1. Introduction

Infrastructure as Code (IaC) Deployment with Terraform

This document provides a detailed guide on how to set up and deploy the infrastructure for a front-end and back-end application on AWS using Terraform. The infrastructure includes a VPC with public and private subnets, EC2 instances, an Application Load Balancer, RDS (MySQL), and an S3 bucket for static file storage.

2. Prerequisites

Prerequisites

- AWS account
- AWS CLI installed and configured
- Terraform installed

3. Architecture Overview

Architecture Overview

The infrastructure consists of the following components:

- **VPC:** Virtual Private Cloud with public and private subnets for secure networking.
- **EC2 Instances:** Deployed in the private subnet for hosting the applications.
- **Application Load Balancer (ALB):** Deployed in the public subnet to distribute traffic to the EC2 instances.
- **RDS (MySQL):** Deployed in the private subnet for database storage.
- **S3 Bucket:** For storing static files such as images, CSS, and JavaScript.

`ALB.tf`

Purpose: This file sets up an Application Load Balancer (ALB) in AWS to distribute incoming traffic across multiple targets, such as EC2 instances, in one or more Availability Zones.

Components:

1. **Application Load Balancer (ALB):**
    - Creates an ALB named "ALB" which is public-facing (not internal).
    - Configured with security groups and subnets.
2. **Target Group:**
    - Defines a target group named "targetgroup" which will be associated with the ALB to route requests.
    - Health checks are configured to ensure targets are healthy.
3. **ALB Target Group Attachment:**

- Attaches the target group to the frontend EC2 instance, enabling the instance to receive traffic from the ALB.

4. **ALB Listener:**
    - Creates a listener on port 80 (HTTP) to forward traffic to the target group.

`Key-Pair.tf`

Purpose: This file generates and manages an SSH key pair for secure access to EC2 instances.

Components:

1. **TLS Private Key:**
    - Generates a 4096-bit RSA private key.
2. **AWS Key Pair:**
    - Creates an AWS key pair named "tfkey" using the generated public key.
3. **Null Resource for PEM File:**
    - Creates a local PEM file on the user's computer containing the private key.
4. **Null Resource to Change Permissions:**
    - Changes the permissions of the PEM file to ensure it is secure (chmod 400).

`main.tf`

Purpose: This file defines the main EC2 instances for the frontend and backend servers.

Components:

1. **Frontend EC2 Instance:**
    - Creates an EC2 instance using specified AMI and instance type.
    - Associates it with a security group and subnet.
    - Uses the generated key pair for SSH access.
2. **Backend EC2 Instance:**
    - Similar to the frontend instance, but for the backend server.
    - Also configured with a security group, subnet, and key pair.

`output.tf`

Purpose: This file defines outputs for important information generated by the Terraform configuration.

Components:

1. **Frontend and Backend Server IPs:**
    - Outputs the private IP addresses of the frontend and backend EC2 instances.
2. **Attached Key:**
    - Outputs the name of the PEM file used for SSH access.
3. **Frontend Access URL:**
    - Outputs the DNS name of the ALB, providing a URL to access the frontend server.

`provider.tf`

Purpose: This file specifies the Terraform provider configuration for AWS.

Components:

1. **Terraform Provider Requirements:**
    - Specifies that the AWS provider version 5.40.0 is required.
2. **AWS Provider Configuration:**
    - Configures the AWS provider with the region, access key, and secret key.

`RDS.tf`

Purpose: This file sets up an RDS (MySQL) instance for database services.

Components:

1. **RDS Subnet Group:**
    - Creates a subnet group for the RDS instance, specifying the subnets in which the RDS can be launched.
2. **RDS Instance:**
    - Creates an RDS instance with specified engine version, instance class, storage, username, and password.
    - Configured to be within a private subnet and associated with a security group.

`S3-bucket.tf`

Purpose: This file sets up an S3 bucket for static file storage.

Components:

1. **Random String:**
    - Generates a random string to ensure the S3 bucket name is unique.
2. **S3 Bucket:**
    - Creates an S3 bucket named "development-{random_string}" for storing static files.
    - Tags the bucket for identification.

`Security-groups.tf`

Purpose: This file defines security groups to control inbound and outbound traffic to and from the EC2 instances, ALB, and RDS.

Components:

1. **Frontend Security Group:**
    - Allows inbound SSH (port 22) and HTTP (port 80) traffic.
2. **Backend Security Group:**
    - Allows inbound SSH (port 22) traffic.
3. **ALB Security Group:**
    - Allows inbound HTTP (port 80) traffic from any IP.
4. **RDS Security Group:**
    - Allows inbound MySQL (port 3306) traffic from within the VPC.

`terraform.tfvars`

Purpose: This file contains the variable values used in the Terraform configuration.

Components:

- Specifies the AWS region, access key, secret key, AMI ID, instance type, database username, and password.

`variable.tf`

Purpose: This file defines the variables used in the Terraform configuration.

Components:

- Defines variables for access key, secret key, region, AMI ID, instance type, database username, and password.

`vpc.tf`

Purpose: This file sets up the VPC (Virtual Private Cloud) and its related components like subnets, internet gateway, and route tables.

Components:

1. **VPC:**
    - Creates a VPC with a specified CIDR block.
2. **Internet Gateway:**
    - Creates an internet gateway for the VPC.
3. **Public Subnets:**
    - Creates public subnets in specified availability zones.
    - Associates route tables to enable internet access.
4. **Private Subnets:**
    - Creates private subnets in specified availability zones.
    - Associates route tables without internet access.

4. Terraform Configuration (setup terraform.tfvars)

The terraform.tfvars file will include the variables used in the project.
The below variables are used in the project .

terraform.tfvars

```
region = "us-west-1"

access_key = "YOUR_ACCESS_KEY"

secret_key = "YOUR_SECRET_KEY"

ami = "ami-12345678"

instance_type = "t2.micro"

db_username = "your_db_username"
```

```
db_password = "your_db_password"
```

6. Deployment Instructions

Deployment Instructions

1. **Initialize Terraform:**

```
terraform init
```

2. **Validate the Configuration:**

```
terraform validate
```

3. **Plan the Deployment:**

```
terraform plan
```

4. **Apply the Configuration:**

```
terraform apply
```

7. Design Decisions

Design Decisions

- **VPC and Subnets:** Separate public and private subnets for enhanced security and isolation of resources.
- **EC2 Instances:** Deployed in the private subnet to reduce exposure to the internet.
- **ALB:** Used for distributing traffic to ensure high availability and scalability.
- **RDS (MySQL):** Placed in the private subnet to restrict direct access from the internet.
- **S3 Bucket:** Utilized for storing static files, reducing the load on the EC2 instances.

8. Conclusion

Conclusion

This document provides a comprehensive guide to setting up and deploying a secure and scalable infrastructure on AWS using Terraform. The architecture ensures high availability, security, and efficient resource management.