**Exercise 1: Basic Terraform Configuration**

**Objective**: Create a basic Terraform configuration to deploy an AWS EC2 instance.

**Task**:

1. Write a Terraform configuration to deploy a single EC2 instance in AWS.
2. Use the aws\_instance resource.
3. Set the instance type to t2.micro.
4. Use the latest Amazon Linux 2 AMI.
5. Define the AWS region as us-east-1.

**Solution**:

# Configure the AWS provider

provider "aws" {

region = "us-east-1"

}

# Fetch the latest Amazon Linux 2 AMI

data "aws\_ami" "latest\_amazon\_linux" {

most\_recent = true

owners = ["amazon"]

filter {

name = "name"

values = ["amzn2-ami-hvm-\*-x86\_64-gp2"]

}

}

# Create an EC2 instance

resource "aws\_instance" "example" {

ami = data.aws\_ami.latest\_amazon\_linux.id

instance\_type = "t2.micro"

}

**Commands to Run**:

terraform init

terraform plan

terraform apply

**Exercise 2: Managing Terraform State**

**Objective**: Explore the Terraform state by viewing and understanding the state file.

**Task**:

1. Create a configuration that defines an AWS S3 bucket.
2. Apply the configuration.
3. Use terraform show and terraform state list commands to explore the state.

**Solution**:

# Configure AWS provider

provider "aws" {

region = "us-east-1"

}

# Create an S3 bucket

resource "aws\_s3\_bucket" "example" {

bucket = "my-unique-bucket-name-123"

acl = "private"

}

**Commands to Run**:

terraform init

terraform apply

terraform show

terraform state list

**Exercise 3: Variables and Outputs**

**Objective**: Use input variables and output values in your configuration.

**Task**:

1. Create a variable for the EC2 instance type.
2. Create a variable for the AWS region.
3. Output the public IP address of the EC2 instance.

**Solution**:

# Variables

variable "instance\_type" {

description = "EC2 instance type"

type = string

default = "t2.micro"

}

variable "aws\_region" {

description = "AWS region"

type = string

default = "us-east-1"

}

# Provider configuration

provider "aws" {

region = var.aws\_region

}

# Fetch the latest Amazon Linux 2 AMI

data "aws\_ami" "latest\_amazon\_linux" {

most\_recent = true

owners = ["amazon"]

filter {

name = "name"

values = ["amzn2-ami-hvm-\*-x86\_64-gp2"]

}

}

# EC2 instance resource

resource "aws\_instance" "example" {

ami = data.aws\_ami.latest\_amazon\_linux.id

instance\_type = var.instance\_type

}

# Output the public IP

output "instance\_public\_ip" {

value = aws\_instance.example.public\_ip

}

**Commands to Run**:

terraform init

terraform apply

**Exercise 4: Using Data Sources**

**Objective**: Fetch data using data sources in Terraform.

**Task**:

1. Use a data source to fetch information about an existing AWS VPC.
2. Output the VPC ID.

**Solution**:

# AWS provider

provider "aws" {

region = "us-east-1"

}

# Data source to fetch VPC details

data "aws\_vpc" "default" {

default = true

}

# Output VPC ID

output "vpc\_id" {

value = data.aws\_vpc.default.id

}

**Commands to Run**:

terraform init

terraform apply

**Exercise 5: Managing Resources with terraform destroy**

**Objective**: Learn to destroy resources with Terraform.

**Task**:

1. Create a resource, such as an S3 bucket.
2. Use terraform destroy to remove the resource.

**Solution**:

# AWS provider

provider "aws" {

region = "us-east-1"

}

# Create an S3 bucket

resource "aws\_s3\_bucket" "example" {

bucket = "my-unique-bucket-to-delete"

acl = "private"

}

**Commands to Run**:

terraform init

terraform apply

terraform destroy

**Exercise 6: Terraform Format and Validate**

**Objective**: Practice using terraform fmt and terraform validate to ensure code quality.

**Task**:

1. Write a basic Terraform configuration for an EC2 instance.
2. Use terraform fmt to format your code.
3. Use terraform validate to check for syntax errors.

**Solution**:

# AWS provider configuration

providers "aws" {

region = "us-east-1"

}

# EC2 instance resource

resource "aws\_instance" "example" {

ami = "ami-0c55b159cbfafe1f0" # Example AMI ID

instance\_type = "t2.micro"

}

**Commands to Run**:

terraform fmt

terraform validate

terraform apply