KPLABS Course

HashiCorp Certified: Terraform Associate

Domain 4

ISSUED BY
Zeal

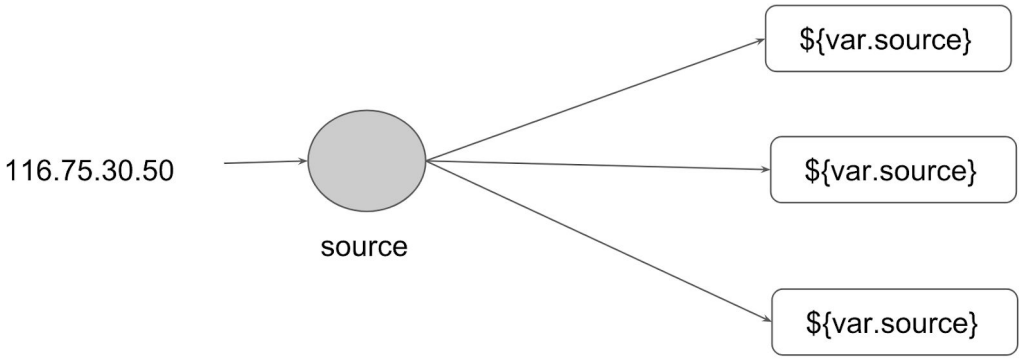
REPRESENTATIVE
instructors@kplabs.in

Domain 4 - Terraform Modules & Workspaces

Module 1: Understanding the DRY principle

In software engineering, don't repeat yourself (DRY) is a principle of software development aimed at reducing repetition of software patterns.

In the earlier lecture, we were making static content into variables so that there can be a single source of information.



1.1 Generic Scenario:

We do repeat multiple times various terraform resources for multiple projects.

Sample EC2 Resource

```
resource "aws_instance" "myweb" {
    ami = "ami-bf5540df"
   instance_type = "t2.micro"
   security_groups = ["default"]
 }
```

Instead of repeating a resource block multiple times, we can make use of a centralized structure.

1.2 Centralized Structure:

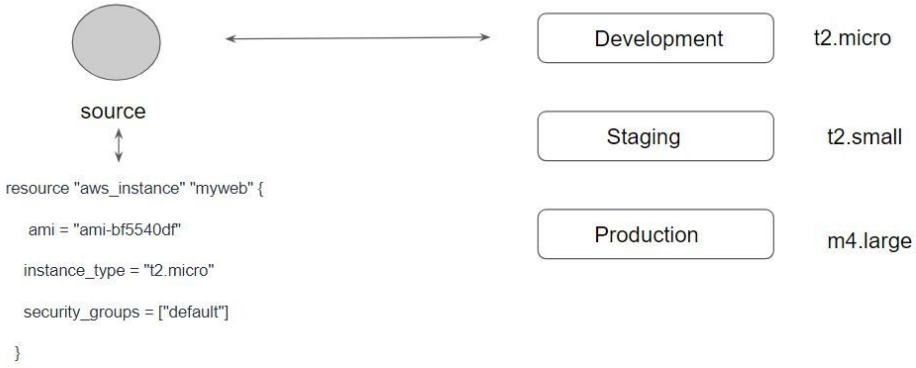
We can centralize the terraform resources and can call out from TF files whenever required.

•

Module 2: Challenges with Terraform Modules

One common need for infrastructure management is to build environments like staging, production with a similar setup but keeping environment variables different.

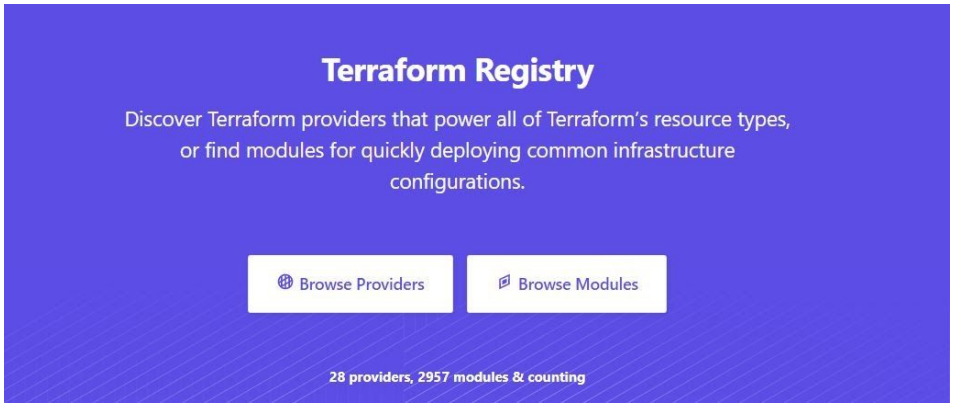
When we use modules directly, the resources will be a replica of code in the module.



Module 3: Terraform Registry

The Terraform Registry is a repository of modules written by the Terraform community.

The registry can help you get started with Terraform more quickly



3.1 Module Location

If we intend to use a module, we need to define the path where the module files are present.

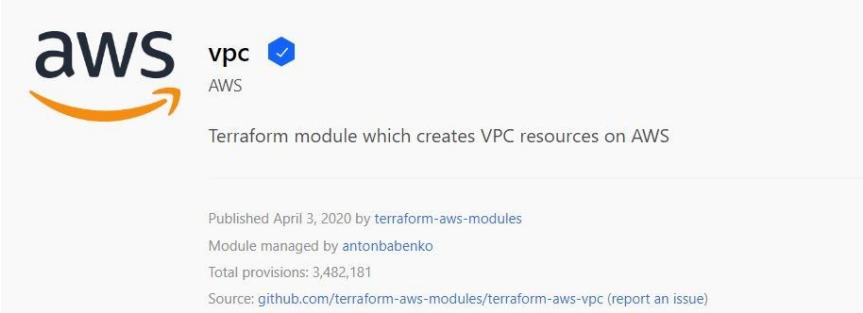
The module files can be stored in multiple locations, some of these include:

- Local Path
- GitHub
- Terraform Registry
- S3 Bucket
- HTTP URLs

3.2 Verified Modules in Terraform Registry

Within Terraform Registry, you can find verified modules that are maintained by various third-party vendors.

These modules are available for various resources like AWS VPC, RDS, ELB, and others



Verified modules are reviewed by HashiCorp and actively maintained by contributors to stay up-to-date and compatible with both Terraform and their respective providers.

The blue verification badge appears next to modules that are verified.

Module verification is currently a manual process restricted to a small group of trusted HashiCorp partners.

3.3 Using Registry Modules in Terraform

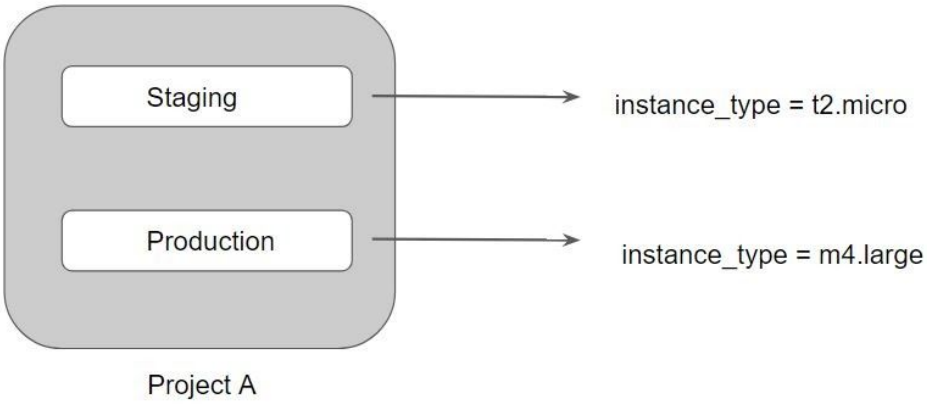
To use Terraform Registry module within the code, we can make use of the source argument that contains the module path.

Below code references to the EC2 Instance module within terraform registry.

```
module "ec2-instance" {
  source  = "terraform-aws-modules/ec2-instance/aws"
  version = "2.13.0"
  # insert the 10 required variables here
}
```

Module 4: Terraform Workspace

Terraform allows us to have multiple workspaces, with each of the workspaces we can have a different set of environment variables associated



Terraform starts with a single workspace named "default".

This workspace is special both because it is the default and also because it cannot ever be deleted.

If you've never explicitly used workspaces, then you've only ever worked on the "default" workspace.

Workspaces are managed with the terraform workspace set of commands.

To create a new workspace and switch to it, you can use terraform workspace new; to switch workspaces you can use terraform workspace select; etc.