



# INTRODUCTION TO TERRAFORM



**SONALI GOEL**  
Senior Engineer @Yoox-Net-a-porter



# Agenda



01

## What is Terraform ?

Terraform is an open-source Infrastructure as Code (IaC) tool created by HashiCorp

02

## Why IAC ?

IaC allows you to manage and provision infrastructure using code.

03

## Key features of Terraform

Details about some important features.

04

## Terraform Providers

Terraform Providers are responsible for interacting with APIs and exposing resources for a particular platform (e.g., AWS, Azure, GCP).

05

## Terraform Workflow

The Terraform workflow consists of several key steps that users typically follow when working with Terraform to manage infrastructure resources.

06

## Getting Started: A Simple Example

Basic sample explaining the syntax and structure.



# What is Terraform ?

Language ??



New Cloud Provider  
??

FRamework ??



# Terraform a Tool



Terraform is an infrastructure as code software tool. While Terraform does have its own syntax and configuration language for defining infrastructure, it primarily functions as a tool for managing and provisioning infrastructure resources across various cloud providers and services.



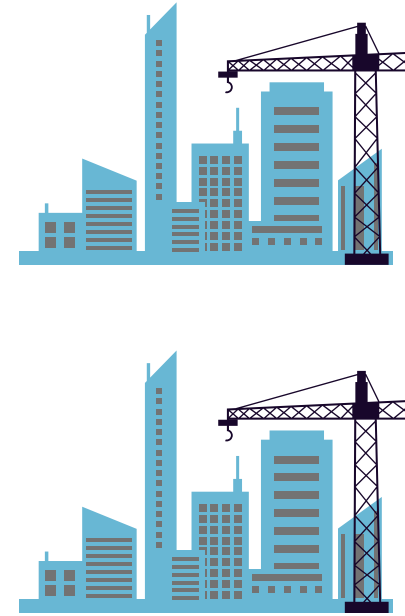
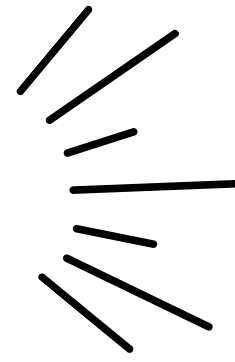
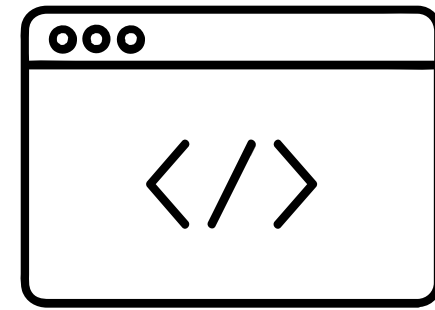
Terraform creates and manages resources on cloud platforms and other services through their application programming interfaces (APIs). Providers enable Terraform to work with virtually any platform or service with an accessible API.



It is an open-source tool uses its own declarative language called HashiCorp Configuration Language (HCL).



# Why IaC ?



**Consistency**

**Efficiency &  
Automation**

**Scalability &  
Agility**

**Version  
Control &  
Change  
Mgmt**

**Collaboration  
& Document**

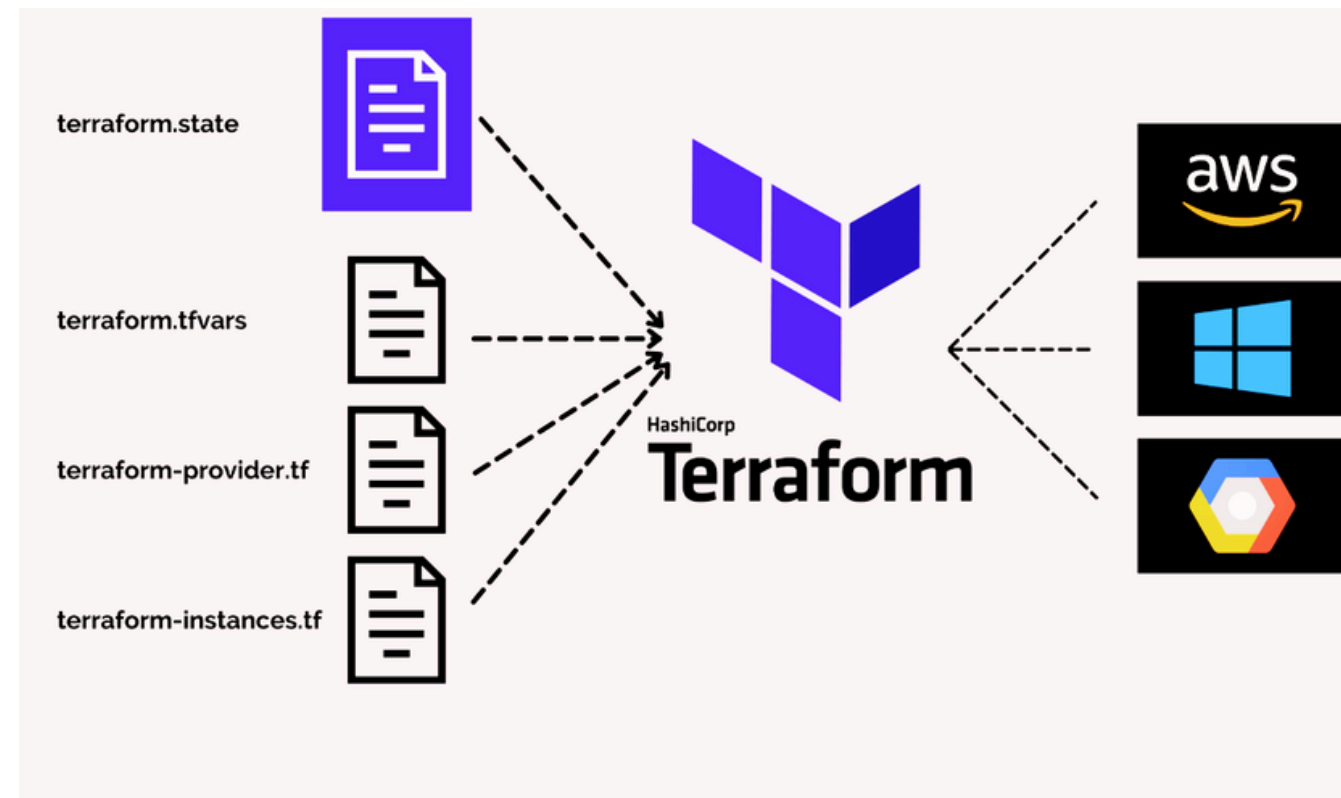


# Terraform Features

Cloud Provider  
Agnostic

Easy integration with  
tools

Parallel Environment  
Setup in a go



Declarative  
Programming

Enables Immutability

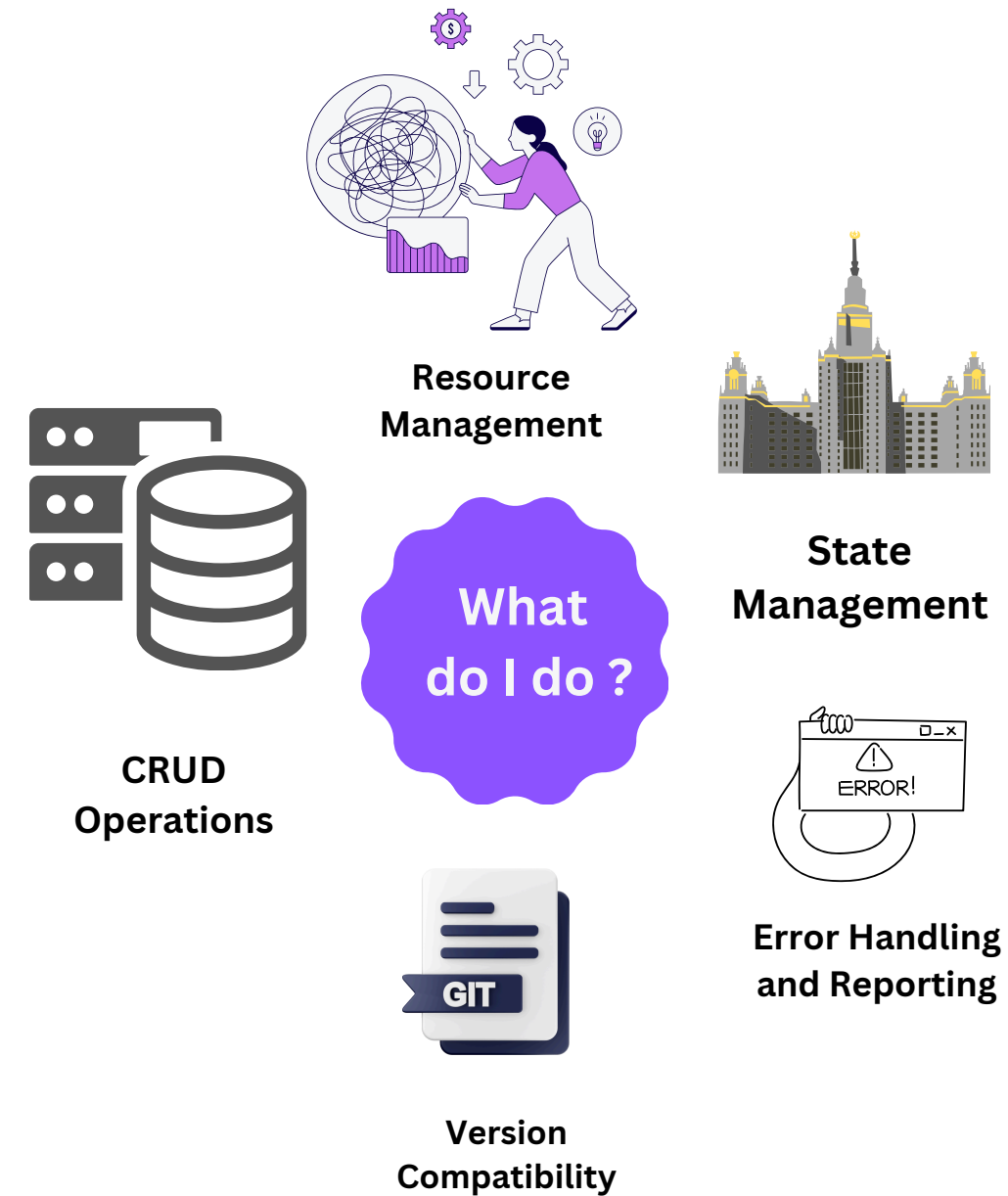
Open-Source

Most Popular



# Terraform Providers

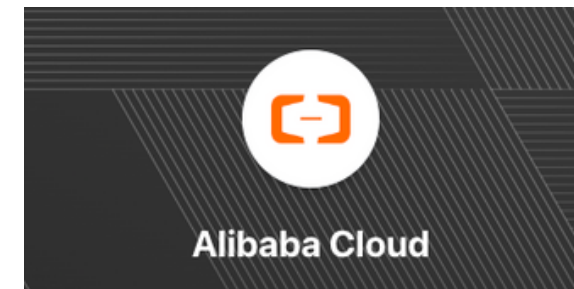
Terraform provider acts as an interface between Terraform and the APIs of various infrastructure platforms, SaaS-Service, allowing Terraform to create, update, and delete resources within those platforms.





# Terraform Providers

Some “Famous” terraform providers



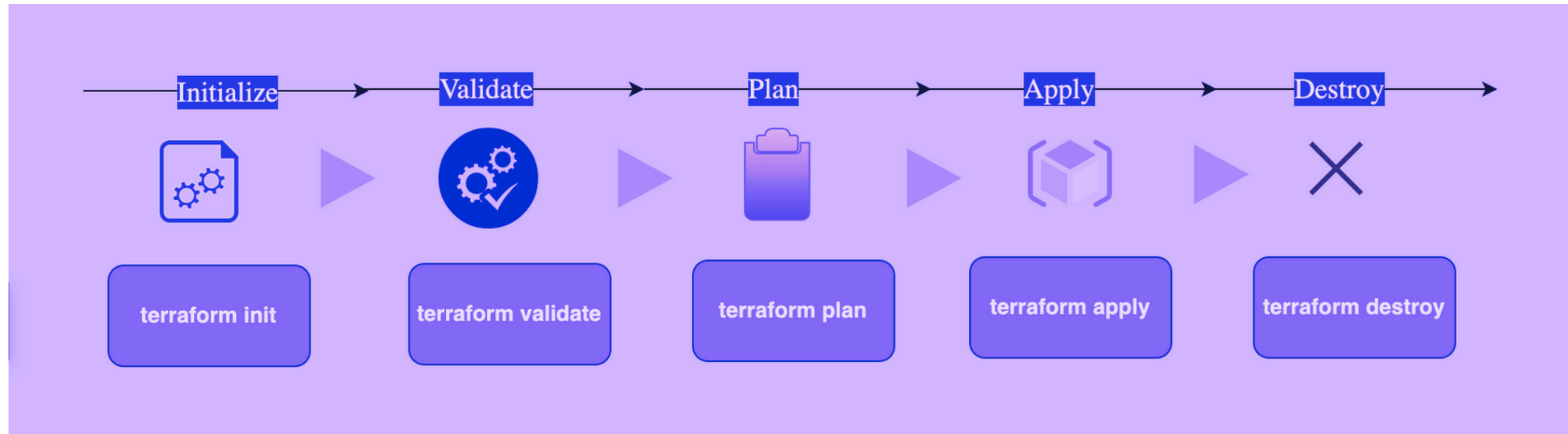
Link to the [Terraform Providers](#)





# Terraform Workflow

To provision infrastructure using terraform, you need to follow 5 core commands known as “Terraform Workflow”  
And they are !!!





# Demo

In this demo, Will use the AWS provider to provision a S3 bucket and place a file in the bucket via Terraform

```
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

terraform-aws-infrastructure % terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_s3_bucket.s3_bucket will be created
+ resource "aws_s3_bucket" "s3_bucket" {
+   acceleration_status      = (known after apply)
+   acl                      = (known after apply)
+   arn                     = (known after apply)
+   bucket                  = "tf-aws-bucket-25-03-24"
+   bucket_domain_name      = (known after apply)
+   bucket_prefix           = (known after apply)
+   bucket_regional_domain_name = (known after apply)
+   force_destroy           = false
+   hosted_zone_id         = (known after apply)
+   id                     = (known after apply)
+   object_lock_enabled     = (known after apply)
+   policy                  = (known after apply)
+   region                 = (known after apply)
+   request_payer           = (known after apply)
+   tags                    = {
+     "Name" = "TF S3 bucket"
+   }
+   tags_all                = {
+     "Name" = "TF S3 bucket"
+   }
+   website_domain          = (known after apply)
+   website_endpoint        = (known after apply)
}

# aws_s3_object.object will be created
+ resource "aws_s3_object" "object" {
+   acl                = (known after apply)
+   arn                = (known after apply)
+   bucket             = "tf-aws-bucket-25-03-24"
+   bucket_key_enabled = (known after apply)
+   checksum_crc32    = (known after apply)
+   checksum_crc32c   = (known after apply)
+   checksum_sha1     = (known after apply)
+   checksum_sha256   = (known after apply)
+   content_type       = (known after apply)
+   etag              = (known after apply)
+   force_destroy      = false
+   id                = (known after apply)
+   key               = "introduction.txt"
+   kms_key_id        = (known after apply)
+   server_side_encryption = (known after apply)
+   source             = "resources/introduction.txt"
+   storage_class     = (known after apply)
+   tags_all          = (known after apply)
+   version_id        = (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
```

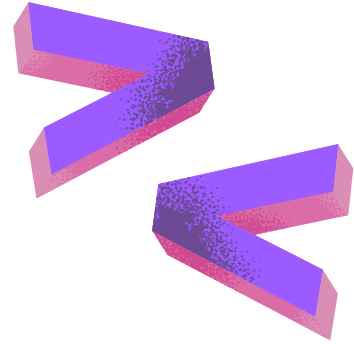


# References

- <https://www.terraform.io/>
- <https://www.terraform-best-practices.com/>

## Source Code

- <https://github.com/goelsonali/terraform-aws-infrastructure>



THANK  
YOU

