Terraform AWS Infrastructure Project

Overview

This project provisions a complete AWS infrastructure environment using Terraform. It follows an Infrastructure-as-Code (IaC) approach to define, configure, and deploy AWS networking components, load balancers, security groups, compute instances, and routing.

The architecture is modular, reusable, and scalable, built using Terraform modules that align with best practices for AWS resource provisioning.

Key Features

- Modular AWS infrastructure setup using Terraform modules
- VPC creation with public and private subnets
- Internet Gateway and NAT Gateway configuration
- Route tables for network communication
- Application Load Balancer (ALB)
- EC2 instances for backend and proxy layers
- Auto-configuration of instances via userdata scripts
- Security groups for layered network protection
- Terraform remote state locking supported via `.terraform.lock.hcl`

Project Structure

```
Terraform/
│
├── main.tf                  # Root module orchestrating all components
├── variables.tf             # Input variables for the infrastructure
├── outputs.tf               # Output values after deployment
├── .terraform/              # Provider and module caches
├── .terraform.lock.hcl      # Lock file to maintain provider versions
├── terraform.tfstate        # Local state file (for local dev only)
├── terraform.tfstate.backup # Backup state file
│
├── modules/                 # Reusable Terraform modules
│   ├── alb/                 # Application Load Balancer
│   ├── backend/             # Backend EC2 instance configuration
│   ├── igw/                 # Internet Gateway
│   ├── nat/                 # NAT Gateway
│   ├── private_subnets/     # Private subnet definitions
```

```
│   ├── proxy/              # Proxy server configuration
│   ├── public_subnets/     # Public subnet definitions
│   ├── route_tables/       # Route table configuration
│   ├── security_groups/    # Security group setups
│   └── vpc/                # VPC main module
│
└── scripts/
    ├── backend_userdata.sh  # Init script for backend EC2 instances
    └── proxy_userdata.sh    # Init script for proxy EC2 instances
```

How to Use

1. Install Dependencies

Ensure the following are installed: - Terraform v1.x or later - AWS CLI configured with valid credentials

2. Initialize the Project

Run:

```
terraform init
```

This downloads providers and initializes modules.

3. Review the Execution Plan

```
terraform plan
```

This allows you to verify the resources that will be created.

4. Apply the Infrastructure

```
terraform apply
```

Confirm when prompted, and Terraform will build the AWS environment.

5. Destroy the Infrastructure (Optional)

```
terraform destroy
```

This removes all resources created by the project.

Modules Overview

VPC Module

Creates the main Virtual Private Cloud with CIDR blocks.

Subnet Modules

- Public subnets for load balancers and NAT
- Private subnets for backend instances

Route Tables Module

Configures routing rules for public and private networks.

Internet Gateway & NAT Gateway

Provides outbound connectivity for resources.

Security Groups Module

Applies layered firewall rules to manage inbound and outbound traffic.

ALB Module

Deploys an Application Load Balancer with listeners and target groups.

Backend & Proxy Modules

EC2 instances with userdata scripts for automatic configuration.

Userdata Scripts

Located in `scripts/`: - `backend_userdata.sh`: Initializes application backend settings - `proxy_userdata.sh`: Configures proxy routing and dependencies

These scripts run automatically when instances boot.

Outputs

The project outputs several useful values, such as: - VPC ID - Subnet IDs - ALB DNS Name - EC2 instance public/private IPs

---

Recommended Improvements

- Use remote backend (S3 + DynamoDB) for production state management
- Add autoscaling groups for backend instances
- Integrate Terraform Cloud or CI/CD pipeline
- Add more detailed tagging for cost management

License

This project uses Terraform modules and AWS resources under their respective licenses. The repo includes AWS provider license text.

If you need enhancement, diagrams, or an explanation section for interview use, let me know.