

**KPLABS Course**

**HashiCorp Certified: Terraform Associate**

**Domain 1**

## **ISSUED BY**

Zeal

## **REPRESENTATIVE**

[instructors@kplabs.in](mailto:instructors@kplabs.in)

# 

# **Domain 1 - Deploying Infrastructure with Terraform**

## 

## **Module 1: Provider and Resources**

1.1 Overview of Providers:

Terraform supports multiple providers.

We have to specify the provider details for which we want to launch the infrastructure for.

With the provider, we also have to add the tokens which will be used for authentication.

On adding a provider, terraform init will download plugins associated with the provider.

Example:

* AWS Access & Secret Keys.
* Digital Ocean Tokens.

1.2 Overview of Resources

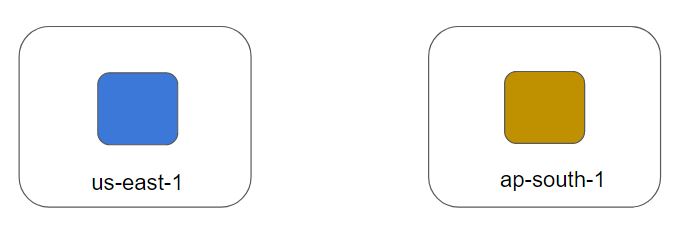
Resources are the reference to the individual services which the provider has to offer

Example:

* resource aws\_instance
* resource aws\_alb
* resource iam\_user
* resource digitalocean\_droplet

1.3 Dealing with Multiple Providers

* You have two tf files within a directory first\_ec2.tf and second\_ec2.tf
* You want first\_ec2.tf to launch in the us-east-1 region.
* You want second\_ec2.tf to launch in ap-south-1 region

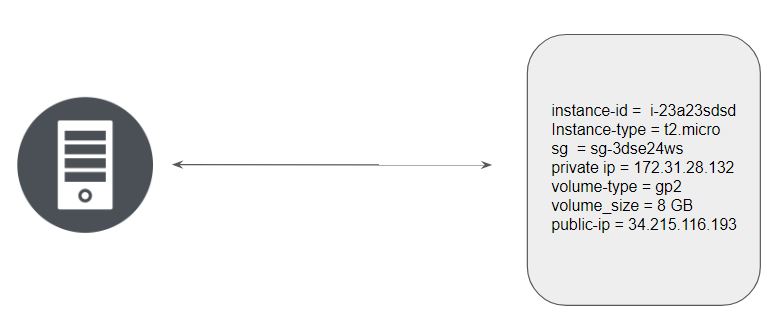


## **Module 2: Terraform State File**

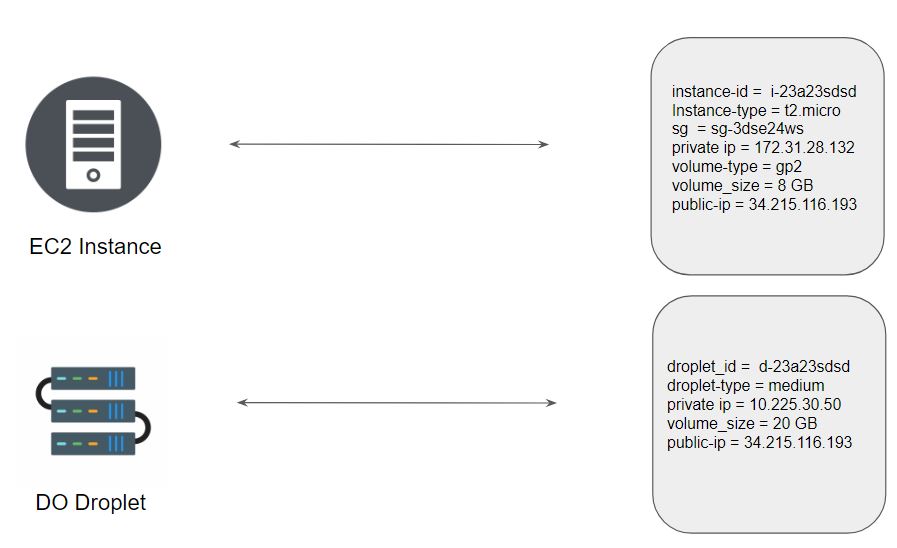
2.1 Overview of State Files:

Terraform stores the state of the infrastructure that is being created from the TF files.

This state allows terraform to map real-world resource to your existing configuration.



Multiple resources in Terraform will have a separate block with the state file.



2.2 Current State vs Desired State

When running a terraform plan, Terraform must know the current state of resources in order to effectively determine the changes that it needs to make to reach your desired configuration.

Current State = Current Infrastructure Resource & Configuration

Desired State = Infrastructure Configuration defined within the Terraform TF Files.

Terraform will plan to match the desired state to the current state. If there is a difference between both, the desired state will take the preference.

**Module 3 Destroying Infrastructure With Terraform**

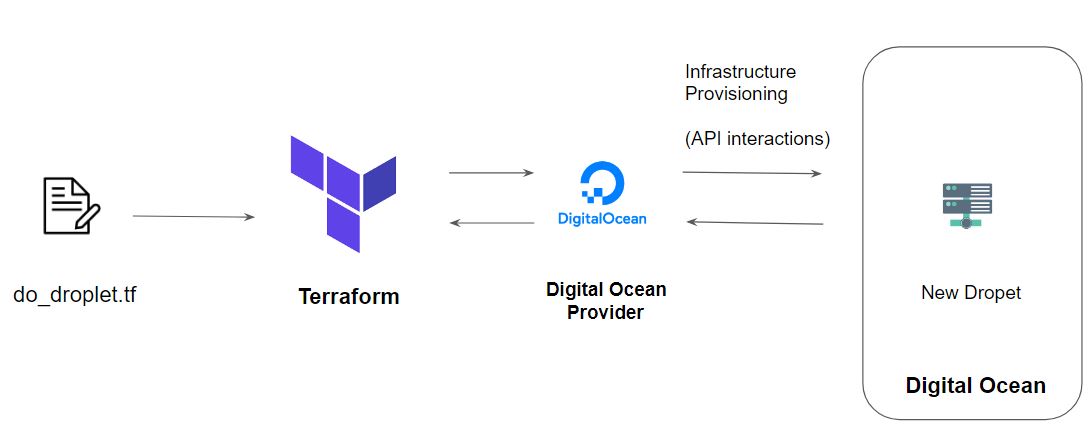
The terraform destroy command is used to destroy the Terraform-managed infrastructure.

This will ask for confirmation before destroying it

If -auto-approve is set, then the destroy confirmation will not be shown..

**Module 4 Provider Versioning**

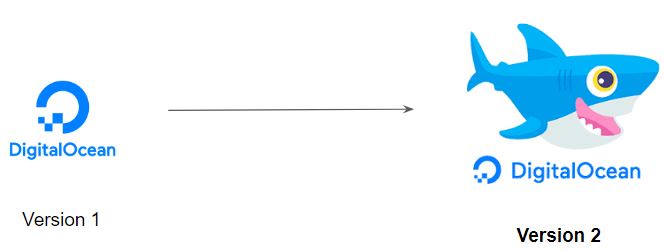
4.1 Overview of Provider Architecture



4.2 Provider Versioning

Provider plugins are released separately from Terraform itself.

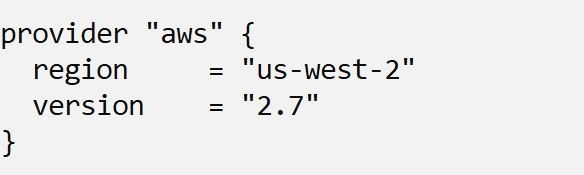
They have a different set of version numbers.



4.3 Explicitly Setting Provider Version

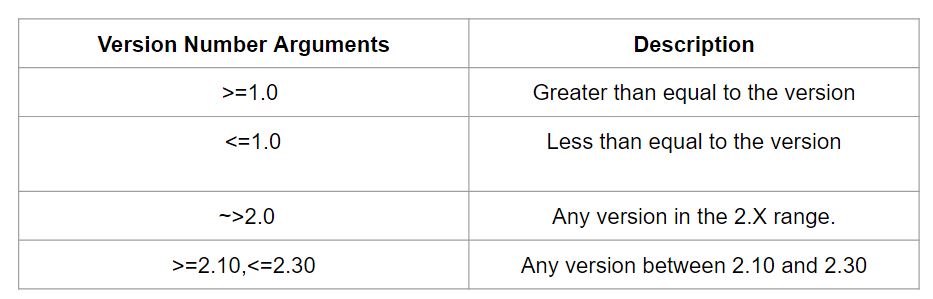
During terraform init, if version argument is not specified, the most recent provider will be downloaded during initialization.

For production use, you should constrain the acceptable provider versions via configuration, to ensure that new versions with breaking changes will not be automatically installed



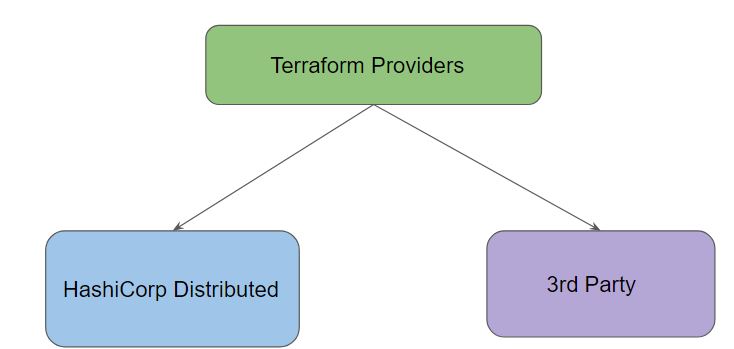
4.4 Arguments for Specifying the provider

There are multiple ways of specifying the version of a provider.



**Module 5 Types of Terraform Providers**

There are two major categories for terraform providers.

****

HashiCorp Distributed providers can be downloaded automatically during terraform init.

terraform init cannot automatically download providers that are not distributed by HashiCorp

5.1 Overview of 3rd Party Providers

It can happen that the official provider does not support specific functionality.

Some organizations might have their proprietary platform for which they want to use Terraform.

For such cases, individuals can decide to develop/use 3rd party providers.

Configuring 3rd Party Provider

Third-party providers must be manually installed, since terraform init cannot automatically download them.

Install third-party providers by placing their plugin executables in the user plugins directory.

