Intro to Infrastructure as Code

Whats is Infrastructure as Code?

Infrastructure as code (IaC) grants the ability to **provision and support** infrastructure using code instead of a manual processes

Manual infrastructure management is **time-consuming** and **prone to errors**

IaC has two approaches

- Declarative
- Imperative

Why do we need it?

- Repeatability
- Automation
- Infrastructure Tracking
- Standardization
- Shareability

IaC Tools

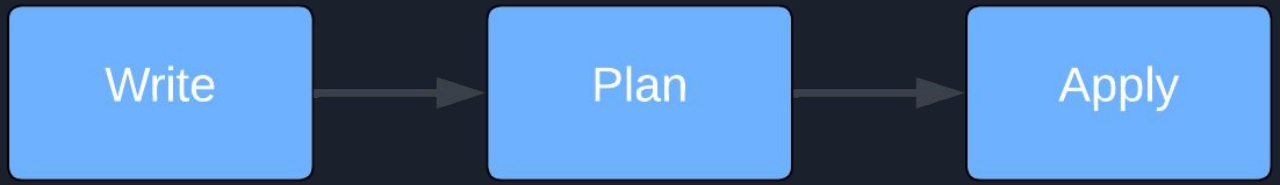
- Terraform
  - OpenTofu
- Pulumi
- CloudFormation
- Ansible

Terraform

Terraform is a tool that lets you define infrastructure in human and machine-readable code

It focuses on the higher-level abstraction of the associated services

The Terraform workflow:

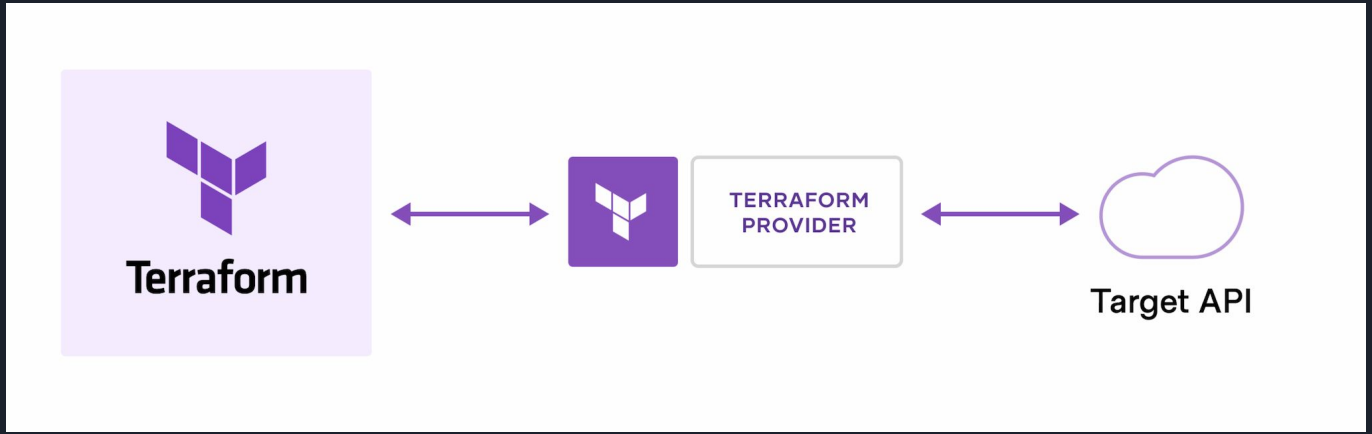


Installation: https://developer.hashicorp.com/terraform/install

Providers

Providers allow Terraform to interact with cloud providers, SaaS providers, and other APIs

AWS, Azure, GCP, Kubernetes, local resources, etc.



Providers

```
terraform {
  required_providers {
    mycloud = {
      source  = "mycorp/mycloud"
      version = "1.0.0"
    }
  }
}

provider "mycloud" {
  key = "value"
}
```

Resources

Each resource block describes one or more infrastructure objects, such as virtual networks, instances, or local file

- The combination of resource type and name must be unique
- Arguments are defined inside of the resources body

```
resource "local_file" "example" {
    content  = "Contents of the file example.txt"
    filename = "~/example.txt"
}
```

Resources

Resources output data associated with them

```
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}
```

One of the attributes of the resource **example** is the **id** of the created VPC

It would be referenced like **aws_vpc.example.id**

Working with State

Terraform must store state about your managed infrastructure and configuration

The **state** is used to map Terraform resources into external resources

The state can be **local** or **remove**

- Local file
- Object Store (S3, Azure Blob, etc.)
- Terraform Cloud

Working with State

```
terraform {
  backend "local" {
    path = "relative/path/to/terraform.tfstate"
  }
}
```

Using Existing Data

When dealing with resources not managed by Terraform you can make requests to fetch data associated with them

The data call queries AWS for the secret **developer-secret-sauce** and retrieve the associated metadata & secret values

```
data "aws_secretsmanager_secret_version" "dev_secret" {
  secret_id = "developer-secret-sauce"
}
```

On the Command Line

The **terraform** command has the following subcommands

- **init -** Initializes the working directory for Terraform. It pulls code used by providers
- **plan -** Preview the differences between the remote state and local configuration
- **apply -** Executes the changes defined in the plan
- **destroy -** Deletes all resources

On the Command Line

- **validate -** Verifies the configuration is valid
- **fmt -** Formats the Terraform files
- **state**
  - **list -** Shows all resources managed by Terraform
  - **rm -** Removes a resources from being managed Terraform
  - **import -** Specify an existing resource not managed by Terraform and link it to a Terraform resources

Other Concepts

- **Modules**
- **Variables**
- **Outputs**

Demo

Follow along
https//iac.infrasec.sh

Questions

Additional Resources

- https://github.com/andrew-aiken/website-ref/tree/main/intro_to_iac
- https://developer.hashicorp.com/terraform/
- https://developer.hashicorp.com/terraform/tutorials/certification-003/associate-study-003

Sources

- https://aws.amazon.com/what-is/iac/
- https://carbon.now.sh/
- https://developer.hashicorp.com/terraform/tutorials/