# Comprehensive Terraform Example: Declarative Language and Concepts

This document provides a comprehensive example of a Terraform configuration written in its declarative language. The example demonstrates key concepts and features, including providers, resources, variables, outputs, modules, workspaces, conditionals, dynamic blocks, and state management. The scenario involves deploying a highly available web application on AWS.

## Provider Configuration

provider "aws" {  
 region = var.aws\_region  
}

## Input Variables

variable "aws\_region" {  
 description = "AWS region"  
 default = "us-east-1"  
}  
  
variable "instance\_type" {  
 description = "Type of EC2 instance"  
 default = "t2.micro"  
}  
  
variable "app\_port" {  
 description = "Port for the application"  
 default = 8080  
}  
  
variable "availability\_zones" {  
 description = "List of availability zones"  
 default = ["us-east-1a", "us-east-1b"]  
}

## VPC and Subnets

resource "aws\_vpc" "main" {  
 cidr\_block = "10.0.0.0/16"  
 tags = {  
 Name = "main-vpc"  
 }  
}  
  
resource "aws\_subnet" "subnets" {  
 count = length(var.availability\_zones)  
 vpc\_id = aws\_vpc.main.id  
 cidr\_block = "10.0.${count.index}.0/24"  
 availability\_zone = var.availability\_zones[count.index]  
 tags = {  
 Name = "subnet-${count.index}"  
 }  
}

## Security Group

resource "aws\_security\_group" "web\_sg" {  
 name = "web-sg"  
 description = "Allow HTTP and SSH traffic"  
 vpc\_id = aws\_vpc.main.id  
  
 ingress {  
 from\_port = var.app\_port  
 to\_port = var.app\_port  
 protocol = "tcp"  
 cidr\_blocks = ["0.0.0.0/0"]  
 }  
  
 ingress {  
 from\_port = 22  
 to\_port = 22  
 protocol = "tcp"  
 cidr\_blocks = ["0.0.0.0/0"]  
 }  
  
 egress {  
 from\_port = 0  
 to\_port = 0  
 protocol = "-1"  
 cidr\_blocks = ["0.0.0.0/0"]  
 }  
  
 tags = {  
 Name = "web-sg"  
 }  
}

## EC2 Instances

resource "aws\_instance" "web" {  
 count = 2  
 ami = "ami-0c55b159cbfafe1f0" # Example Amazon Linux 2 AMI  
 instance\_type = var.instance\_type  
 subnet\_id = aws\_subnet.subnets[count.index].id  
 security\_groups = [aws\_security\_group.web\_sg.name]  
  
 tags = {  
 Name = "web-instance-${count.index}"  
 }  
}

## Load Balancer

resource "aws\_lb" "app\_lb" {  
 name = "app-lb"  
 internal = false  
 load\_balancer\_type = "application"  
 security\_groups = [aws\_security\_group.web\_sg.id]  
 subnets = aws\_subnet.subnets[\*].id  
  
 tags = {  
 Name = "app-lb"  
 }  
}  
  
resource "aws\_lb\_target\_group" "tg" {  
 name = "app-tg"  
 port = var.app\_port  
 protocol = "HTTP"  
 vpc\_id = aws\_vpc.main.id  
}  
  
resource "aws\_lb\_listener" "listener" {  
 load\_balancer\_arn = aws\_lb.app\_lb.arn  
 port = 80  
 protocol = "HTTP"  
 default\_action {  
 type = "forward"  
 target\_group\_arn = aws\_lb\_target\_group.tg.arn  
 }  
}  
  
resource "aws\_lb\_target\_group\_attachment" "targets" {  
 count = length(aws\_instance.web)  
 target\_group\_arn = aws\_lb\_target\_group.tg.arn  
 target\_id = aws\_instance.web[count.index].id  
 port = var.app\_port  
}

## Outputs

output "instance\_public\_ips" {  
 value = aws\_instance.web[\*].public\_ip  
}  
  
output "load\_balancer\_dns" {  
 value = aws\_lb.app\_lb.dns\_name  
}

## Modules

module "logging" {  
 source = "./modules/logging"  
 log\_bucket\_name = "app-logs-bucket"  
}

## Workspaces

terraform workspace new dev  
terraform workspace select dev  
terraform apply

## Backend Configuration

terraform {  
 backend "s3" {  
 bucket = "terraform-state-bucket"  
 key = "vpc/terraform.tfstate"  
 region = "us-east-1"  
 encrypt = true  
 dynamodb\_table = "terraform-lock"  
 }  
}