create a **simple Terraform module example for an S3 bucket**.

**🔹 Step 1: Project Structure**

terraform-s3-module/

│

├── main.tf

├── variables.tf

├── outputs.tf

│

└── modules/

└── s3-bucket/

├── main.tf

├── variables.tf

└── outputs.tf

**🔹 Step 2: S3 Module**

📂 modules/s3-bucket/main.tf

resource "aws\_s3\_bucket" "this" {

bucket = var.bucket\_name

tags = {

Name = var.bucket\_name

Environment = var.environment

}

}

resource "aws\_s3\_bucket\_versioning" "this" {

bucket = aws\_s3\_bucket.this.id

versioning\_configuration {

status = var.versioning

}

}

📂 modules/s3-bucket/variables.tf

variable "bucket\_name" {

description = "Name of the S3 bucket"

type = string

}

variable "environment" {

description = "Environment (dev, test, prod)"

type = string

default = "dev"

}

variable "versioning" {

description = "Enable versioning (Enabled or Suspended)"

type = string

default = "Enabled"

}

📂 modules/s3-bucket/outputs.tf

output "bucket\_id" {

value = aws\_s3\_bucket.this.id

}

output "bucket\_arn" {

value = aws\_s3\_bucket.this.arn

}

**🔹 Step 3: Root Configuration**

📂 main.tf

provider "aws" {

region = "us-east-1"

}

module "my\_s3" {

source = "./modules/s3-bucket"

bucket\_name = "my-terraform-demo-bucket-12345"

environment = "dev"

versioning = "Enabled"

}

📂 outputs.tf

output "s3\_bucket\_id" {

value = module.my\_s3.bucket\_id

}

output "s3\_bucket\_arn" {

value = module.my\_s3.bucket\_arn

}

**🔹 Step 4: Run Terraform**

terraform init

terraform plan

terraform apply -auto-approve

✅ This will create an **S3 bucket** with versioning enabled.

**🔹 Step 1: Project Structure**

Terraform modules are just **directories with .tf files** that can be reused. Let’s create this structure:

terraform-modules-example/

│

├── main.tf

├── variables.tf

├── outputs.tf

│

└── modules/

└── ec2-instance/

├── main.tf

├── variables.tf

└── outputs.tf

**🔹 Step 2: Write the EC2 Module**

📂 modules/ec2-instance/main.tf

resource "aws\_instance" "this" {

ami = var.ami

instance\_type = var.instance\_type

tags = {

Name = var.instance\_name

}

}

resource "aws\_security\_group" "this" {

name = "${var.instance\_name}-sg"

description = "Allow SSH inbound traffic"

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"]

}

}

📂 modules/ec2-instance/variables.tf

variable "ami" {

description = "AMI ID for the EC2 instance"

type = string

}

variable "instance\_type" {

description = "EC2 instance type"

type = string

default = "t2.micro"

}

variable "instance\_name" {

description = "Tag Name for the EC2 instance"

type = string

}

📂 modules/ec2-instance/outputs.tf

output "instance\_id" {

value = aws\_instance.this.id

}

output "public\_ip" {

value = aws\_instance.this.public\_ip

}

**🔹 Step 3: Call the Module from Root**

📂 main.tf

provider "aws" {

region = "us-east-1"

}

module "my\_ec2" {

source = "./modules/ec2-instance"

ami = "ami-0c55b159cbfafe1f0" # Example Amazon Linux 2 AMI (check for your region)

instance\_type = "t2.micro"

instance\_name = "MyTerraformEC2"

}

📂 variables.tf

# (Optional if you want to parameterize root-level configs)

📂 outputs.tf

output "ec2\_id" {

value = module.my\_ec2.instance\_id

}

output "ec2\_public\_ip" {

value = module.my\_ec2.public\_ip

}

**🔹 Step 4: Run Terraform Commands**

terraform init

terraform plan

terraform apply -auto-approve

You’ll get the EC2 **instance ID** and **public IP** from outputs.

**🔹 Step 5: Benefits of Modules**

* **Reusable**: Can use the same ec2-instance module for multiple instances.
* **Maintainable**: Logic stays separate from variables.
* **Scalable**: You can add VPC, RDS, or S3 as more modules.