



Infrastructure Automation with Terraform

Ninh Nguyen— DevOps

Agenda

1. What is Terraform?

- Provider
- Backend & TF-State
- Workspace
- Modules
- Workflows

2. Basic networking [AWS, Azure, GCP]

3. Hands-on labs

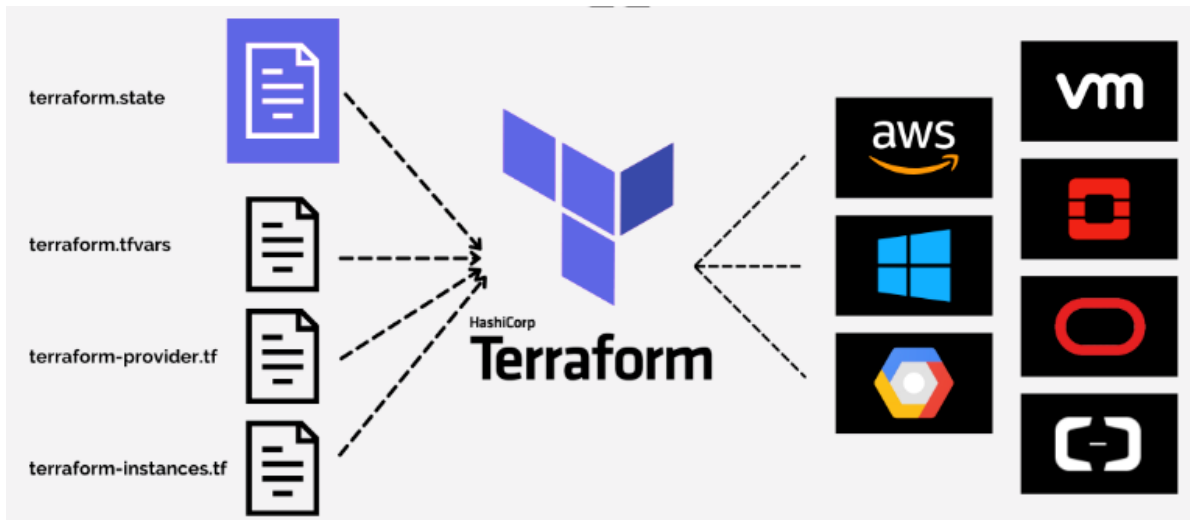
- AWS
- Azure
- GCP

What is Terraform?

HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share.

You can then use a consistent workflow [plan,apply,destroy] to provision and manage all of your infrastructure throughout its lifecycle.

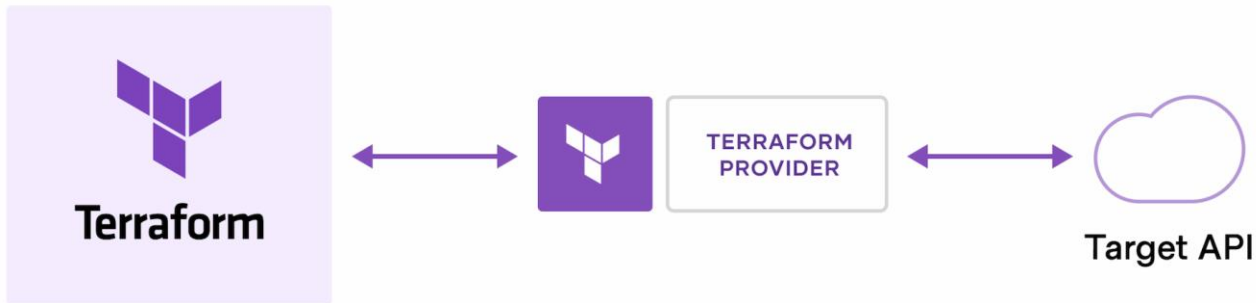
Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.



Terraform Provider

Providers enable Terraform to work with virtually any platform or service with an accessible API.

HashiCorp and the Terraform community provide thousands of providers on the Terraform Registry, including Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, DataDog, and many more.



Terraform Provider

Resources that don't set the provider meta-argument will use the default provider configuration that matches the first word of the resource type name. (For example, an `aws_instance` resource uses the default aws provider configuration unless otherwise stated.)

```
# setup google terraform provider
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
      version = "4.74.0"
    }
  }
}
```

```
# The default provider configuration; resources that begin with `aws_`
# will use it as the default, and it can be referenced as `aws`.
provider "aws" {
  region = "us-east-1"
}
```

```
# Additional provider configuration for west coast region; resources can
# reference this as `aws.west`.
provider "aws" {
  alias = "west"
  region = "us-west-2"
}
```

```
module "aws_vpc" {
  source = "./aws_vpc"
  providers = {
    aws = aws.west
  }
}
```

Terraform Backend

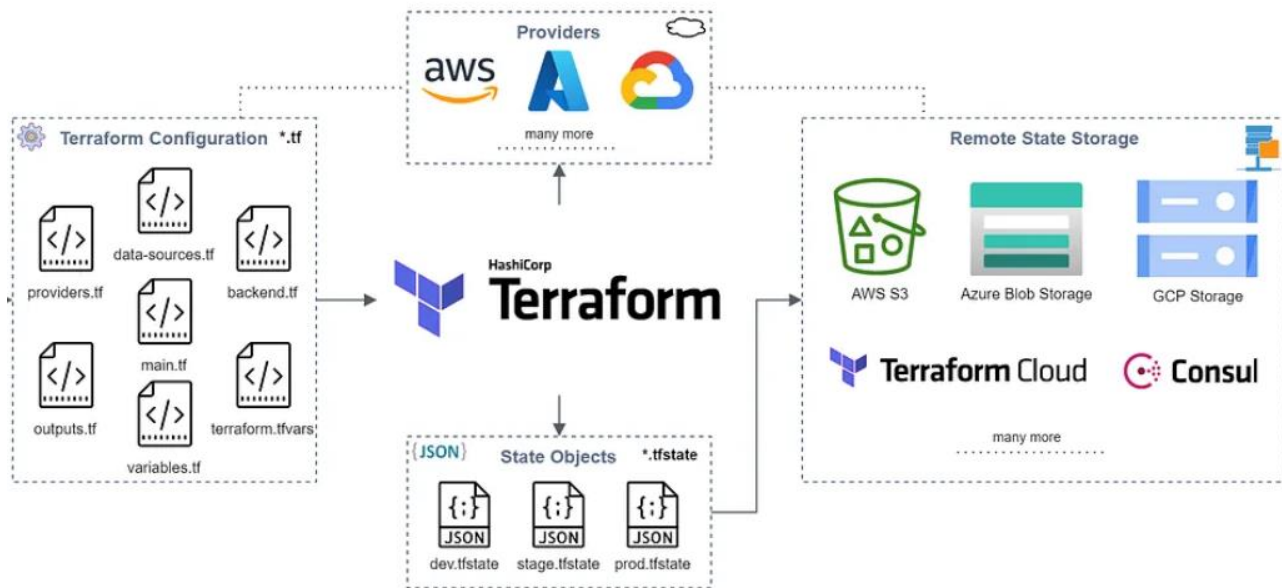
- A backend defines where Terraform stores its state data files.
- Terraform uses persisted state data to keep track of the resources it manages

```
terraform {  
  
  backend "s3" {  
    bucket = "demo12mar"  
    key    = "demo/test-state"  
    region = "eu-central-1"  
  }  
  
  required_version = ">= 1.0.0"  
  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = ">= 4.9.0"  
    }  
  
    kubernetes = {  
      source = "hashicorp/kubernetes"  
      version = ">= 2.7.1"  
    }  
  
    tls = {  
      source = "hashicorp/tls"  
      version = ">= 3.1.0"  
    }  
  
    null = {  
      source = "hashicorp/null"  
      version = ">= 2.0"  
    }  
  }  
}
```

Terraform State

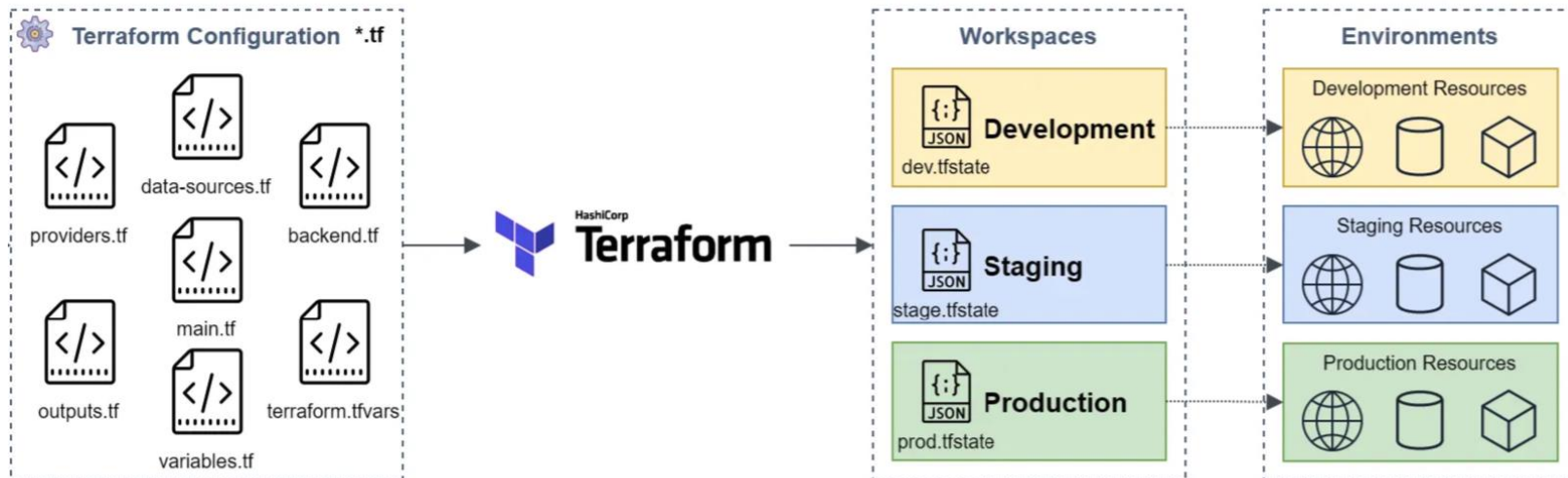
Terraform must store state about your managed infrastructure and configuration. This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.

This state is stored by default in a local file named "terraform.tfstate"



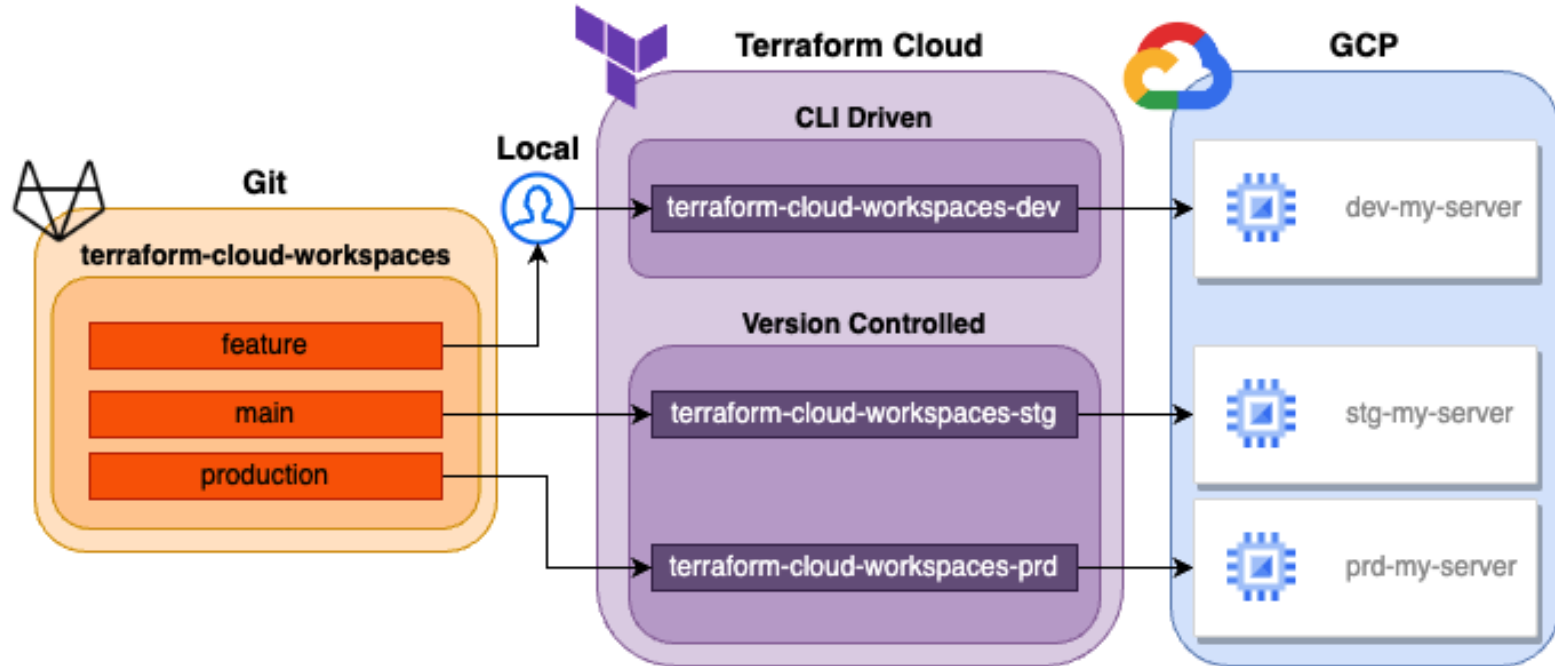
Terraform Workspace

Terraform workspace run locally



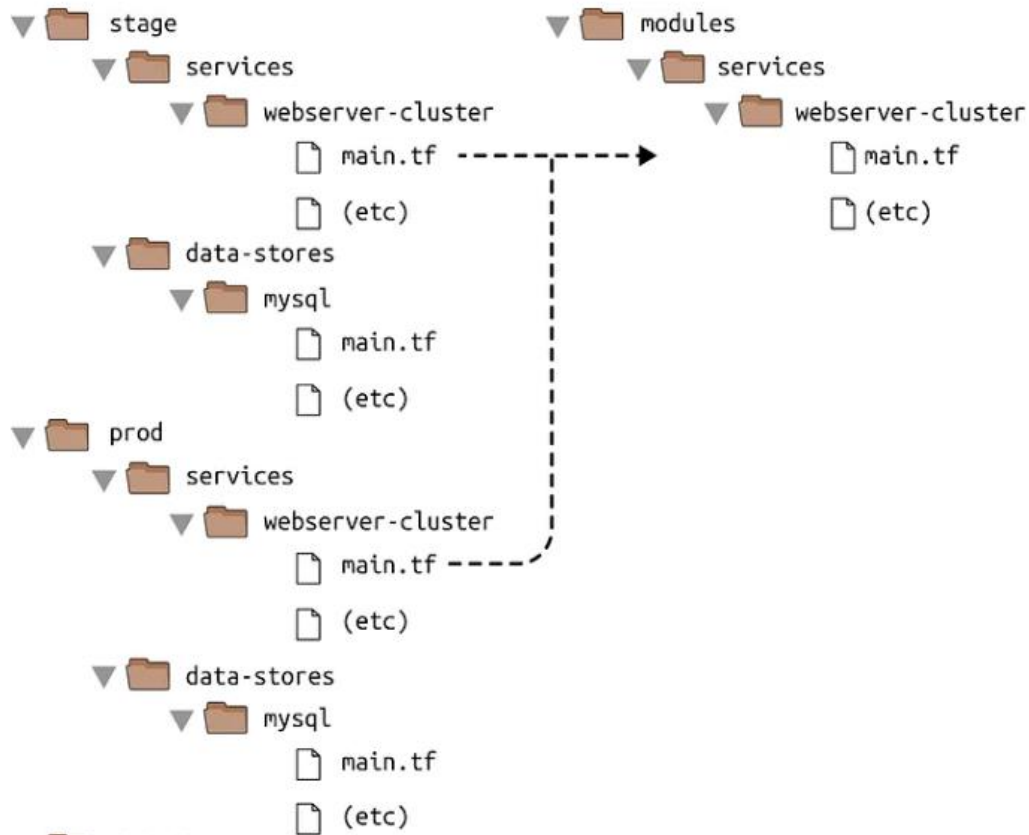
Terraform Workspace

Terraform Cloud workspace



Terraform Modules

With Terraform, you can put your code inside of a Terraform module and reuse that module in multiple places throughout your code. Instead of having the same code copied and pasted in the staging and production environments



Terraform Modules

```
module "eks" {
  source = "terraform-aws-modules/eks/aws"
  version = "~> 20.0"

  cluster_name      = "my-cluster"
  cluster_version   = "1.30"

  cluster_endpoint_public_access = true

  cluster_addons = {
    coredns          = {}
    eks-pod-identity-agent = {}
    kube-proxy       = {}
    vpc-cni           = {}
  }

  vpc_id          = "vpc-1234556abcdef"
  subnet_ids      = ["subnet-abcde012", "subnet-bcde012a", "subnet-fghi345a"]
  control_plane_subnet_ids = ["subnet-xyzde987", "subnet-slkjf456", "subnet-qeiru789"]

  # EKS Managed Node Group(s)
```

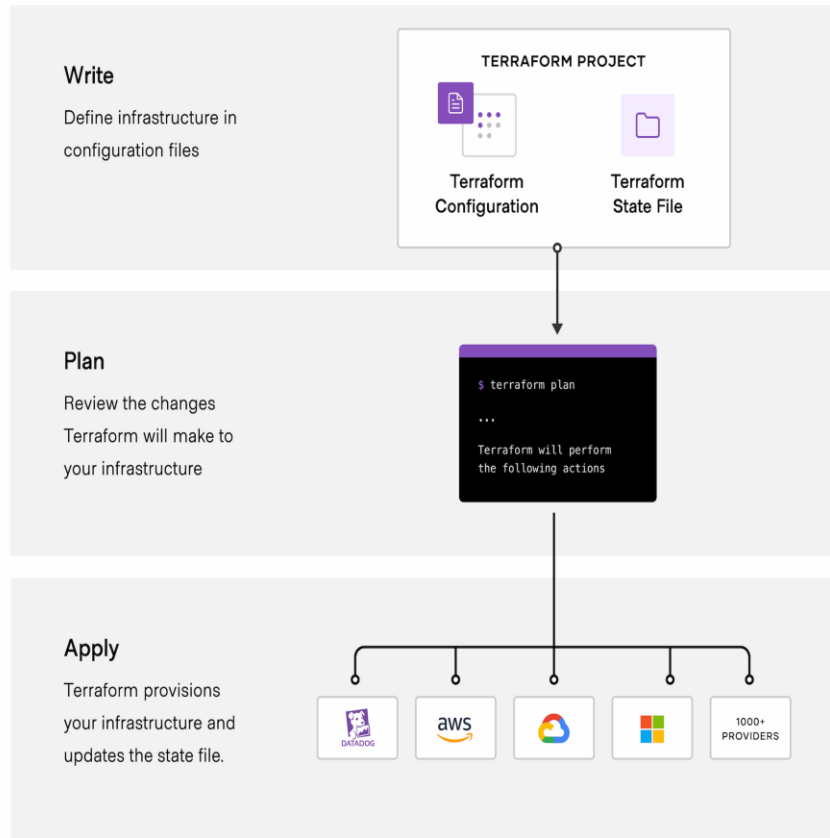
Terraform Workflows

The core Terraform workflow consists of three stages:

Write - Author infrastructure as code.

Plan - Preview changes before applying.

Apply - Provision reproducible infrastructure.






Terraform Alternatives

	Terraform Cloud	Atlantis	Spacelift	Env0	Scalr	Cloudify
On-Premise Support	✓	✓	✗	✗	✓	★
Non-TF IaC Support	✗	✗	✓	✓	✗	★
Policy Enforcement	✓	✓	✓	✓	★	✓
Chained Workspaces	✗	✗	★	✓	✓	✗
Terraform Module Registry	✓	✗	★	✓	✓	✓
Self-Service Infrastructure	✓	✗	✗	✓	✓	✓
Customizable Build Steps	⚠	✓	✓	★	✓	★
Cost Calculation	✓	✗	✓	★	✓	✓
Infrastructure Visibility	✗	✗	★	✗	✗	✓

Hands-on lab [AWS, Azure, GCP]



Basic networking [AWS, Azure, GCP]

	 AWS	 Azure	 Google Cloud
Network Engine (SDN)	Custom	Azure Virtual Network - VNet	Andromeda
Virtual Private Cloud	VPC	VNet	VPC
VPC Span	Single Region - Multi AZ	Single Region - Multi AZ	Multiple Regions
Subnet range	/16 to /28	/8 to /29	/8 to /29
Public IP Addresses	Elastic IP	Public IP Address	Static and Ephemeral IP

Hands-on lab [GCP]

Setup credentials

- Create GCP project
- Create GCP service account for the project
- Generate the JSON key from service account
- Authenticate with GCP APIs using the JSON key

```
gcloud auth activate-service-account --key-file ./devops-425305-23223eb04844.json --project devops-425305
export GOOGLE_CREDENTIALS="./devops-425305-23223eb04844.json"
export GOOGLE_PROJECT="devops-425305"
gcloud container clusters get-credentials "gke-cluster" --zone "us-west1" --project devops-425305
```

Hands-on lab [Azure]

Setup credentials

- Create subscription
- Create AD service principal

```
az ad sp create-for-rbac --sdk-auth --role="Contributor" --scopes="/subscriptions/xx-xxxx-" -n "devops-aks"
```

```
export ARM_CLIENT_ID="9574e97f-5977-4586-8523-8187cca600c8"
```

```
export ARM_CLIENT_SECRET="JRS8Q~8.SNdueYkajlaGrHJGPQU6yhql8RiJLc9y"
```

```
export ARM_TENANT_ID="9236a7c7-8daa-4463-88a1-f15361a1b33f"
```

```
export ARM_SUBSCRIPTION_ID="8cddfaf8-2aa6-4758-9dc3-f5173af21ac3"
```

```
az aks get-credentials --resource-group devops-aks --name devops-aks
```

Hands-on lab [AWS]

Setup credentials

```
#cat ~/.aws/credentials
[vuninhnguyen]
aws_access_key_id = AKIAXXXXXXXXXXXXX
aws_secret_access_key = Nc3rXXXXXXXXXXXXXXXXX
region = us-east-1
output = json
```

```
AWS_PROFILE=vuninhnguyen terraform plan -var-file=sample.tfvars --out terraform.plan
AWS_PROFILE=vuninhnguyen terraform apply "terraform.plan"
AWS_PROFILE=vuninhnguyen terraform destroy -var-file=sample.tfvars
aws eks update-kubeconfig --name devops-eks --region us-east-2 --profile vuninhnguyen
```



Q&A