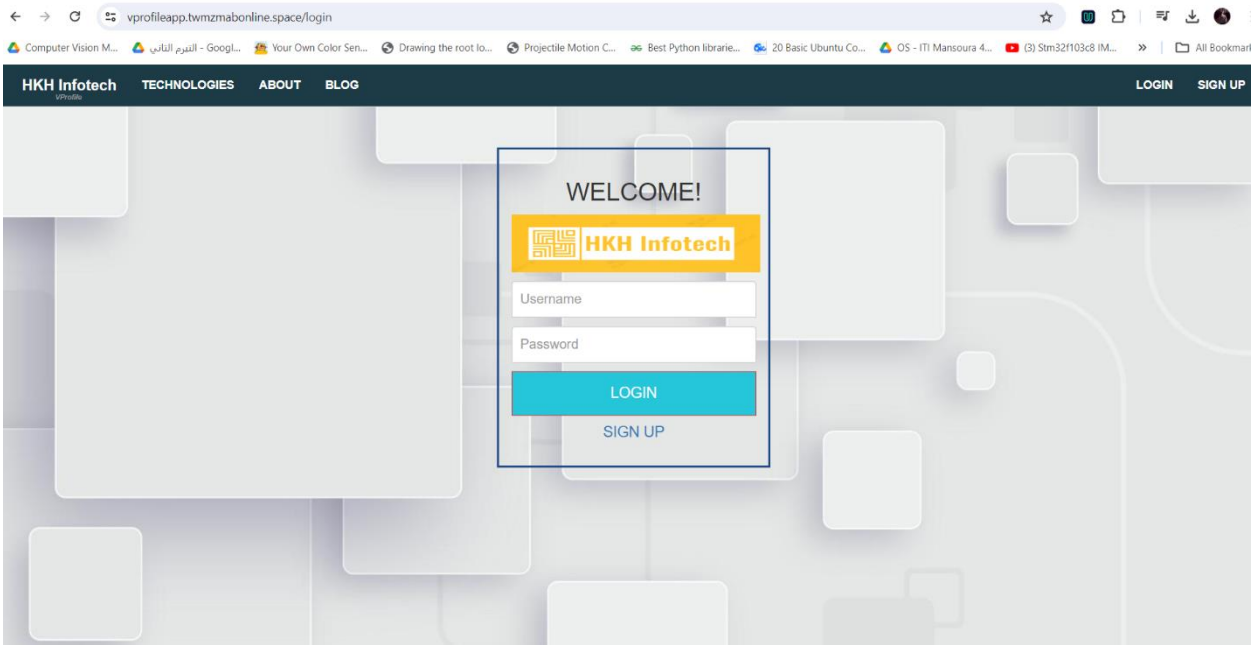
Deployment Process

١. Build application artifacts using Maven.
٢. Upload artifacts to S٣ bucket.
٣. Monitor deployment status and performance metrics using CloudWatch.
٤. Test application functionality and scalability under various load conditions.

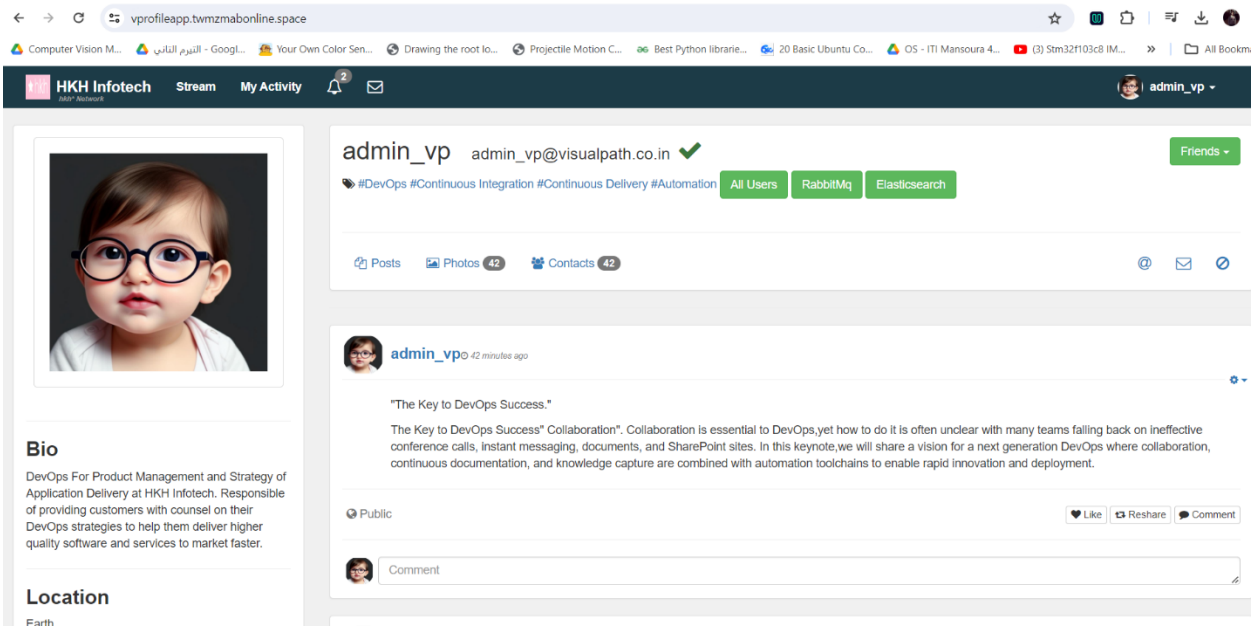
Conclusion

The implementation of the scalable web application architecture on AWS has been successful in achieving the project objectives of high availability, scalability, and reliability. The use of AWS services such as Auto Scaling, Load Balancing, and backend service integration has enabled the deployment of a resilient and efficient web application.

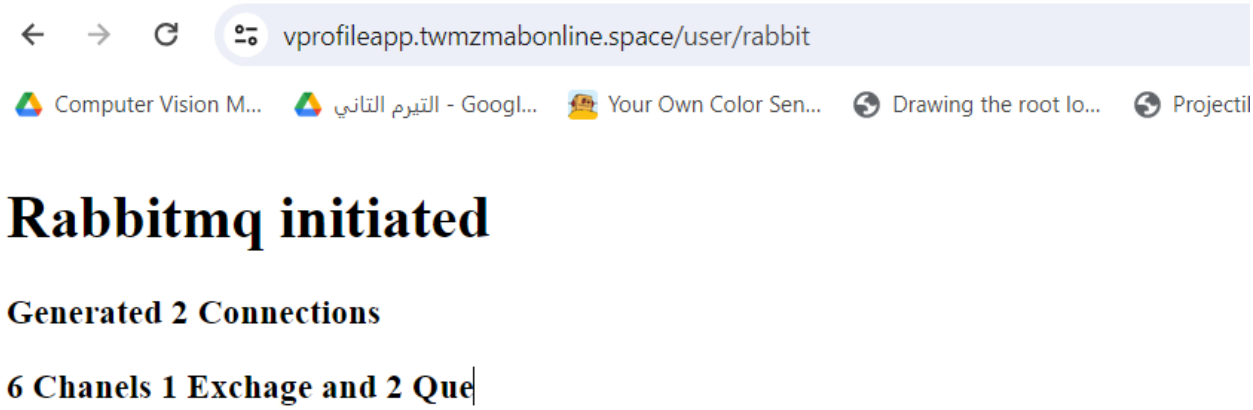
Web App Login Page



After logging in, the application is successfully connecting to the database.



Testing RabbitMQ functionality.



Testing Mem-cache functionality