**Project Title: Deploying a Scalable Web Application Infrastructure Using Terraform**

Project Overview:

In this project, you will build a fully scalable web application infrastructure on AWS using Terraform. The objective is to deploy a complete infrastructure that can scale according to demand and is fault-tolerant. The project starts with basic resource provisioning and progresses to advanced concepts like modules, state management, and multi-environment deployments.

Problem Statement:

You have been tasked with building and managing the infrastructure for a web application that serves users globally. The company expects rapid growth, and the infrastructure must be scalable, resilient, and easy to manage. Your team has chosen AWS as the cloud provider, and you need to use Terraform as the Infrastructure-as-Code (IaC) tool to provision and manage resources.

**Phases of the Project:**

Phase 1: Basic Resource Provisioning

1. Objective: Learn the basics of Terraform by provisioning essential AWS resources.

2. Tasks:

- Write a Terraform configuration to create the following resources:

- A VPC with subnets (public and private).

- An EC2 instance with a specified AMI and instance type.

- An S3 bucket for storing logs.

- Security Groups to control inbound/outbound traffic.

- Apply the Terraform plan to provision the resources.

- Ensure proper SSH access and basic network security.

Phase 2: Scaling and Load Balancing

1. Objective: Add scalability and high availability to the infrastructure.

2. Tasks:

- Add an Auto Scaling Group to automatically adjust the number of EC2 instances based on traffic.

- Use an Elastic Load Balancer (ELB) to distribute traffic across multiple EC2 instances.

- Modify the VPC to add more subnets across multiple Availability Zones.

- Create RDS (MySQL/PostgreSQL) instances in a private subnet for database needs.

- Ensure the web application is accessible through the ELB's DNS.

Phase 3: Using Modules

1. Objective: Modularize the infrastructure code for better reusability and maintainability.

2. Tasks:

- Create reusable Terraform modules for common components like VPC, EC2 instances, and Security Groups.

- Refactor the existing Terraform code to use these modules.

- Ensure the modules can be reused across different environments (development, staging, production).

Phase 4: State Management and Workspaces

1. Objective: Manage Terraform state effectively for collaboration and multiple environments.

2. Tasks:

- Configure a remote backend (e.g., S3 with DynamoDB) for storing Terraform state files.

- Use Terraform workspaces to manage different environments (dev, staging, prod).

- Demonstrate how state locking works and how to avoid conflicts in a team environment.

- Test the setup by deploying the infrastructure to multiple environments.

Phase 5: Advanced Topics (CI/CD Integration, Secrets, and Monitoring)

1. Objective: Implement advanced Terraform features and best practices for production-ready infrastructure.

2. Tasks:

- Integrate CI/CD pipelines (e.g., using GitHub Actions or Jenkins) to automatically apply Terraform plans upon code changes.

- Use AWS Secrets Manager or SSM Parameter Store to securely store and retrieve sensitive information (e.g., database passwords, API keys).

- Implement CloudWatch for monitoring EC2 instance performance and set up alarms for key metrics (e.g., CPU utilization).

- Set up Terraform Sentinel (optional) to enforce policies and security best practices.

Deliverables:

1. Terraform Configuration Files: Well-structured .tf files for each phase.

2. Terraform Modules: Reusable modules for key components.

3. CI/CD Pipeline: A working pipeline for automating Terraform deployments.

4. Documentation: Comprehensive documentation explaining the setup, configuration, and how each phase was executed.

5. Monitoring and Logs: Logs and metrics for monitoring the infrastructure.

Expected Outcomes:

- A production-ready, scalable, and highly available web application infrastructure.

- Terraform best practices like modularization, remote state management, and CI/CD integration.

- A deeper understanding of scaling, secrets management, and monitoring in a cloud environment.