Terraform Module

https://www.terraform.io/language/modules

- Modules are containers for multiple resources that are used together. A module consists of a collection of `.tf` files kept together in a directory.
- Modules are the main way to package and reuse resource configurations with Terraform.
- Every Terraform configuration has at least one module, known as its **root module**, which consists of the resources defined in the .tf files in the **main working directory**
- A module that has been called by another module is often referred to as a **child module.**
- Child modules can be called multiple times within the same configuration, and multiple configurations can use the same child module.
- In addition to modules from the **local filesystem**, Terraform can load modules from a **public or private registry.**
- This makes it possible to **publish modules** for others to use and to use modules that others have published.

Terraform Directory Structure

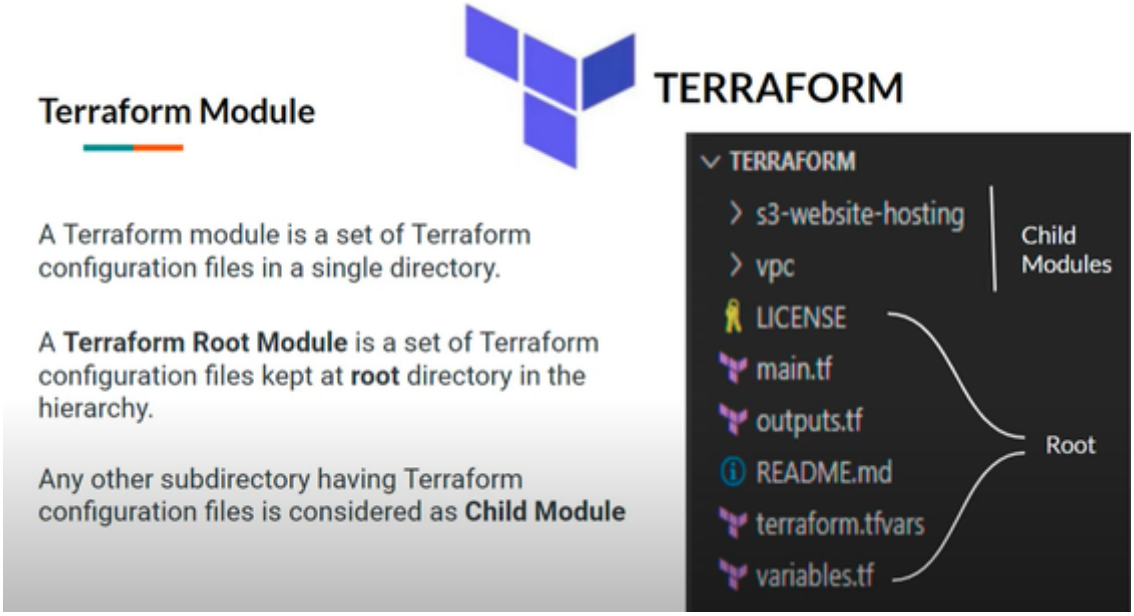
To design our Module, we create a new directory and put one or more **.tf** files on that directory, just as we would do for a **root module**. Because the Terraform can read modules either from our **local computer or remote repositories**.

The directory below is a structure of a module and root module.

```
.
 README.md
 main.tf
 variables.tf
 outputs.tf
```

Example of a root and child module

https://github.com/leonardtia1/21_days_of_aws_using_terraform



Public registry module version

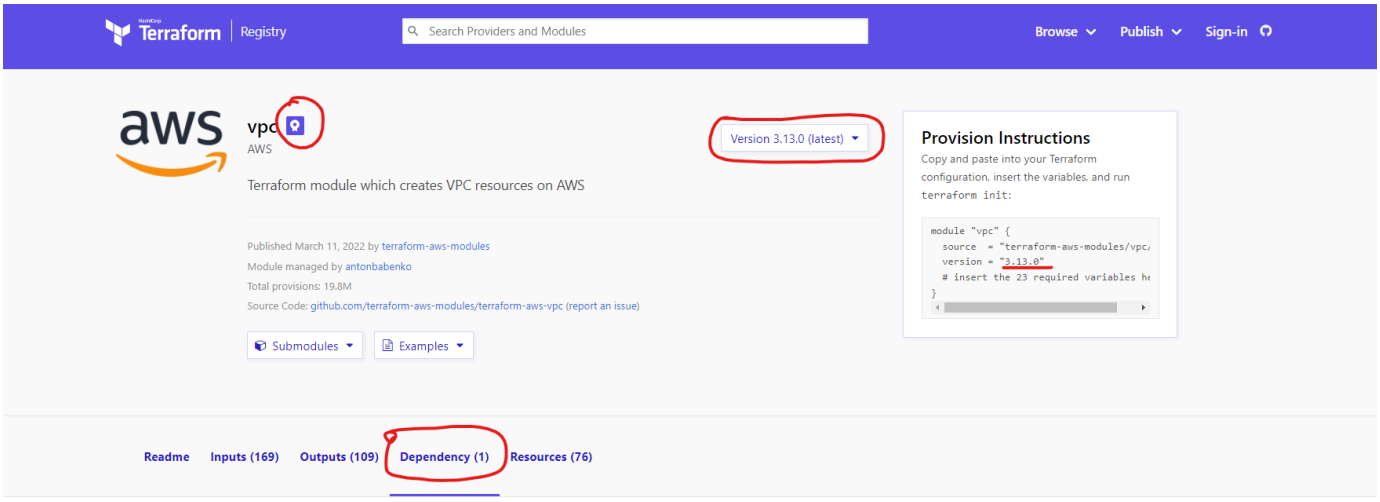
- while using a module from the public registry, make sure it is verified by **Hashicorp** and also set the version because these modules are updated frequently

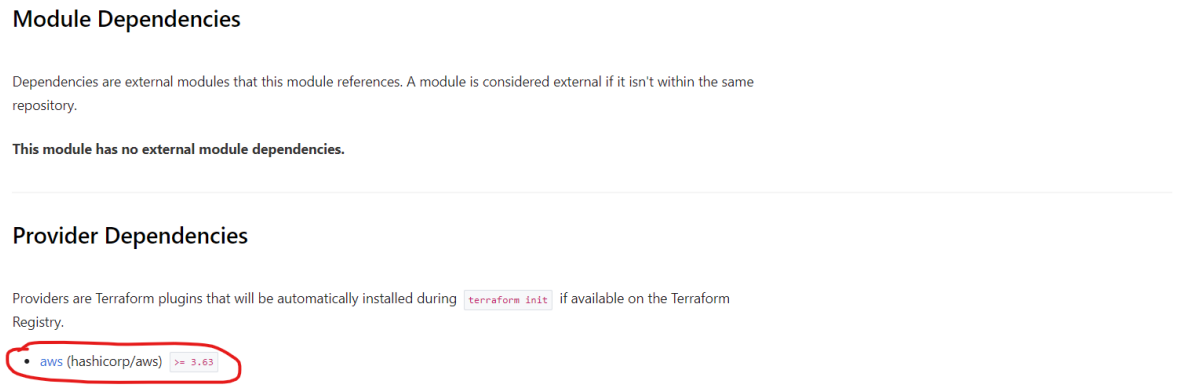
- Choose the provider dependency base on the module version: `source = "git::https://github.com/company_name/terraform-modules//terragrunt/modules/zedcloud-azure?ref=1.0.15"`
- If you build your own custom module, please use the **tag** to specify the **module version** this is because modules are updated frequently by the team
- each module has a source code on Github. https://github.com/terraform-aws-modules/terraform-aws-vpc

https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest

```
# This VPC will not be created
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  version = "2.78.0"
  # version = "~> 2.78"
}
```





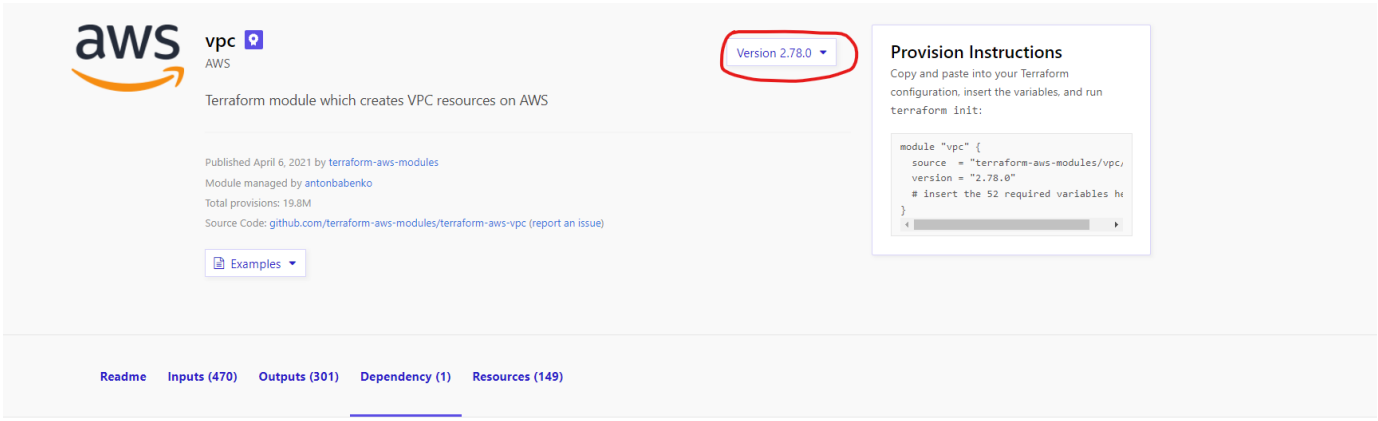
For a module version of `2.78.0` , the provider should be `>= 2.70`

```
# Terraform Block
terraform {
  required_version = "> 0.14"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
```

```
      version = "~> 3.0"
      # version = ">= 2.70"

    }
  }
}

# Provider Block
provider "aws" {
  region  = var.aws_region
  profile = "default"
}
```




```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.78.0"
  # version = "~> 2.78"

  # VPC Basic Details
  name = "vpc-dev"
  cidr = "10.0.0.0/16"
  azs                 = ["us-east-1a", "us-east-1b"]
  private_subnets     = ["10.0.1.0/24", "10.0.2.0/24"]
  public_subnets      = ["10.0.101.0/24", "10.0.102.0/24"]
```

```
  # Database Subnets
  create_database_subnet_group = true
  create_database_subnet_route_table= true
  database_subnets    = ["10.0.151.0/24", "10.0.152.0/24"]

  # NAT Gateways - Outbound Communication
  enable_nat_gateway = true
  single_nat_gateway = true

  # VPC DNS Parameters
  enable_dns_hostnames = true
  enable_dns_support = true

  public_subnet_tags = {
    Type = "public-subnets"
  }

  private_subnet_tags = {
    Type = "private-subnets"
  }

  database_subnet_tags = {
    Type = "database-subnets"
  }

  tags = {
    Owner = "kalyan"
    Environment = "dev"
  }

  vpc_tags = {
    Name = "vpc-dev"
  }
}
```