Project Title: Resilient and Scalable Web Application Deployment on AWS

Project Description

This project aims to design and implement a highly available and scalable infrastructure for a web application on AWS. The architecture will leverage AWS services to ensure the web application can efficiently handle varying loads and maintain high availability across multiple Availability Zones.

Objectives

High Availability: Achieve minimal downtime by utilising multiple Availability Zones.
Scalability: Use AWS Auto Scaling to adjust resources automatically in response to traffic changes.
Security: Implement security measures focusing on security groups and secure communication.
Resilience: Develop an application setup that can withstand failures and traffic spikes without manual intervention.

Core AWS Services Utilisation

Virtual Private Cloud (VPC): Provides an isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define.
Elastic File System (EFS): Offers a simple, scalable, elastic file storage for use with AWS Cloud services and on-premises resources.
Elastic Compute Cloud (EC2): Provides scalable computing capacity in the AWS cloud. It allows developers to scale up or down based on demand.
AWS Auto Scaling: Monitors applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.
Application Load Balancer (ALB): Automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses.
Route 53: A highly available and scalable cloud Domain Name System (DNS) web service, designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications.

Architecture Overview

High-Level Design

- Diagram: Include a cloud architecture diagram showing the relationship between VPC, EFS, EC2 instances, Auto Scaling, ALB, and Route 53.
- VPC Configuration: Detail the setup of the VPC, including subnets across multiple Availability Zones for high availability.
- Security: Describe the security groups, IAM roles, and policies to ensure secure access and communication.

Detailed Component Design

VPC

- Purpose: Isolation and provision of a secure network for AWS resources.
- Configuration: Subnet creation across multiple AZs for resilience and high availability.

EFS

- Purpose: Provide a scalable file storage system accessible by EC2 instances.
- Configuration: Set up to ensure it is mounted on all EC2 instances for shared storage needs.

EC2

- Purpose: Host the web application and handle compute tasks.
- Configuration: Instance types, AMIs, and initial bootstrapping scripts.

AWS Auto Scaling

- Purpose: Dynamically adjust the number of EC2 instances based on traffic.
- Configuration: Define scaling policies based on metrics like CPU utilisation and request rates.

ALB

- Purpose: Distribute incoming traffic across multiple targets to increase the availability and fault tolerance of your application.
- Configuration: Listener and rule setup for routing traffic to the appropriate target groups.

Route 53

- Purpose: Connect user requests to the infrastructure running in AWS.
- Configuration: DNS management, health checks, and routing policies to ensure high availability and traffic management.

Security Measures

- Security Groups: Define inbound and outbound traffic rules for EC2 instances and other services.
- IAM Roles: Secure API calls from EC2 instances.
- Data Encryption: At rest (EFS) and in transit using TLS.

Scalability and High Availability Strategy

- Auto Scaling: Setup and configuration details, including scaling policies.
- Multi-AZ Deployment: Ensure the application is deployed across multiple AZs to achieve high availability.

Conclusion

This document outlines the initial design for deploying a resilient and scalable web application on AWS. It emphasises high availability, scalability, security, and resilience, leveraging AWS services to achieve these objectives. The next steps include detailed planning, implementation, testing, and deployment phases, ensuring the web application meets the outlined objectives.