**What Is Terraform?**

**Terraform** is one of the most popular **Infrastructure-as-code (IaC) tool**, used by DevOps teams to automate infrastructure tasks. It is used to automate the provisioning of your cloud resources. Terraform is an open-source, cloud-agnostic provisioning tool developed by HashiCorp and written in GO language.



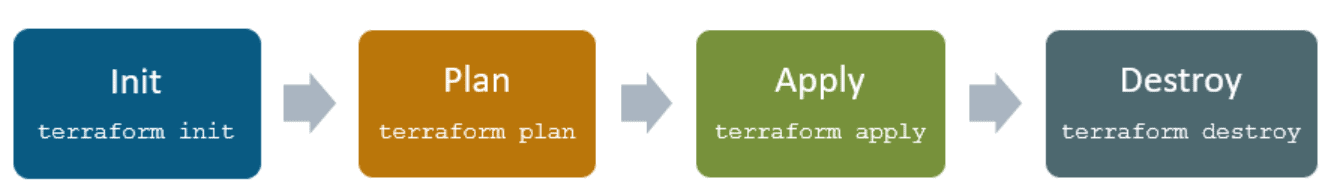
**Benefits of using Terraform:**

* Does orchestration, not just configuration management
* Supports multiple providers such as AWS, Azure, Oracle, GCP, and many more
* Provide immutable infrastructure where configuration changes smoothly
* Uses easy to understand language, HCL (HashiCorp configuration language)
* Easily portable to any other provider

Check out our blog for everything you need to know about Terraform

**Terraform Lifecycle**

Terraform lifecycle consists of – **init**, **plan**, **apply**, and **destroy**.



1. **Terraform init** initializes the (local) Terraform environment. Usually executed only once per session.

2. **Terraform plan** compares the Terraform state with the as-is state in the cloud, build and display an  
execution plan. This does not change the deployment (read-only).

3. **Terraform apply** executes the plan. This potentially changes the deployment.

4. **Terraform destroy** deletes all resources that are governed by this specific terraform environment.

**Terraform Core Concepts**

**1. Variables**: Terraform has input and output variables, it is a key-value pair. Input variables are used as parameters to input values at run time to customize our deployments. Output variables are return values of a terraform module that can be used by other configurations.

**2. Provider**: Terraform users provision their infrastructure on the major cloud providers such as AWS, Azure, OCI, and others. A *provider* is a plugin that interacts with the various APIs required to create, update, and delete various resources.

**3. Module**: Any set of Terraform configuration files in a folder is a *module*. Every Terraform configuration has at least one module, known as its ***root module.***

**4. State**: Terraform records information about what infrastructure is created in a Terraform *state* file. With the state file, Terraform is able to find the resources it created previously, supposed to manage and update them accordingly.

**5. Resources**: Cloud Providers provides various services in their offerings, they are referenced as Resources in Terraform. Terraform resources can be anything from compute instances, virtual networks to higher-level components such as DNS records. Each resource has its own attributes to define that resource.

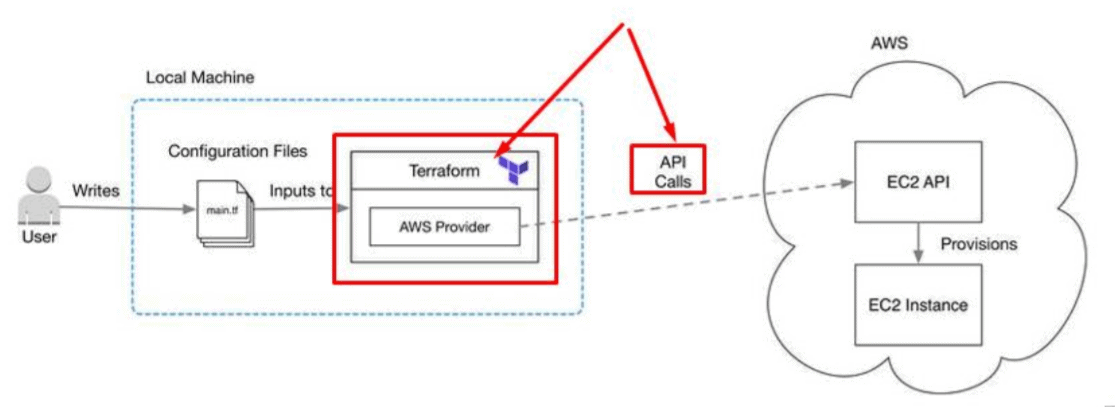
**6. Data Source**: Data source performs a read-only operation. It allows data to be fetched or computed from resources/entities that are not defined or managed by Terraform or the current Terraform configuration.

**7. Plan**: It is one of the stages in the Terraform lifecycle where it determines what needs to be created, updated, or destroyed to move from the real/current state of the infrastructure to the desired state.

**8. Apply**: It is one of the stages in the Terraform lifecycle where it applies the changes real/current state of the infrastructure in order to achieve the desired state.

**Terraform Providers**

A provider is responsible for understanding API interactions and exposing resources. It is an executable plug-in that contains code necessary to interact with the API of the service. Terraform configurations must declare which providers they require so that Terraform can install and use them.

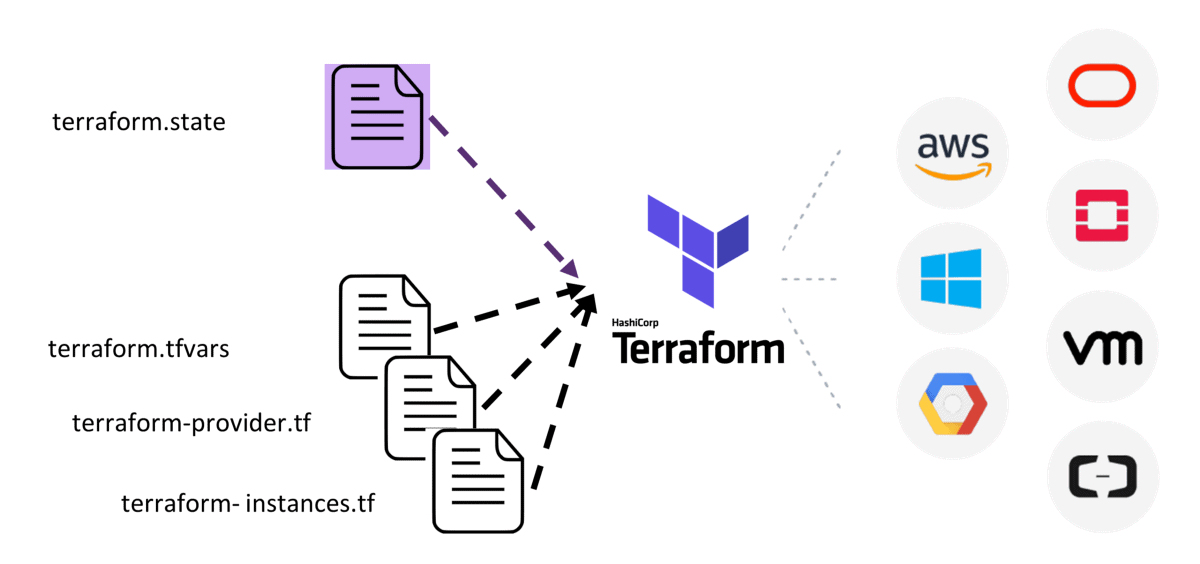


Terraform has over a hundred providers for different technologies, and each provider then gives terraform user access to its resources. So through AWS provider, for example, you have access to hundreds of AWS resources like EC2 instances, the AWS users, etc.

**Read More:**About [**Terraform Workflow**](https://k21academy.com/terraform-iac/terraform-workflow-and-its-use-case/).

**Terraform Configuration Files**

Configuration files are a set of files used to describe infrastructure in Terraform and have the file extensions **.tf** and **.tf.json**. Terraform uses a declarative model for defining infrastructure. Configuration files let you write a configuration that declares your desired state. Configuration files are made up of resources with settings and values representing the desired state of your infrastructure.



A Terraform configuration is made up of one or more files in a directory, provider binaries, plan files, and state files once Terraform has run the configuration.

**1. Configuration file (\*.tf files):** Here we declare the provider and resources to be deployed along with the type of resource and all resources specific settings

**2. Variable declaration file (variables.tf or variables.tf.json):** Here we declare the input variables required to provision resources

**3. Variable definition files (terraform.tfvars):** Here we assign values to the input variables

**4. State file (terraform.tfstate):** a state file is created once after Terraform is run. It stores state about our managed infrastructure.