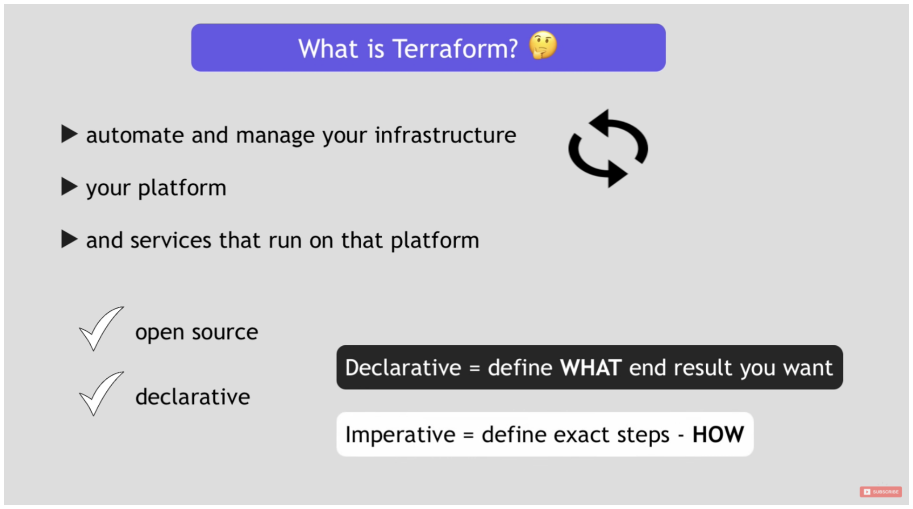
Terraform

HashiCorp Terraform

Overview



What is Terraform? 🤔

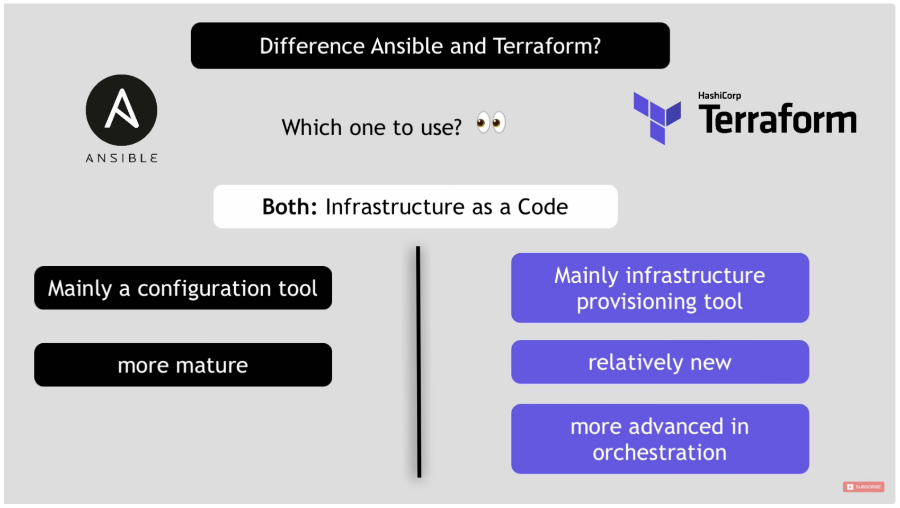
- ▶ automate and manage your infrastructure
- ▶ your platform
- ▶ and services that run on that platform

✓ open source

✓ declarative

Declarative = define **WHAT** end result you want

Imperative = define exact steps - **HOW**



Difference Ansible and Terraform?

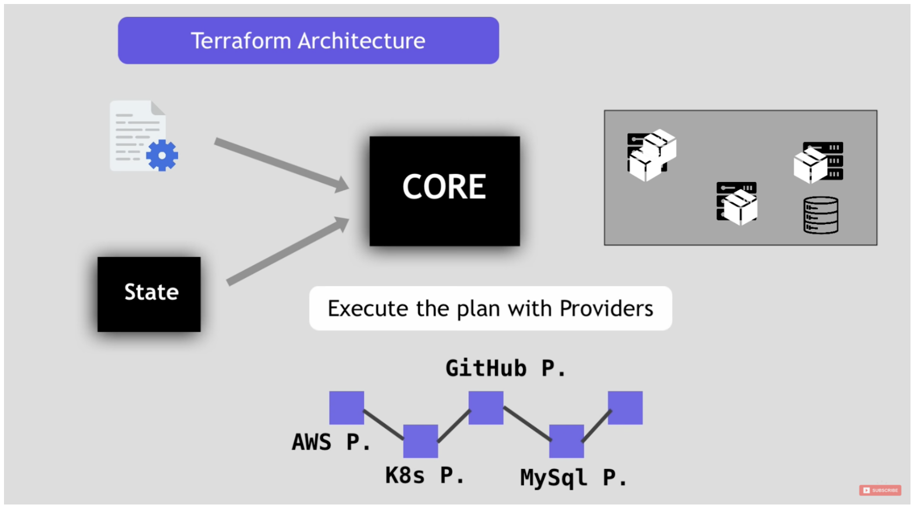
ANSIBLE
HashiCorp Terraform

Which one to use? 👀

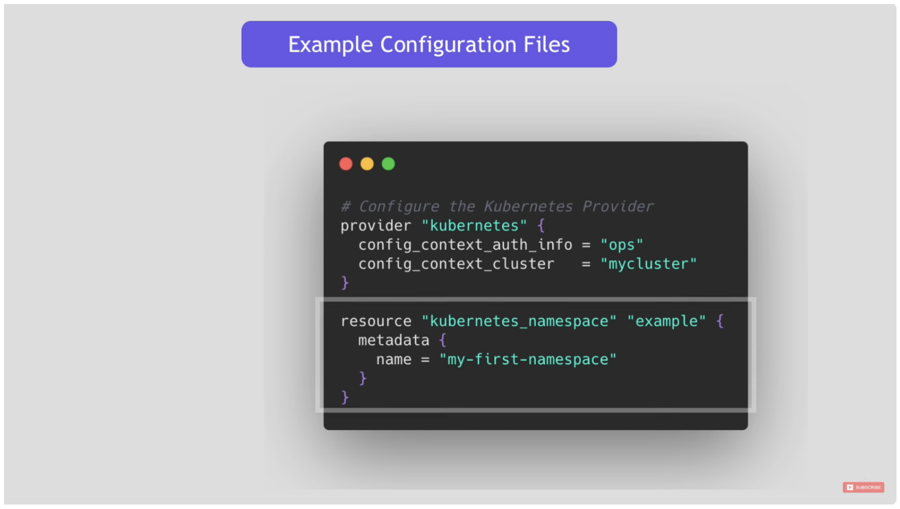
Both: Infrastructure as a Code

Ansible	Terraform
Mainly a configuration tool	Mainly infrastructure provisioning tool
more mature	relatively new
	more advanced in orchestration

Terraform Architecture



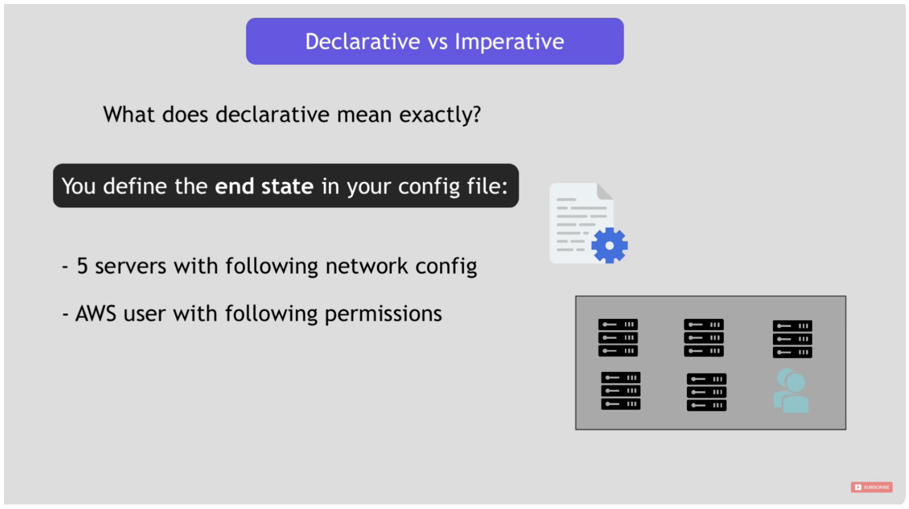
Example Configuration Files



```
# Configure the Kubernetes Provider
provider "kubernetes" {
  config_context_auth_info = "ops"
  config_context_cluster   = "mycluster"
}

resource "kubernetes_namespace" "example" {
  metadata {
    name = "my-first-namespace"
  }
}
```

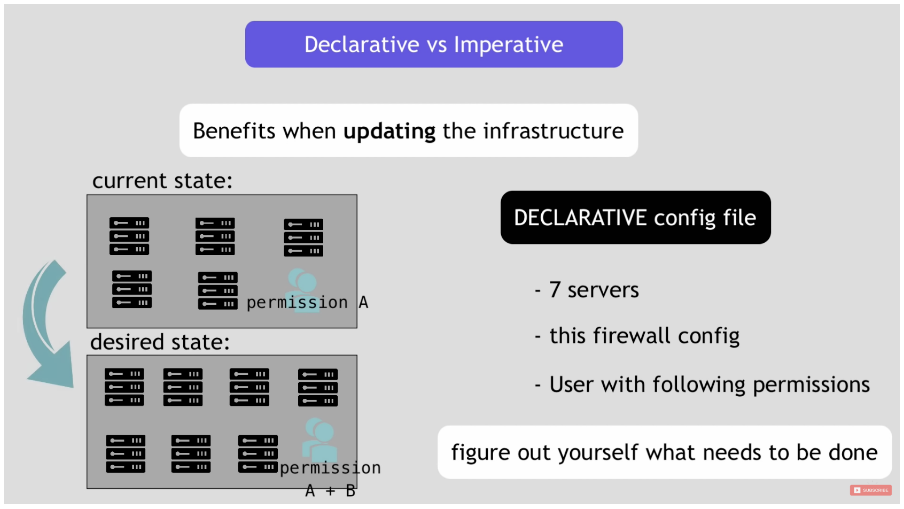
Declarative vs Imperative



What does declarative mean exactly?

You define the **end state** in your config file:

- 5 servers with following network config
- AWS user with following permissions



Declarative vs Imperative

Benefits when **updating** the infrastructure

current state:

permission A

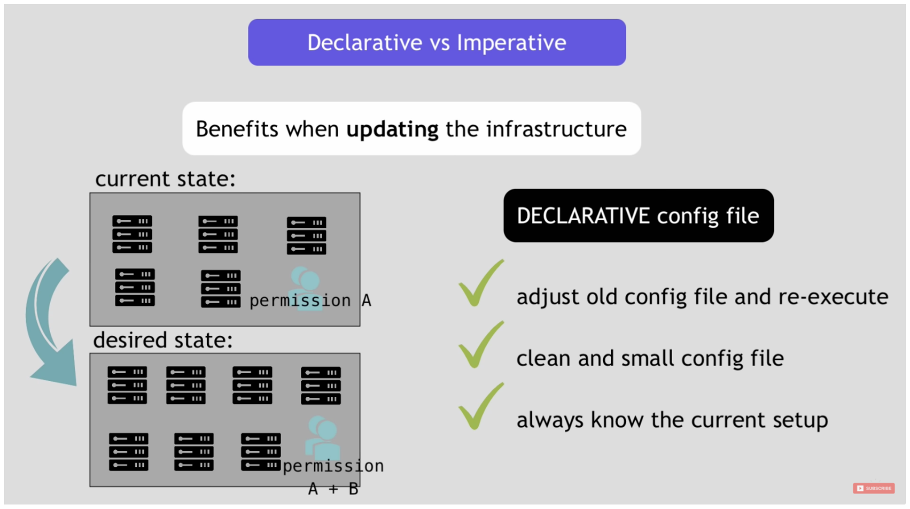
desired state:

permission A + B

DECLARATIVE config file

- 7 servers
- this firewall config
- User with following permissions

figure out yourself what needs to be done



Declarative vs Imperative

Benefits when **updating** the infrastructure

current state:

permission A

desired state:

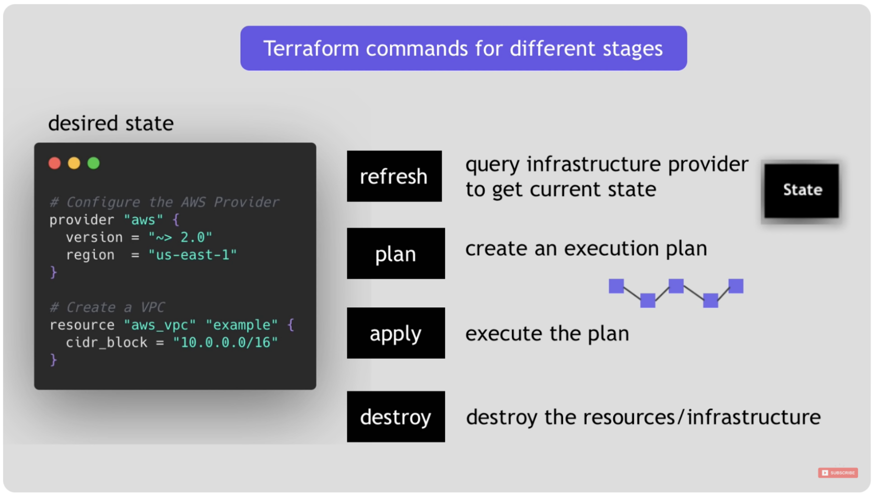
permission A + B

DECLARATIVE config file

✓ adjust old config file and re-execute

✓ clean and small config file

✓ always know the current setup



Terraform commands for different stages

desired state

```
# Configure the AWS Provider
provider "aws" {
  version = "~> 2.0"
  region  = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}
```

refresh	query infrastructure provider to get current state
plan	create an execution plan
apply	execute the plan
destroy	destroy the resources/infrastructure

Install Terraform Client

See https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

On Mac:

```
brew install terraform
```

Example

https://gist.github.com/ivaravko/501b4e8e5f1bc1bf5fecd149186b99b9

Sample: main.tf

mail.tf

```
# Configure Kubernetes provider and connect to the Kubernetes API server
provider "kubernetes" {
  config_path    = "~/.kube/config"
  config_context = "docker-desktop"
}

# Create an Nginx pod
resource "kubernetes_pod" "nginx" {
  metadata {
    name = "terraform-example"
    labels = {
      app = "nginx"
    }
  }

  spec {
    container {
      image = "nginx:1.23.2"
      name  = "example"
    }
  }
}

# Create an service
resource "kubernetes_service" "nginx" {
  metadata {
    name = "terraform-example"
  }
  spec {
    selector = {
      app = kubernetes_pod.nginx.metadata.0.labels.app
    }
    port {
      port        = 80
    }

    type = "NodePort"
  }

  depends_on = [
    kubernetes_pod.nginx
  ]
}
```

Initialize

```
terraform init
```

Run plan to check for errors

```
terraform plan
```

Apply changes

```
terraform apply
```

Destroy changes applied

```
terraform destroy
```

References

Reference	URL
Terraform Home	https://www.terraform.io
Terraform explained in 15 mins | Terraform Tutorial for Beginners	https://www.youtube.com/watch?v=l5k1ai_GBDE
Complete Terraform Course - From BEGINNER to PRO! (Learn Infrastructure as Code)	https://www.youtube.com/watch?v=7xngnjfIlK4