What is a Terraform Module?

A **Terraform Module** is a container for multiple Terraform resources that are **grouped together** and **used repeatedly** across projects.

Think of it like a **function** in programming:

- It accepts **inputs (variables)**
- It performs tasks (**creates infrastructure**)
- It returns **outputs**

Why Use Modules?

Benefit	Description
🔄 Reusability	Define once, use anywhere (e.g., EC2 setup, VPC, S3, etc.)
🧼 Clean Code	Break complex infra into smaller, manageable pieces
📦 Consistency	Same infra config for multiple environments (dev, test, prod)
⬆ Scalability	Easily scale infra using loops, count, and maps
📚 Versioning	Share via Terraform Registry or Git

Module Folder Structure

Here's a basic structure of a module:

```
project/
│
├── main.tf
├── variables.tf
├── outputs.tf
├── terraform.tfvars
│
└── modules/
    └── ec2/
        ├── main.tf
        ├── variables.tf
        └── outputs.tf
```

Example: Creating an EC2 Module

Step 1: Create the Module (modules/ec2/)

modules/ec2/main.tf

```
resource "aws_instance" "this" {
  ami           = var.ami
  instance_type = var.instance_type
  tags = {
    Name = var.name
  }
}
```

modules/ec2/variables.tf

```
variable "ami" {
  type = string
}

variable "instance_type" {
  type = string
  default = "t2.micro"
}
```

```
variable "name" {
  type = string
}
```

modules/ec2/outputs.tf

```
output "instance_id" {
  value = aws_instance.this.id
}
```

Step 2: Call the Module from Root Module

main.tf (root project)

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


module "web_instance" {
  source        = "./modules/ec2"
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.medium"
  name          = "WebServer"
}
```

Step 3: Run Terraform

```
terraform init

terraform plan

terraform apply
```

Reusing Modules

You can use modules in **multiple places** like this:

```
module "web_instance" {
  source = "./modules/ec2"
  ami    = "ami-0xyz"
  name   = "Web"
}


module "db_instance" {
  source = "./modules/ec2"
  ami    = "ami-0xyz"
  name   = "Database"
  instance_type = "t3.medium"
}
```

Using Public Modules

You can also use modules from the **Terraform Registry**:

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "5.1.0"


  name = "my-vpc"
  cidr = "10.0.0.0/16"
  azs  = ["us-east-1a", "us-east-1b"]
  ...
}
```

Outputs from Modules

To expose data from a module (like instance ID, IP), define output in module and access it like:

```
output "web_instance_id" {

  value = module.web_instance.instance_id

}
```

Best Practices for Modules

Practice	Description
🧩 One purpose per module	e.g., one for EC2, one for S3
📝 Document variables	Use description fields
🔐 Avoid hardcoding	Use variables and tfvars
📦 Version control	If using remote source, pin versions
🔄 Use locals	For computed values inside modules

When to Use Modules

Use modules when:

- You're **repeating** the same set of resources
- You want **environment separation** (dev/prod)
- You want **cleaner Terraform files**
- You're working on a **team project**