DevOps

On Thursday

- In the last session, we saw how CI/CD pipeline looks like.
- We talked about continuous integration and continuous delivery concepts.
- We tackle different CI tools like bamboo, Travis CI and Jenkins
- We saw the reason why we are using Jenkins as the CI server instead of Bamboo or Travis CI.
- We also defined the difference between a job and a build
- We saw that Jenkins can automate almost any job because it is friendly with MacOS, Windows and Linux
- We learnt about tools like Maven, Selenium, Git, and GitHub.
- We covered some of the disadvantages of using Jenkins as a CI server
- We also defined ways to solve the challenges we can face while using Jenkins as the CI server.
- We also learnt about different environments such as Development, QA, Production/operations

Before we go to terraform, let add the following to the DevOps Pipeline

- Docker
- Ansible
- Kubernetes

Let define them

Docker? -

Ansible – This is a service that is used to save time and increase the productivity of the DevOps team. It does this by automating the deployment of the complex, repetitive and tedious processes in the DevOps pipeline. Ansible is an automation tool. It is highly used in deployment and configuration processes. It is very important because it helps to organize, run complex, repetitive and tedious tasks in the DevOps pipeline. For instance, rolling updates for continuous deployment to happen, you can use ansible. You can also use ansible in the continuous delivery process after the code has been released from the QA team and ansible can automate some complex, tedious, and repetitive process involved in continuous delivery process.

When you use Ansible to configure the tools in the DevOps pipeline, difficult manual tasks become repeatable and less vulnerable to error.

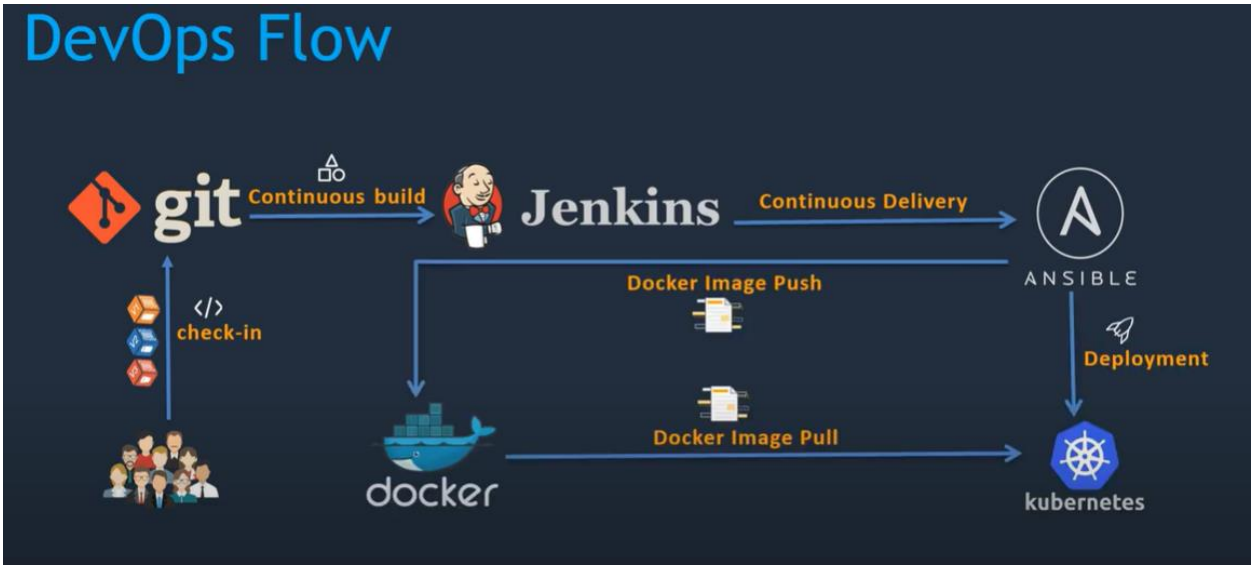
Other areas you can use ansible include storing the artifacts to the antifactory, deploying images to hub.docker.com or pushing them to docker

Kubernetes - On the other hand, Kubernetes is a system designed to coordinate Docker containers. Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

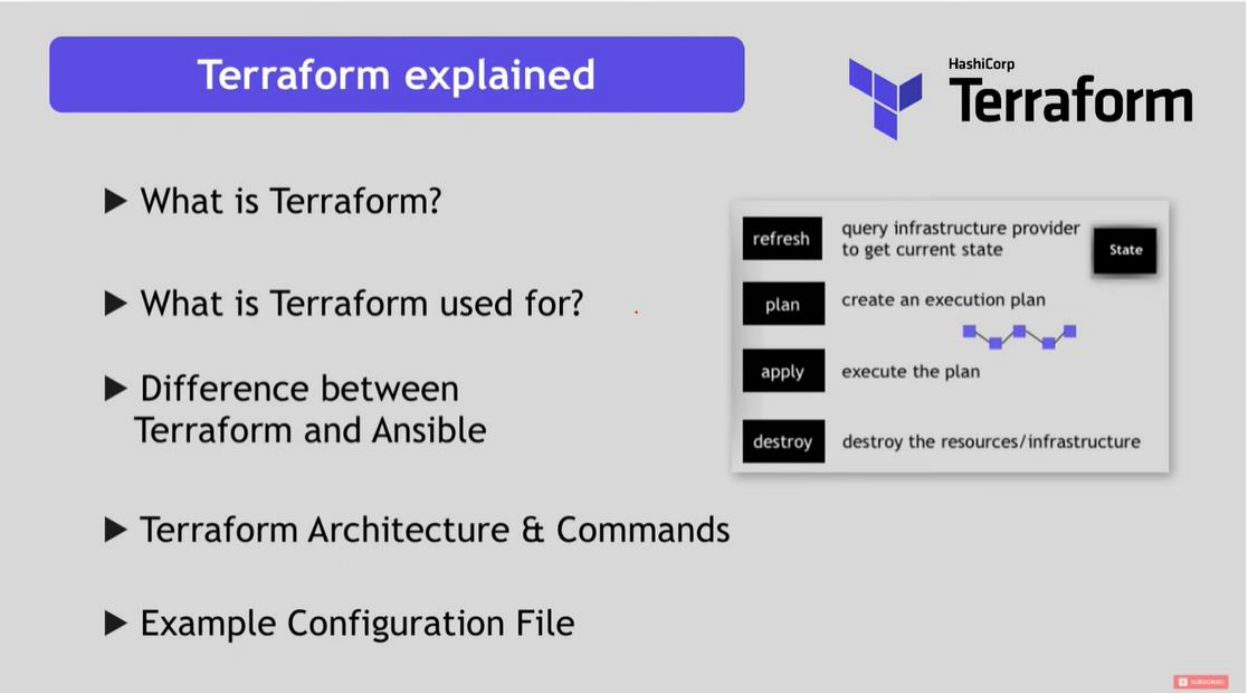
What type of scaling is available in Kubernetes?

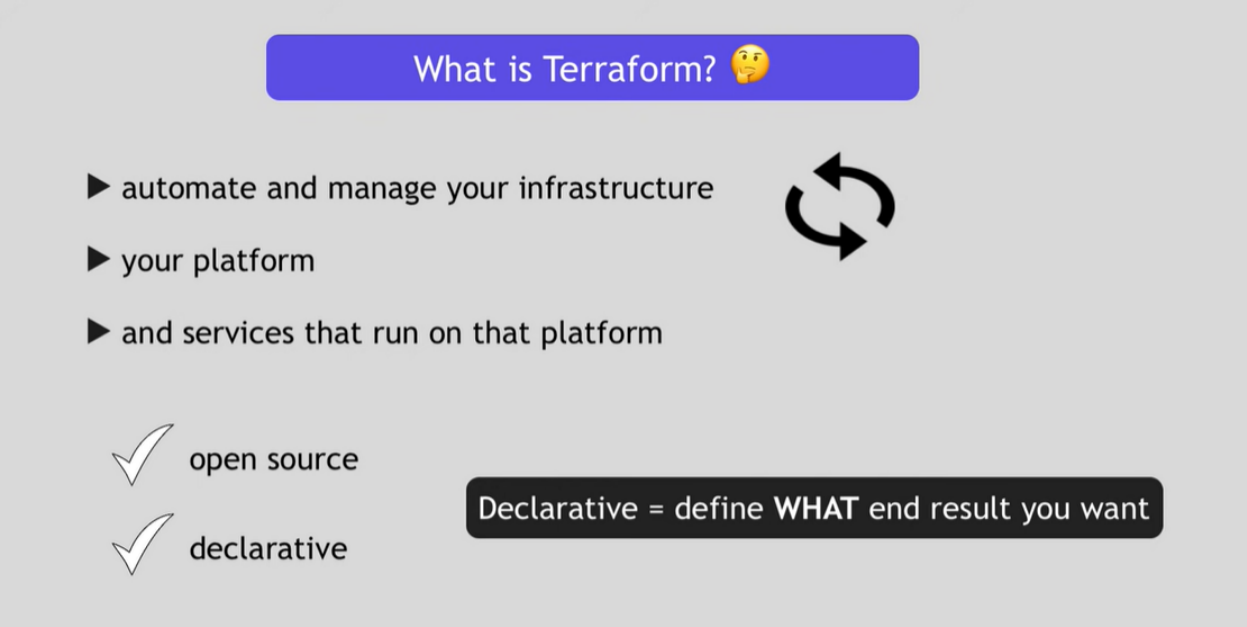
Do not forget that K8s can also be used in managing other containers and not just docker.

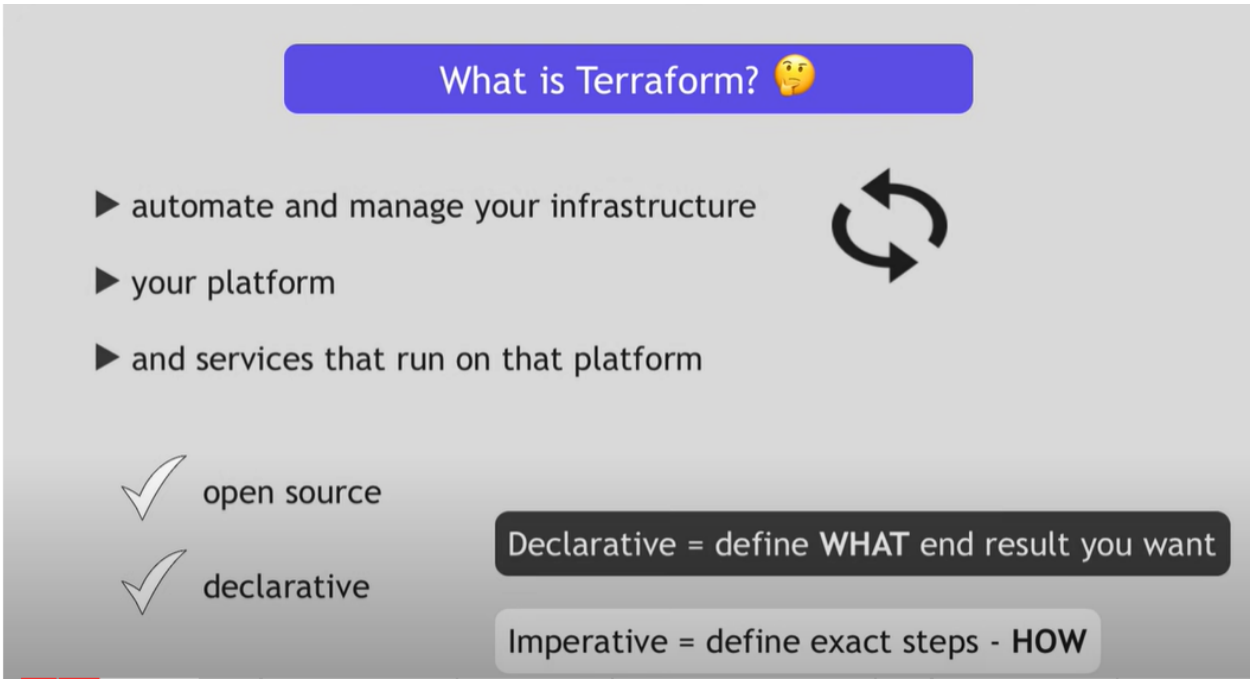
Kubernetes clusters allow containers to run across multiple machines and environments: virtual, physical, cloud-based, and on-premises. Kubernetes containers are not restricted to a specific operating system, unlike virtual machines. Instead, they can share operating systems and run anywhere.



Terraforms


Terraform explained
HashiCorp
Terraform
▶ What is Terraform?
▶ What is Terraform used for?
▶ Difference between Terraform and Ansible
▶ Terraform Architecture & Commands
▶ Example Configuration File
refresh
query infrastructure provider to get current state
State
plan
create an execution plan
apply
execute the plan
destroy
destroy the resources/infrastructure


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



When using terraform, since it is a declarative tool, you do not need to define the steps that will allow you to get the result. You just need to define What end results you want and terraform will figure out the steps to take to give you the results.

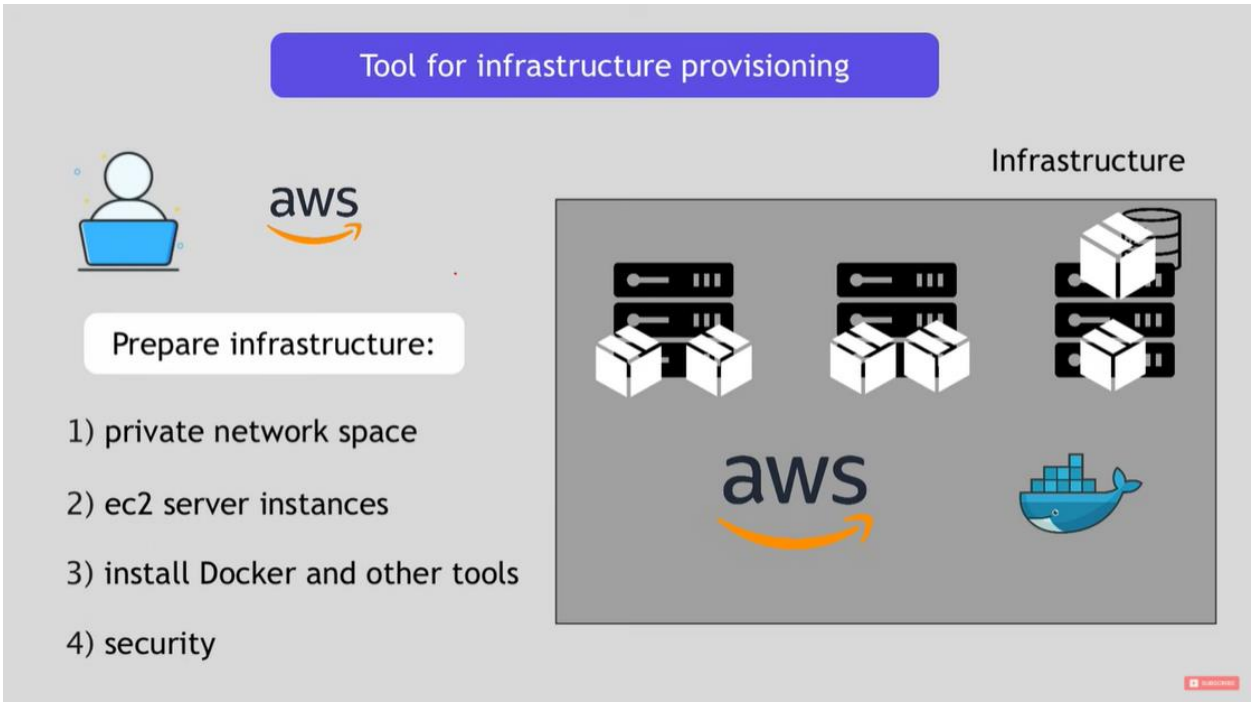
Imperative is the opposite. You must define the steps that will help you to get what end results you want.

Let look at what is infrastructure provisioning

In this scenario, we want to deploy docker application.

The first step is to prepare the infrastructure we are going to deploy.

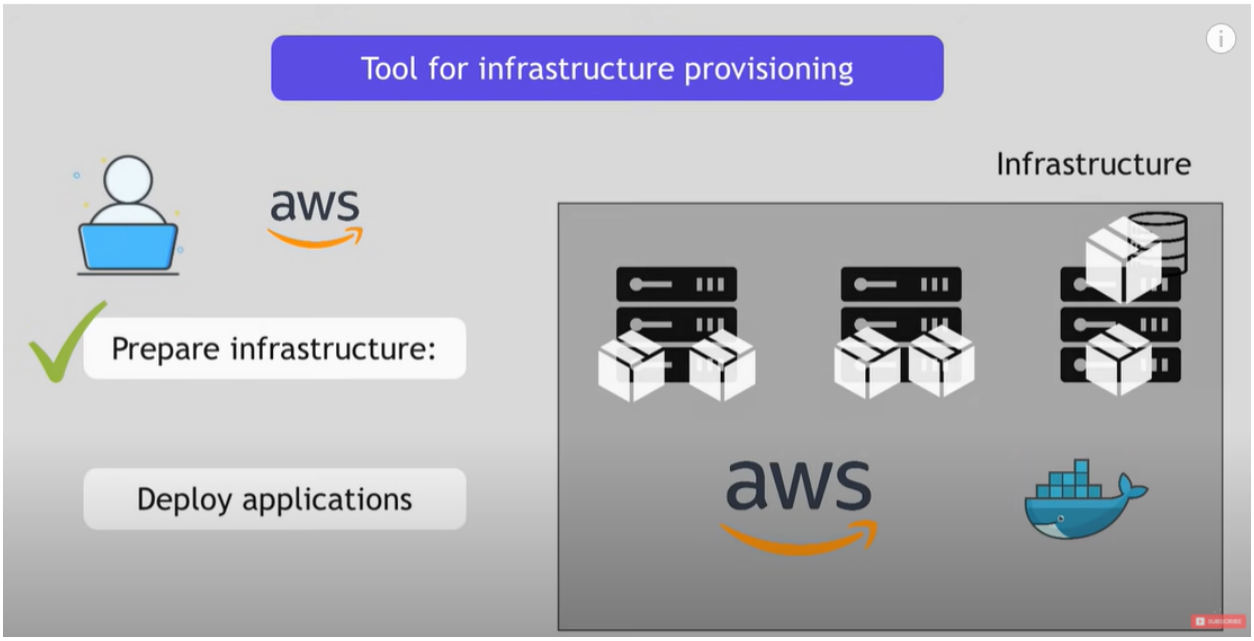
Remember how we create AWS diagram. That was infrastructure provisioning because we never deployed the diagram.



When the infrastructure is ready, we then deploy the code.

There are two things in one

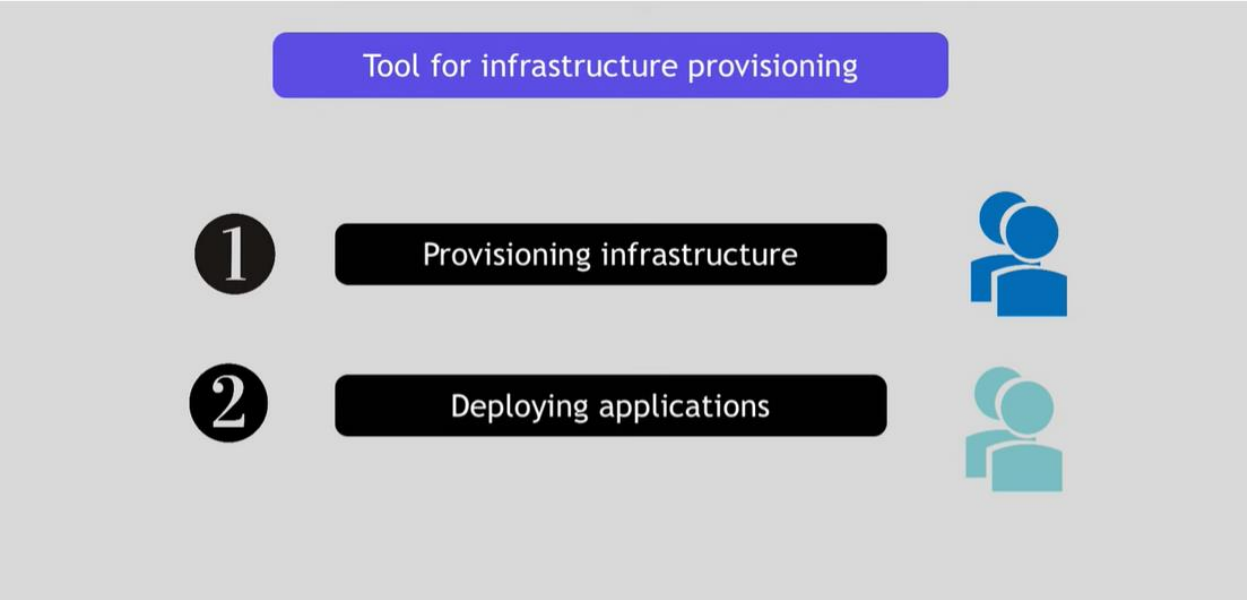
Preparing the infrastructure (provisioning) and d deploying the ready infrastructure



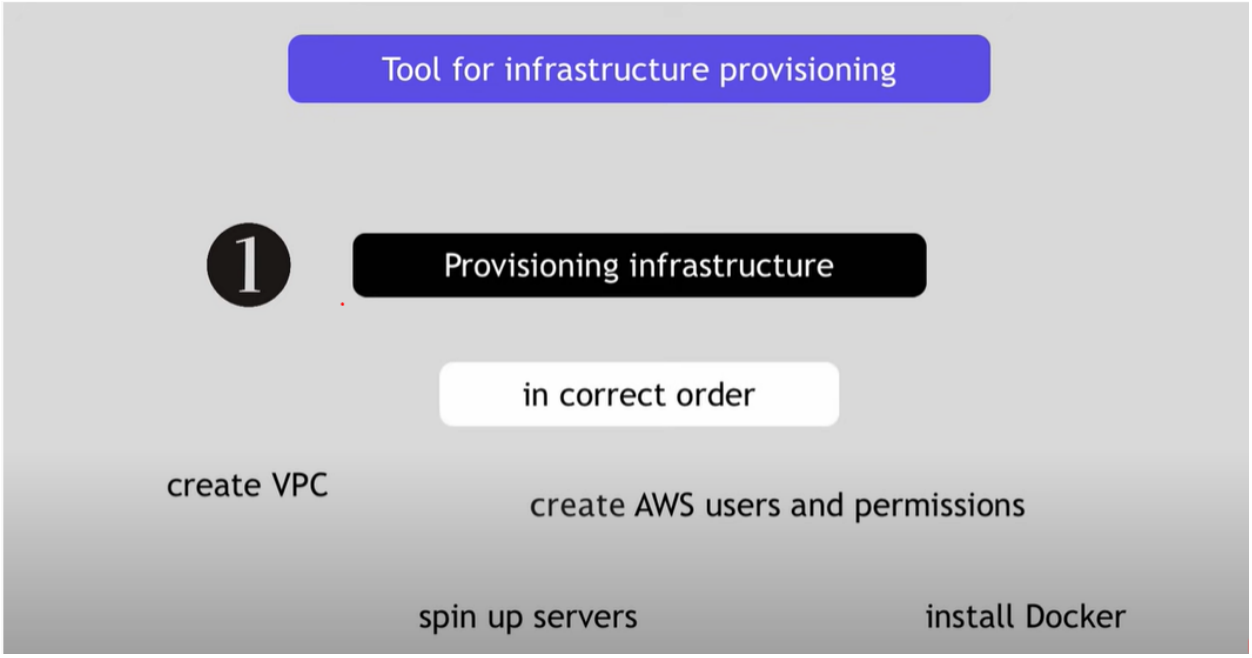
You can have two teams doing the two tasks.

Provisioning team – DevOps

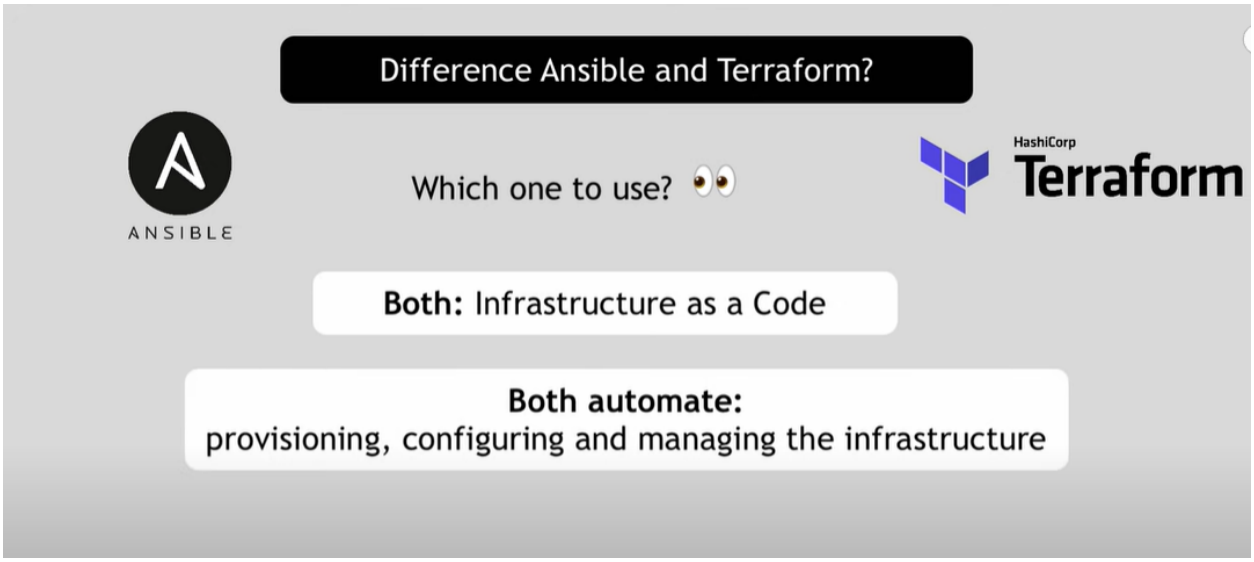
Deploying – Operational/production team



You must build the architecture in order because that how terraform will execute the declarative commands. Remember the document we had in AWS. We defined all the steps in provisioning an AWS infrastructure.

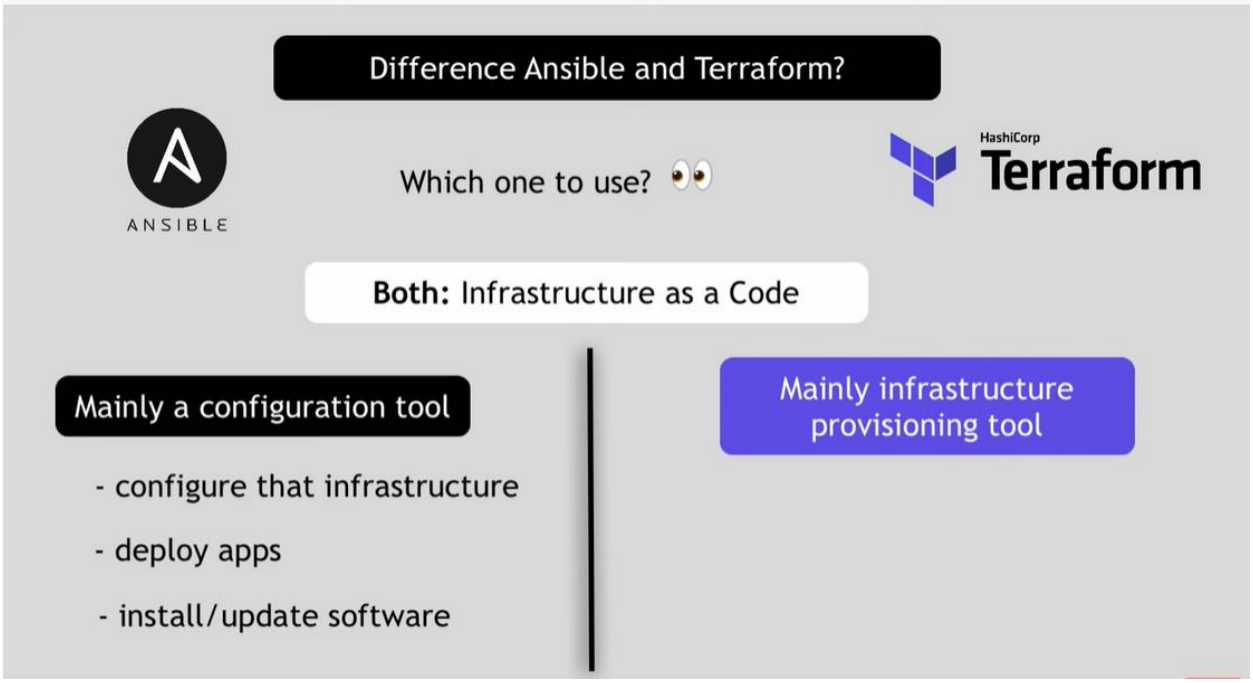


Terraform comes in handy in infrastructure provisioning



Both are used for provisioning and automation

BUT

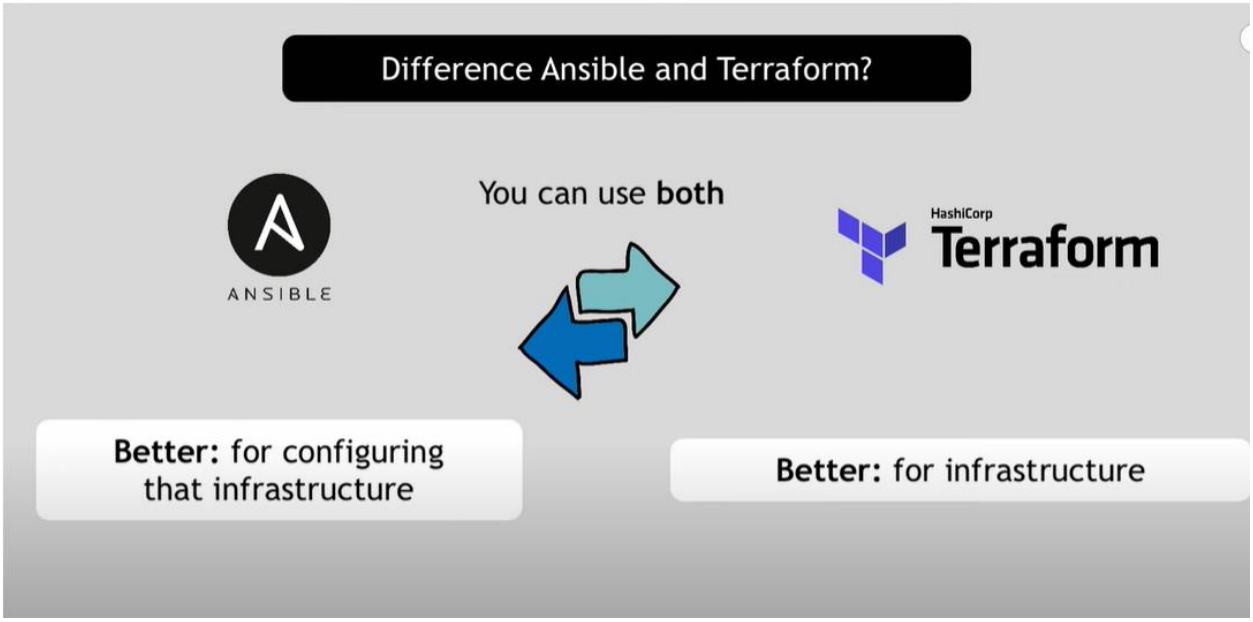


Ansible is more mature

Let make a final statement

Ansible is a better tool for configuring and deploying the readymade infrastructures

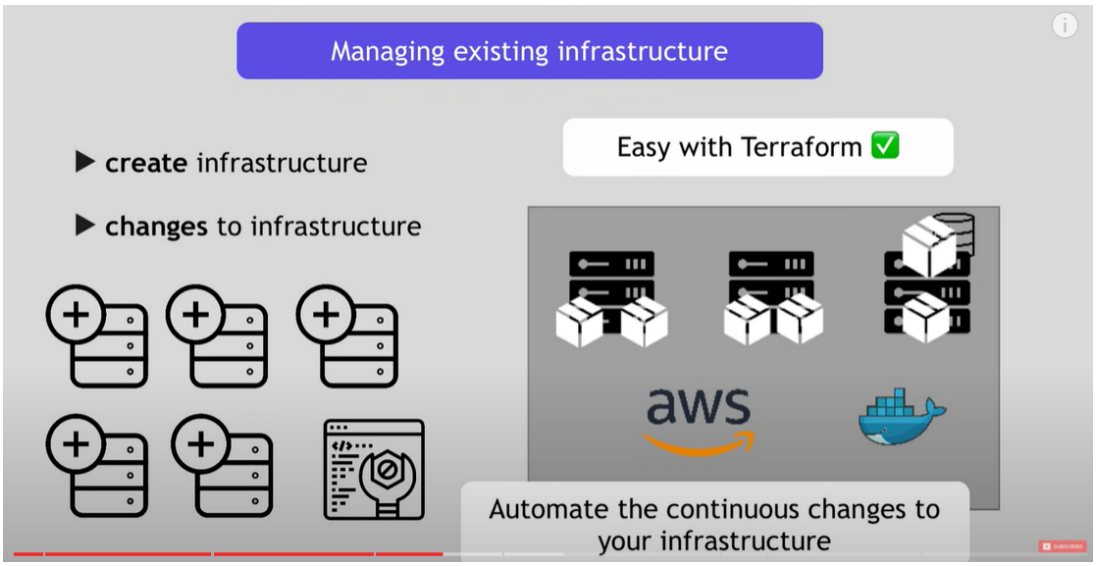
Terraform is a better tool in the provisioning of the infrastructure – The process of preparing and ensuring the infrastructure is ready.



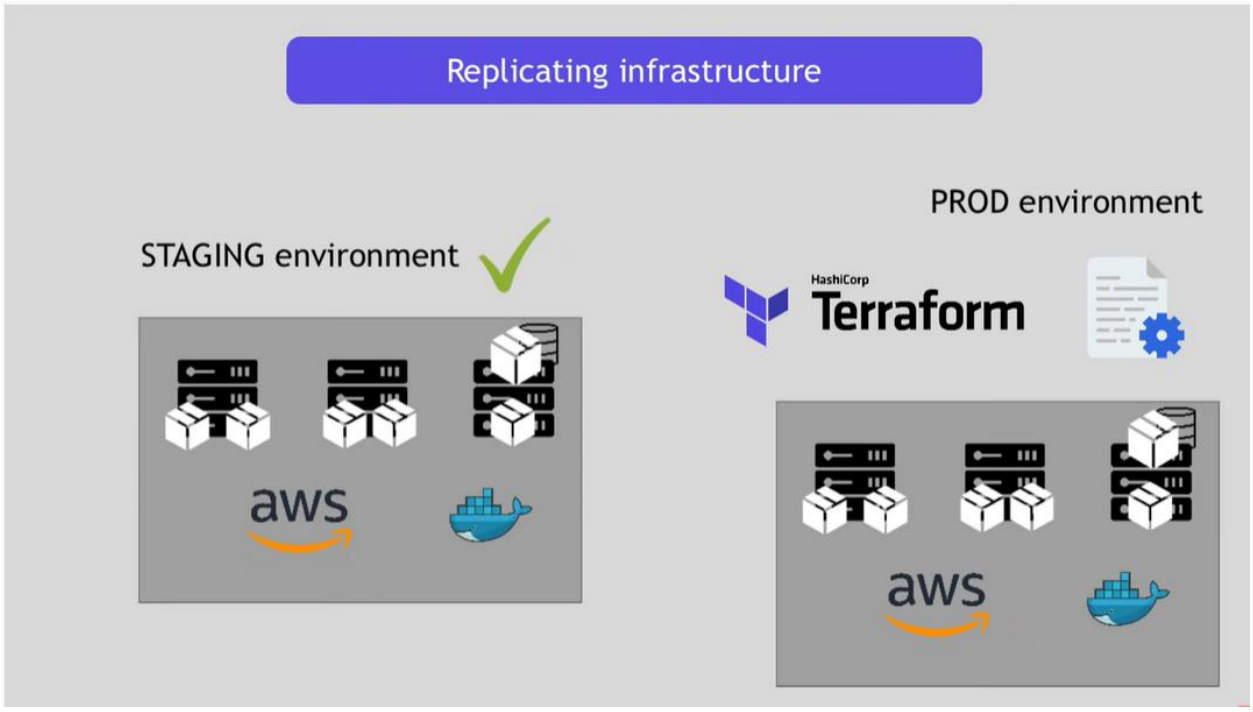
DevOps engineers use both tools to cover the application/software building end to end and I believe you now know why.

Different Infrastructure as Code Tools

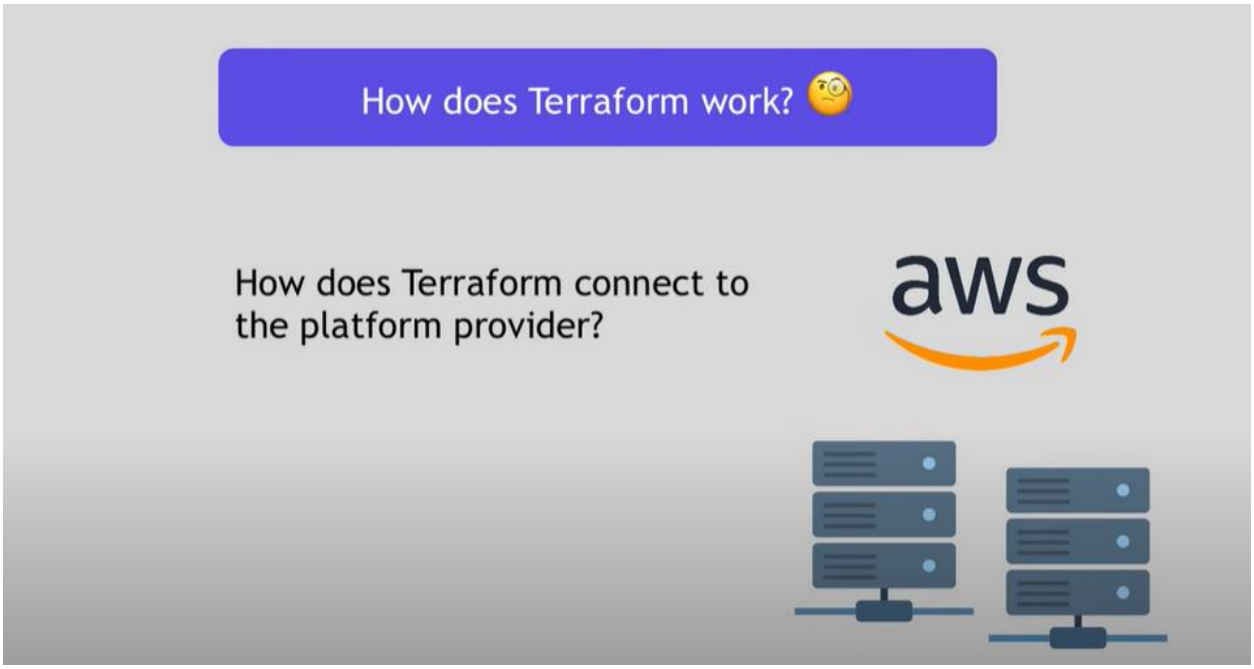
Managing the existing infrastructure is easy with terraform. Just declaring what the end results of the changes should look like and terraform will figure out the steps to give you the end result.



Creating an identical availability zone becomes easy because all you are doing is take the replica of the code and deploy it.



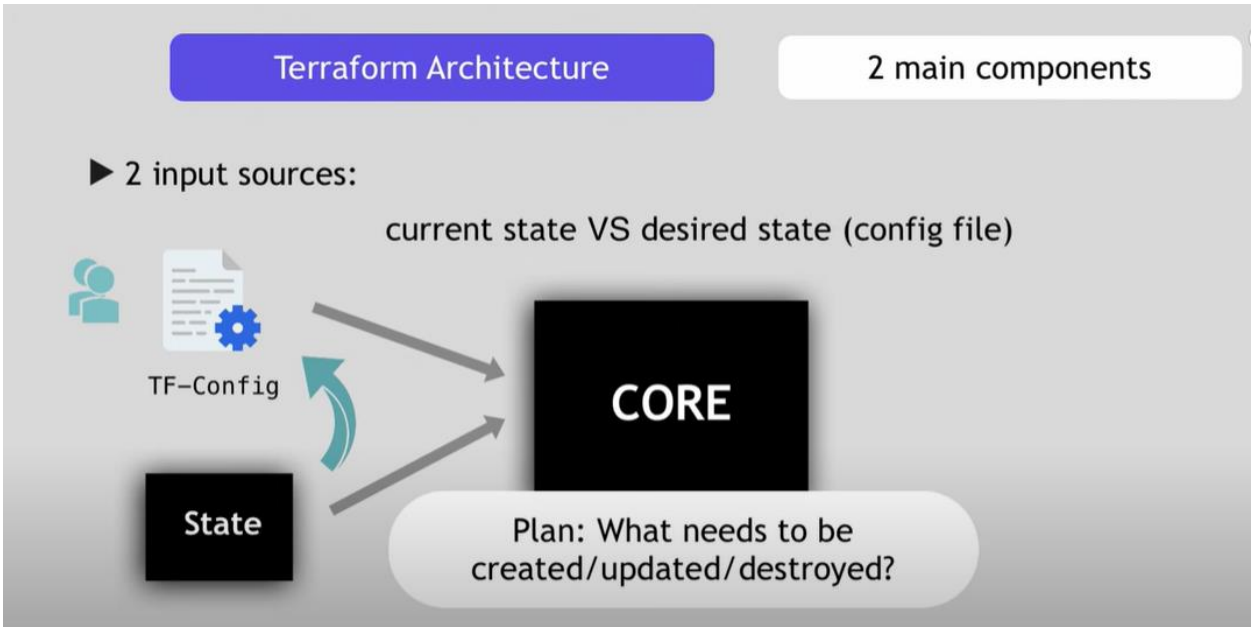
Remember how we created the availability zones. You can replicate with terraform



How does terraform connect with AWS to create instances

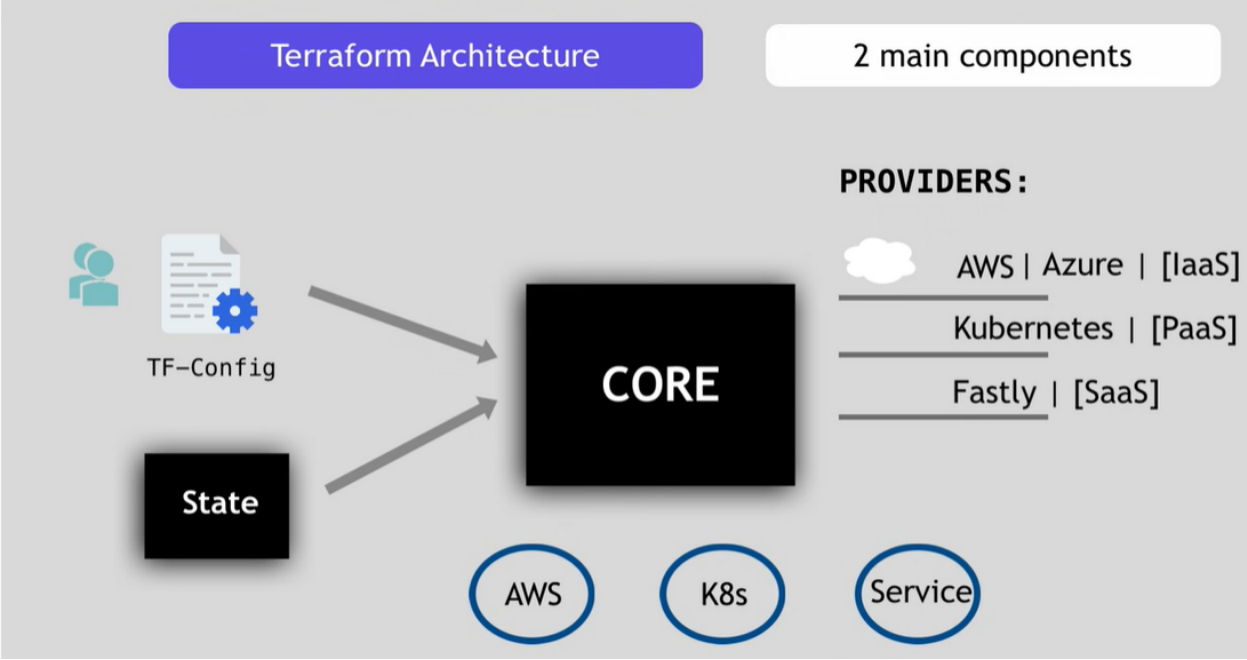
It has two key components to its architecture

1. Terraform core - this has two input sources
   The TF-config file you write the declarative commands here. State which keeps the memory of the current status of the setup/infrastructure you want to create.



The terraform compares the command and the current state and decides which needs to be created, deleted, and updated so that it can get to what end results you want.

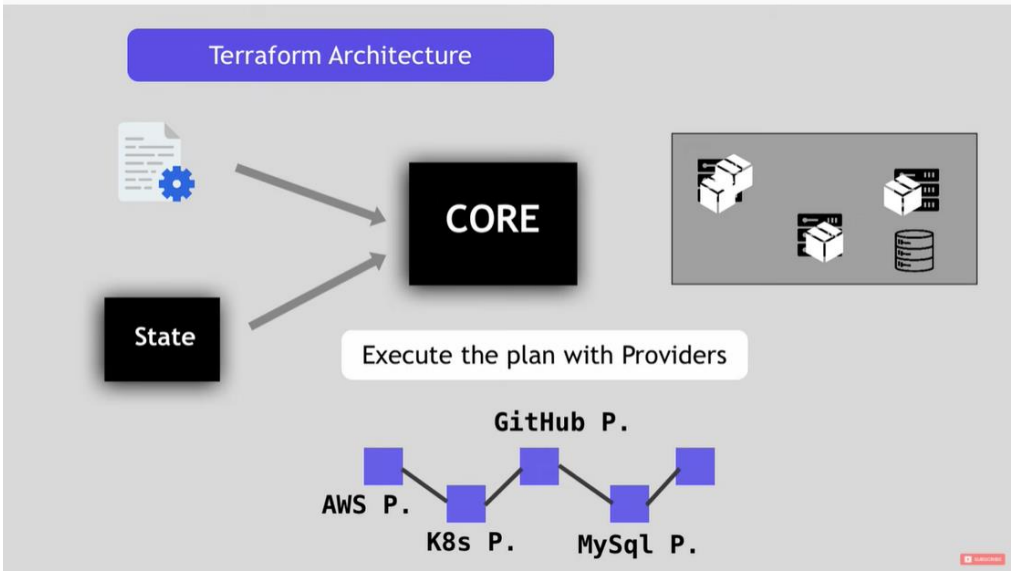
The second component is the PROVIDERS

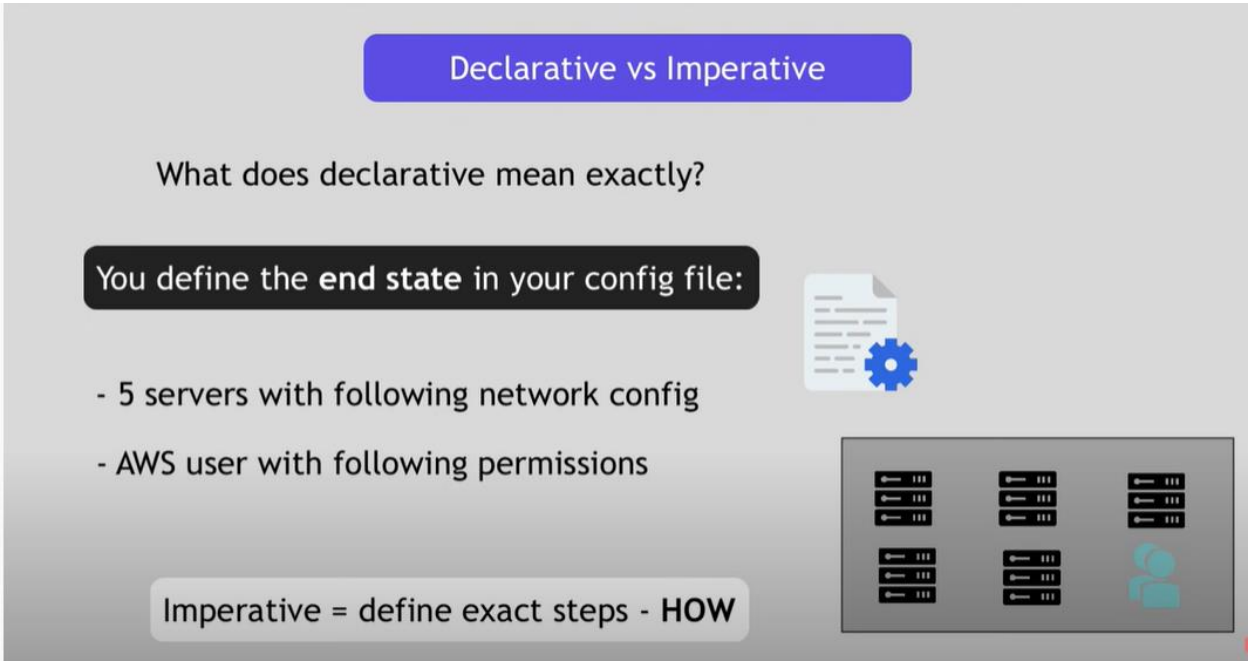


It has over 100 provides who give terraform access to the resources like in AWS and that is how terraform is able to help you provision the infrastructure and set it up ready for deployment with ansible. Literally, terraform is making your life very easy by building your end results and figuring out all the steps to take to give you that product/result that you want.

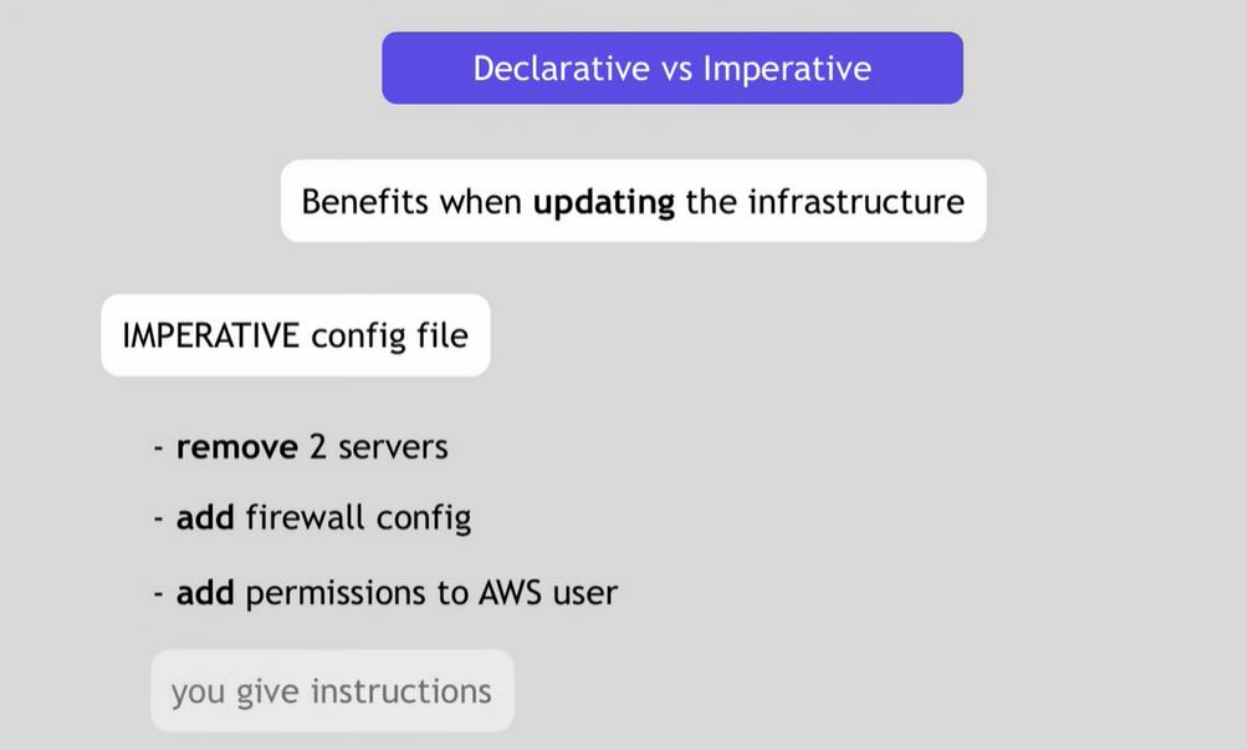
However, if you want to deploy and configure the provisioned infrastructure, the best tool is the Ansible or any other configuration tool out there.

To give you the what end results you requested, the terraform check the tf-config file and compare with the status to see the updates and the changes needed, then it connects with the service provider you requested and execute the plan you gave it.

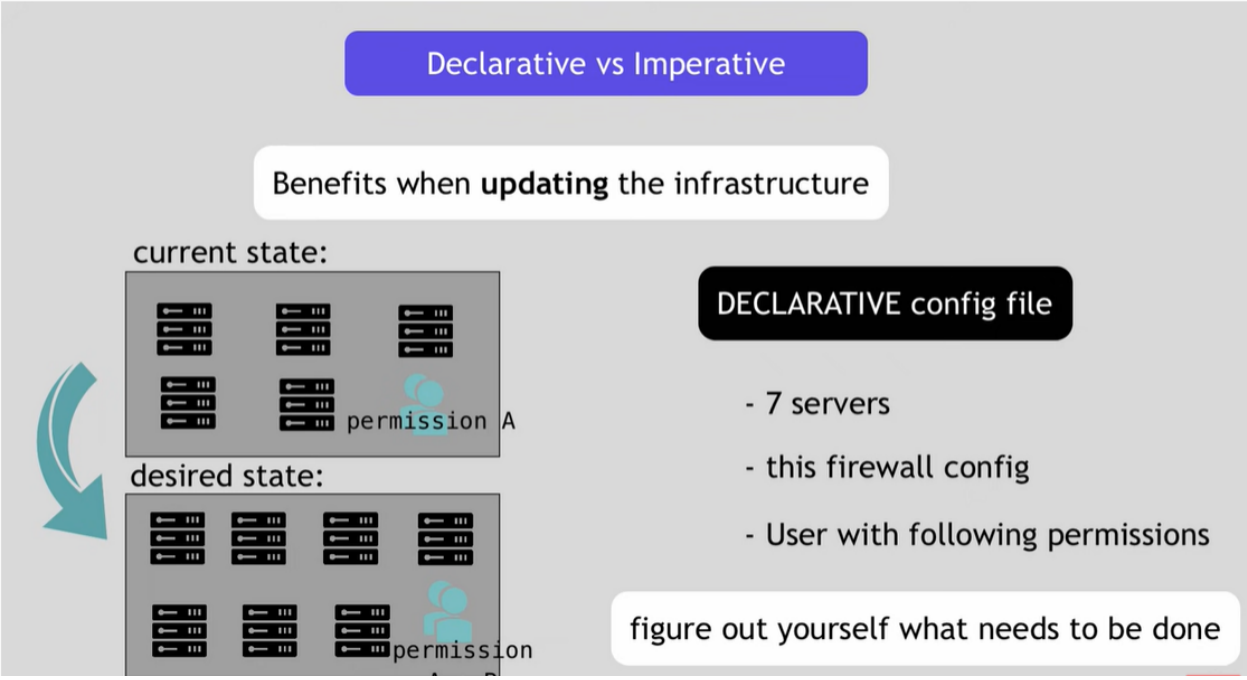

Terraform Architecture
CORE
State
Execute the plan with Providers
GitHub P.
AWS P.
K8s P.
MySql P.


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
Imperative = define exact steps - HOW

Imperative

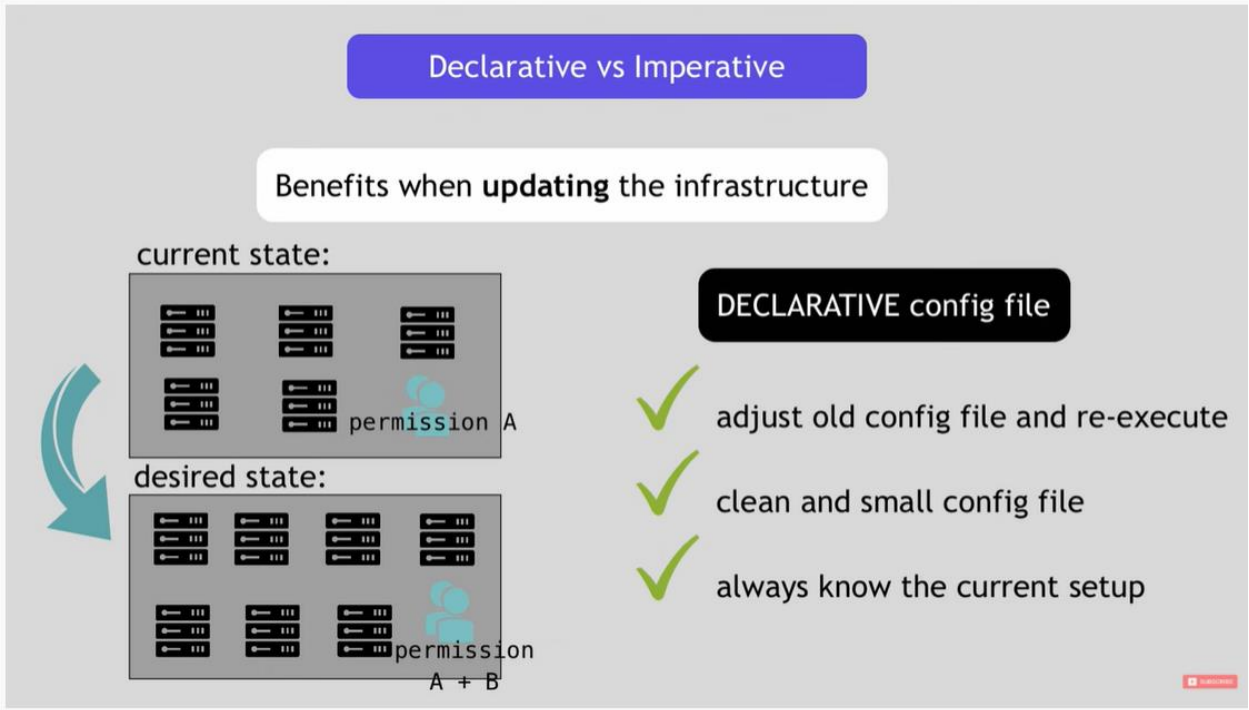


Declarative

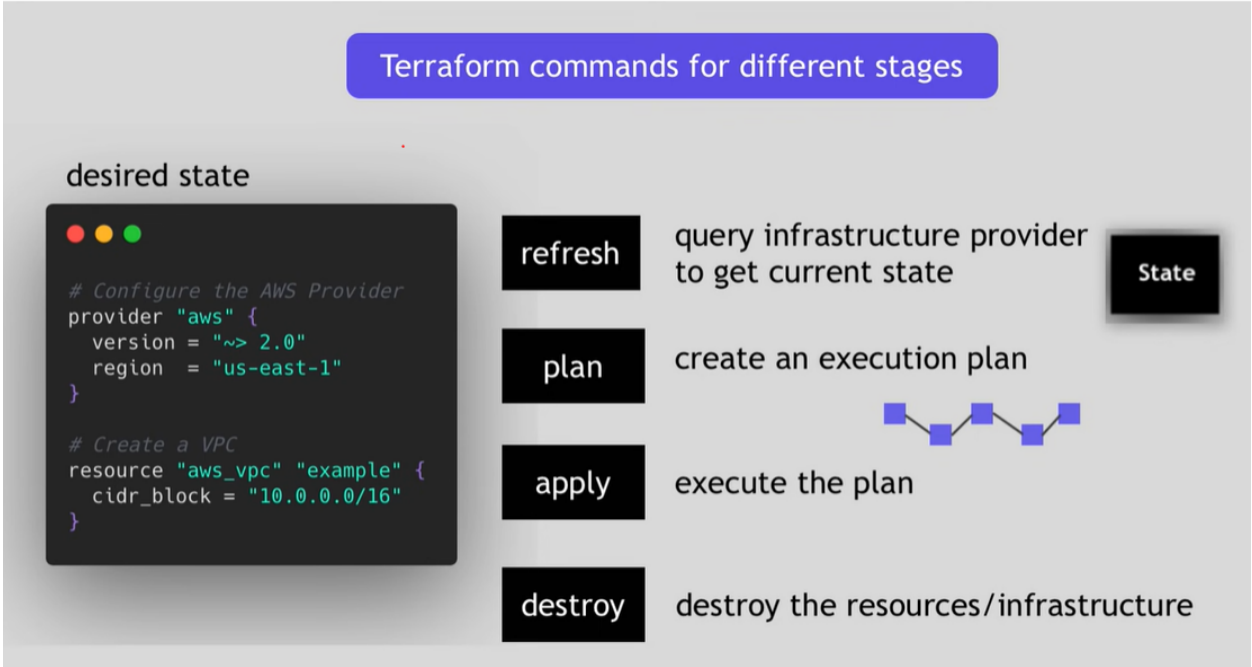


You just declare the end results that you want and terraform will figure out how to satisfy your end results.

In imperative you must define all the steps of setting the 7 servers. You have to answer the question how I set the 7 servers, and which will be steps will be required.



In declarative terraforms always know the end results. All it needs to do is adjust the old code or tf-config file to the current codes that you just stated



#REFRESH is to refresh so that you can be in the current state in the setup of the provide you want to use.

#PLAN – terraform compares the current tf-config file with the current state of the infrastructure services you want to us, makes the necessary changes in the form of updating or deleting and then come up with a plan to execute the desired end results.

#Apply – it is now going to execute the plan that we created.

This means that if you just use the #APPLY command, terraform is going to REFRESH, PLAN AND EXECUTE THE PLAN and you will get the desired results.

DESTROY – this command will remove all the declared services one by one until everything is cleared.