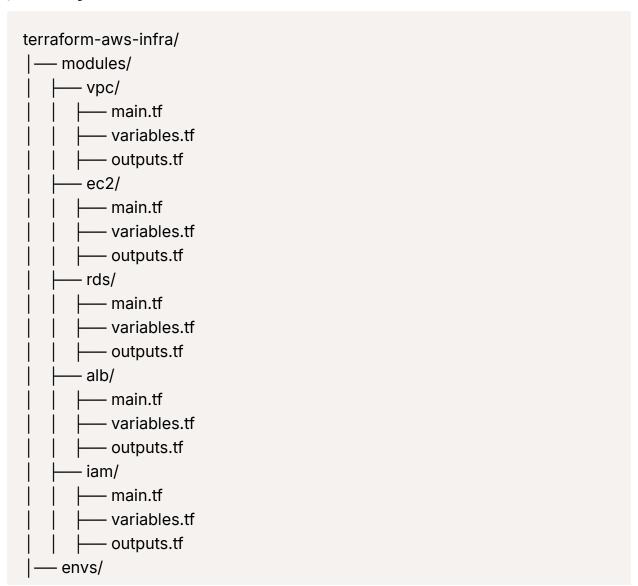
Terraform Example Project

GitHub-ready, production-level Terraform repository structure project. This will include:

- Modular Terraform setup (VPC, EC2, RDS, ALB, IAM)
- State Management with S3 & DynamoDB
- Production Best Practices

★ Project Structure



- **✓ Modular approach** Each service (VPC, EC2, RDS) is a separate module.
- **▼ Environment separation** dev and prod folders for different AWS accounts.
- ▼ Remote state backend Terraform state stored in S3 with DynamoDB locking.

★ 1. Backend Configuration (backend.tf)

```
terraform {
  backend "s3" {
  bucket = "terraform-state-bucket"
  key = "prod/terraform.tfstate"
  region = "us-east-1"
  encrypt = true
  dynamodb_table = "terraform-locks"
  }
}
```

Stores state file in S3 with locking via DynamoDB.

2. Provider Configuration (provider.tf)

```
provider "aws" {
region = "us-east-1"
```

Specifies AWS provider & region.

★ 3. VPC Module (modules/vpc/main.tf)

```
resource "aws_vpc" "main" {
    cidr_block = "10.0.0.0/16"
    enable_dns_support = true
    enable_dns_hostnames = true
}

resource "aws_subnet" "public" {
    count = 2
    vpc_id = aws_vpc.main.id
    cidr_block = element(["10.0.1.0/24", "10.0.2.0/24"], count.index)
    map_public_ip_on_launch = true
}

resource "aws_internet_gateway" "gw" {
    vpc_id = aws_vpc.main.id
}
```

Creates a custom VPC with subnets and internet gateway.

★ 4. EC2 Module (modules/ec2/main.tf)

```
resource "aws_launch_template" "web_server" {
    name_prefix = "web-server"
    image_id = "ami-0c55b159cbfafe1f0"
    instance_type = "t3.micro"
    user_data = filebase64("user_data.sh")
}

resource "aws_autoscaling_group" "web_asg" {
    vpc_zone_identifier = [var.subnet_id]
    desired_capacity = 2
    max_size = 4
```

```
min_size = 2
launch_template {
  id = aws_launch_template.web_server.id
  version = "$Latest"
  }
}
```

Auto Scaling + Launch Template for high availability.

★ 5. RDS Module (modules/rds/main.tf)

```
resource "aws_db_instance" "app_db" {
    allocated_storage = 20
    engine = "mysql"
    instance_class = "db.t3.micro"
    db_name = "app_db"
    username = var.db_user
    password = var.db_password
    vpc_security_group_ids = [var.db_sg_id]
    skip_final_snapshot = true
}
```

✓ Manages an RDS instance with security groups.

★ 6. Application Load Balancer (modules/alb/main.tf)

Sets up an ALB for better traffic distribution.

7. IAM Role for EC2 to Access S3

modules/iam/main.tf

```
resource "aws_iam_role" "ec2_s3_role" {
    name = "ec2-s3-access"
    assume_role_policy = <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "sts:AssumeRole",
          "Principal": { "Service": "ec2.amazonaws.com" },
          "Effect": "Allow"
      }
    ]
}
EOF
}

resource "aws_iam_role_policy_attachment" "s3_attachment" {
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    role = aws_iam_role.ec2_s3_role.name
}
```

Allows EC2 instances to securely access S3.

★ 8. Environment-Specific Config (envs/prod/main.tf)

```
module "vpc" {
  source = "../modules/vpc"
}
```

```
module "ec2" {
  source = "../modules/ec2"
  subnet_id = module.vpc.public_subnet_id
}

module "rds" {
  source = "../modules/rds"
  db_user = "admin"
  db_password = "YourSecurePassword"
}
```

✓ Manages separate environments (dev , prod) using modules.

📌 9. Deploying Infrastructure

```
# Step 1: Initialize Terraform
terraform init

# Step 2: Preview Changes
terraform plan

# Step 3: Deploy Infra
terraform apply -auto-approve
```

Automates deployment of all AWS resources.

10. GitHub Actions for Terraform CI/CD

```
name: Terraform Deployment
```

on:
push:
branches:

- main

jobs:

terraform:

runs-on: ubuntu-latest

steps:

- name: Checkout Repo

uses: actions/checkout@v2

- name: Setup Terraform

uses: hashicorp/setup-terraform@v1

- name: Terraform Init run: terraform init

 name: Terraform Plan run: terraform plan

- name: Terraform Apply

run: terraform apply -auto-approve

▼ Triggers Terraform deployment on every Git push.



- 1. Host this project on GitHub.
- 2. Integrate it with Jenkins/GitHub Actions for CI/CD.
- 3. Enhance it with Kubernetes (EKS) or Serverless (Lambda).