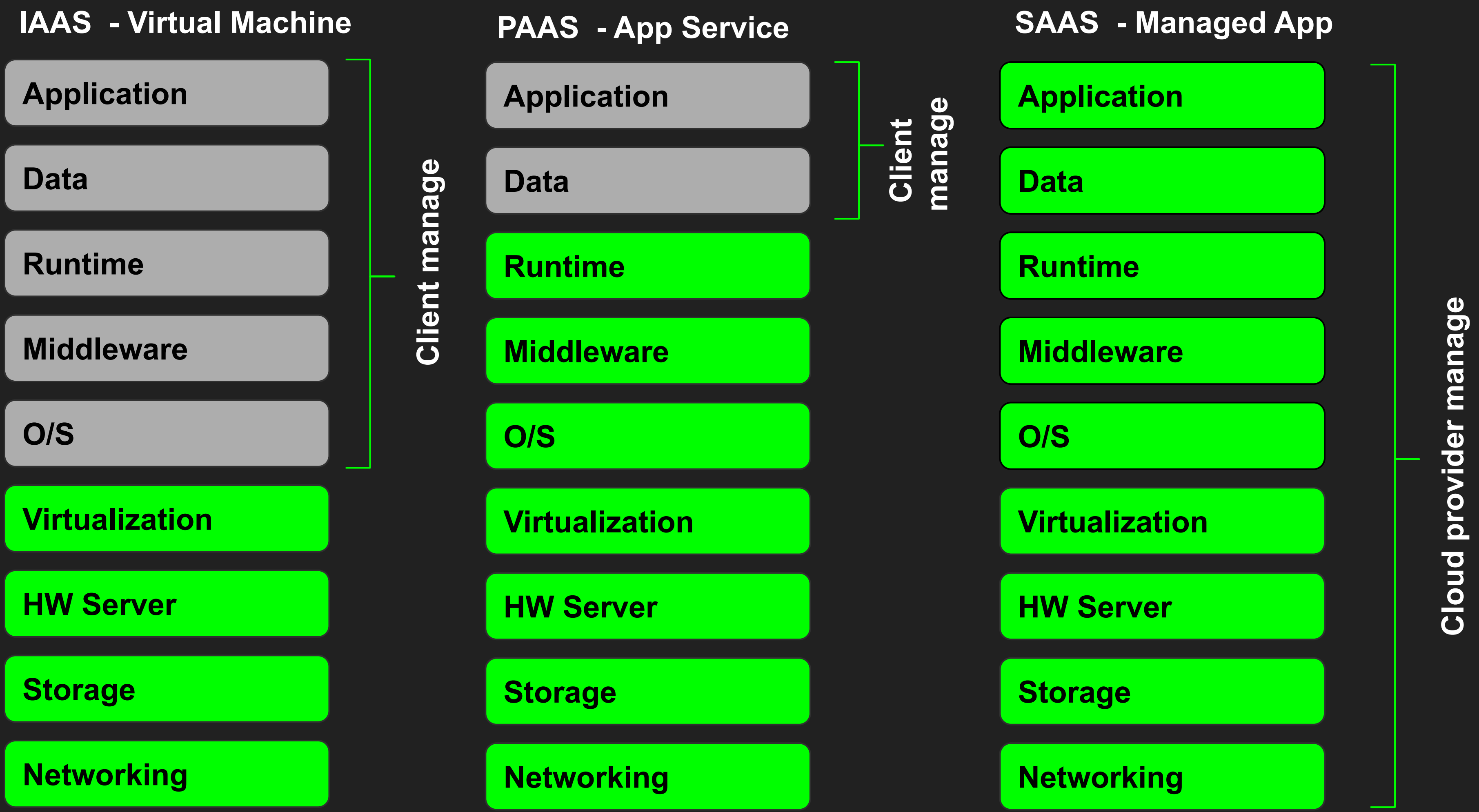
Section 1

Introduction to DevOps,IaC and Terraform

What’s the deal with devops, the cloud and using code for everything..

What’s the deal with devops, the cloud and using code for everything..

- The rapid migration to cloud is transforming how applications and infrastructure are being deployed and managed.
- This gave the rise to DevOps to manage both Dev and Ops in code and deliver software efficiently and quickly.


IAAS - Virtual Machine
Application
Data
Runtime
Middleware
O/S
Virtualization
HW Server
Storage
Networking
Client manage
PAAS - App Service
Application
Data
Runtime
Middleware
O/S
Virtualization
HW Server
Storage
Networking
Client manage
SAAS - Managed App
Application
Data
Runtime
Middleware
O/S
Virtualization
HW Server
Storage
Networking
Cloud provider manage

The cloud offers infrastructure elements as software services and is API driven

API or application programming interface is the language spoken between two applications

The management of the infrastructure is now using software rather than physically managing racks, cables and hardware..

- DevOps is embodied with the CALMS model which defines DevOps more as a culture shift than just a technical one which enable the automation of the software delivery process from code to production.

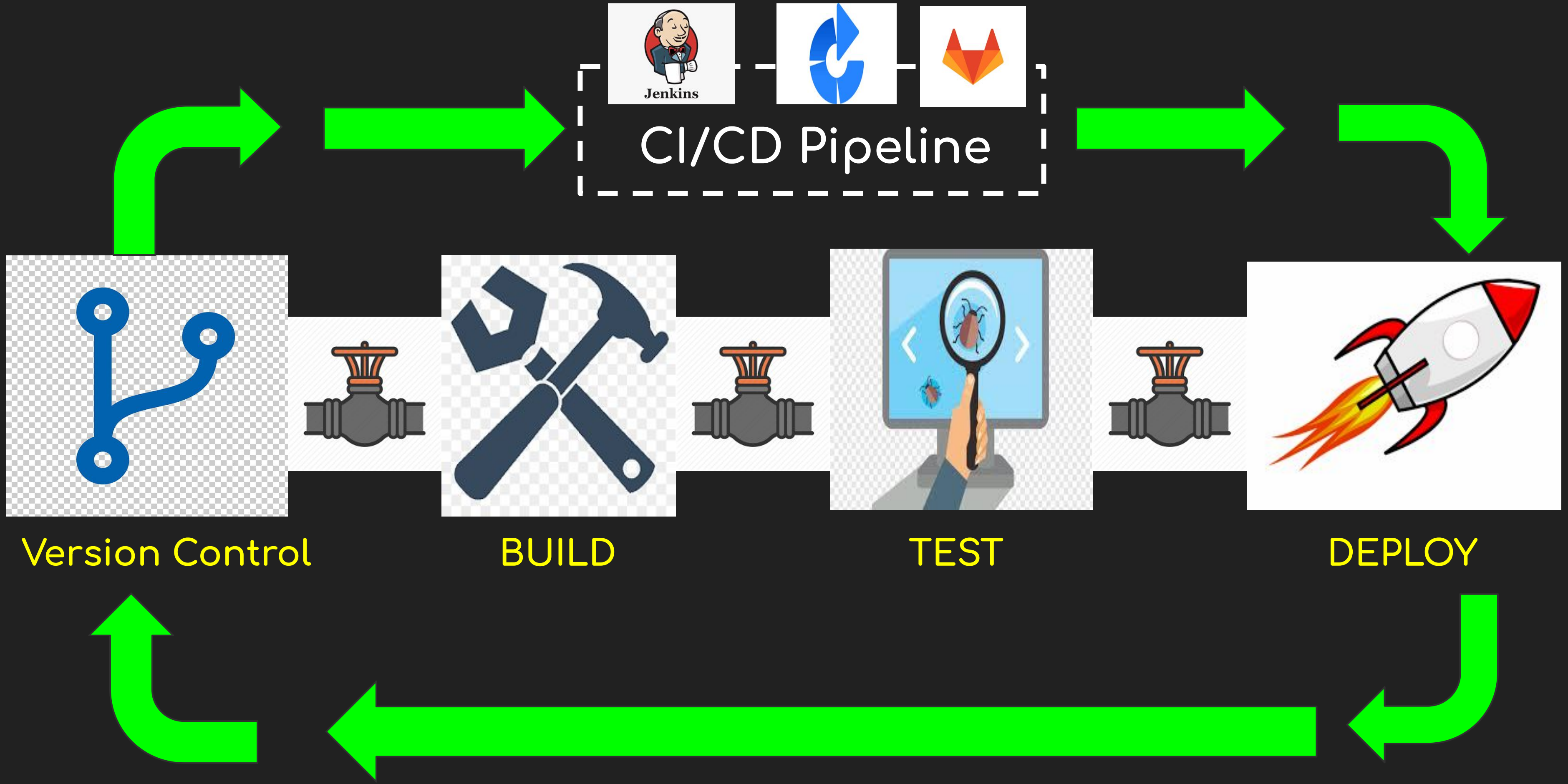
collaboration

automation

lean

measurement

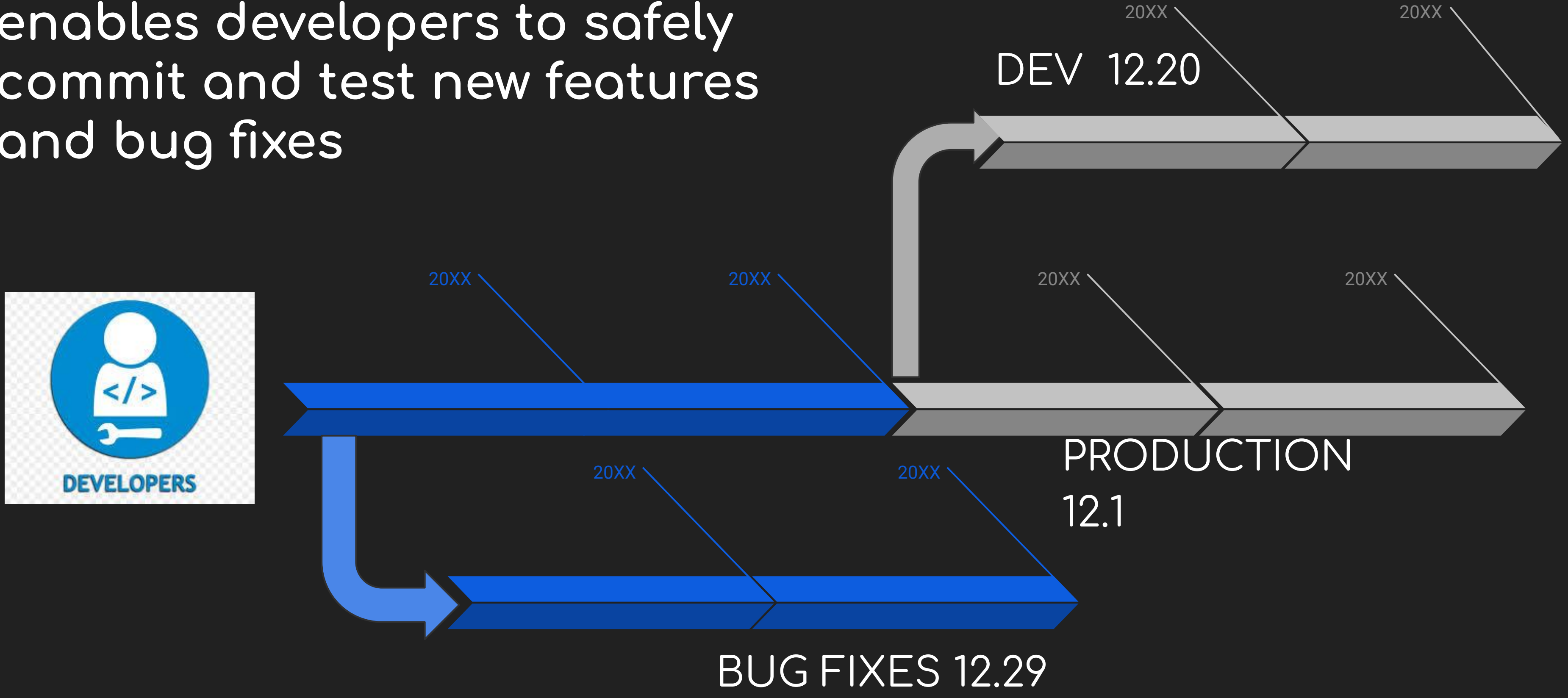
sharing


Jenkins
CI/CD Pipeline
Version Control
BUILD
TEST
DEPLOY

Version Control Systems or Source Code Management systems enables users to commit and share their code and version using branches.

These servers are also known as repositories such as GitHub, Git, BitBucket, AWS codecommit or Azure devops server.

Branches represent a certain version of the code and enables developers to safely commit and test new features and bug fixes



Infrastructure as Code and why use it..

Before Infrastructure as Code:

- manual
- ticket driven
- resources vertically scaled
- siloed infrastructure teams
- prone to mistakes and potential outages
- anxiety and low self esteem

Infrastructure as code or IaC enables users to create, provision, update and manage every aspect of their infrastructure using high level code.

But wait..isn't IaC the same as using Ad Hoc scripts..? Nothing new right??

Ad Hoc scripts tend to break.

Infrastructure as Code comes in many flavors:

- Infrastructure provisioning tools:

  Terraform, CloudFormation, AzureResourceManager, Heat

- Configuration management tools:

  Ansible, Chef, Puppet...

Benefits to actually make the jump to the coding side..

- Self service...no more dreaded IT tickets.
- You can now see and read your infrastructure
- It can be versioned like code using VCS and automated at the click of a button..
- Its safe....it can be validated and secured..It can be used for disaster recovery
- It will bring confidence!

Introduction to Terraform

Terraform is an Infrastructure provisioning tool ...not configuration management or application deployment!

Its Open Source

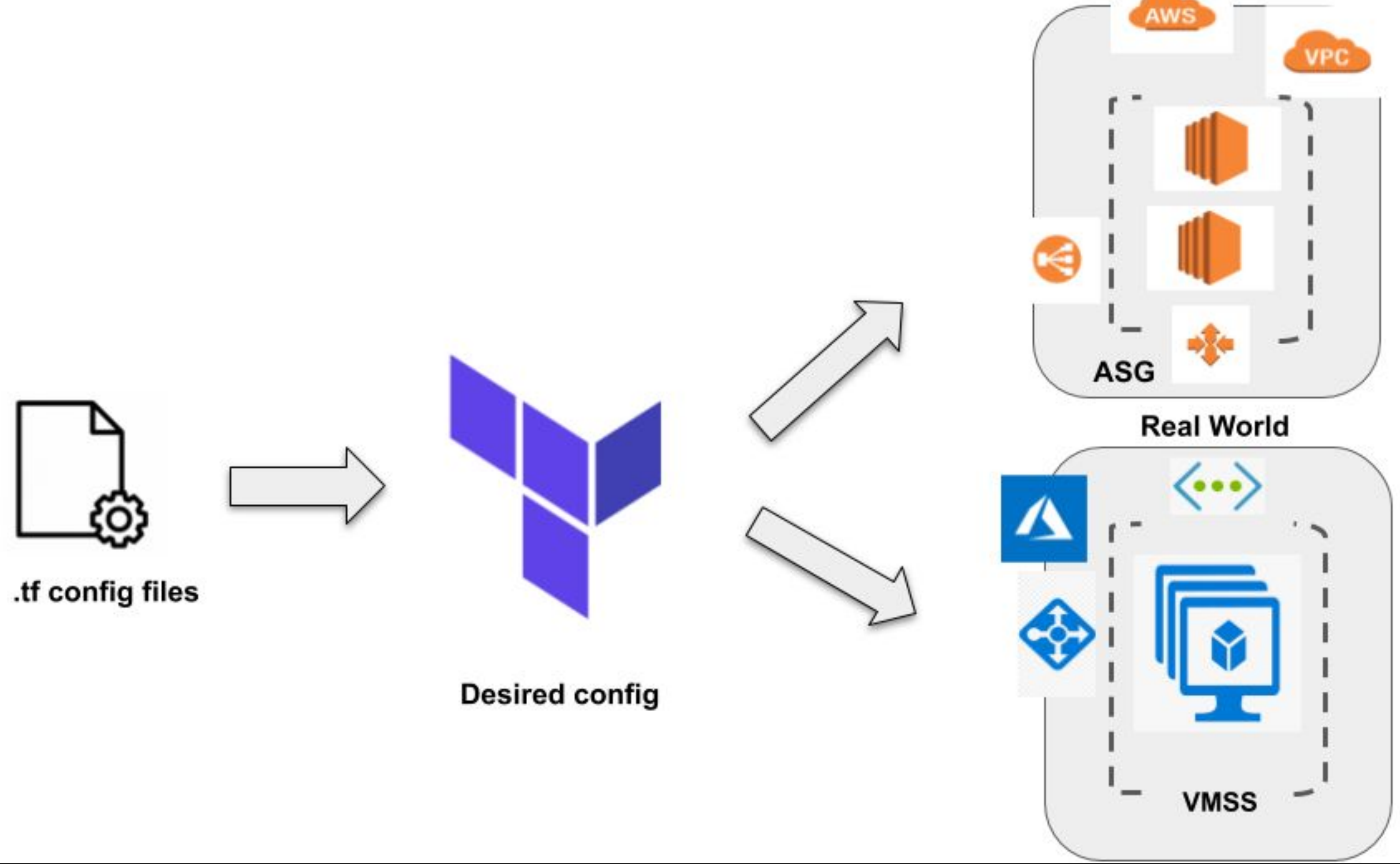
It is human readable and machine editable..no need to be a programmer.

Uses the Hashicorp Configuration Language or HCL but can be expressed in JSON as well.

Cool factor..

Platform agnostic..provision in any infrastructure provider..no dependency on the Terraform software

> One code syntax, One workflow ...any infrastructure


.tf config files
Desired config
AWS
VPC
ASG
Real World
VMSS

One Terraform workflow or execution plan is used to manage the execution of these configuration files in a provider.

The reusable and repeatable nature of the Terraform workflow will give the operator or admin confidence in managing the infrastructure

Use cases can be to provision IaaS services such as VMs, Virtual networks, security groups, firewalls, containers or Kubernetes clusters..

Hub and Spoke security architecture

Multi-cloud

app hosting service like Heroku

Disaster recovery sites or Test environments like canary or blue-green

Benefits of using Terraform

Terraform is declarative.

Declarative only defines the desired end state of a resource or service rather than having to define all the steps to achieve that desired state.

No need of domain knowledge

Terraform is immutable

Reduce configuration drift
Easier to test and qualify for deployment
Easier to upgrade or roll back to a known working version without downtime.
No need to keep tracking changes made to a build with complex deployments.

Terraform is idempotent

- Ensure that desired end state is always what is provisioned in the real world
- Compares the real world with the desired end state to decide what to add, delete or update without having to reprovision the whole infrastructure
- Any manual changes will be destroyed.

Terraform is idempotent, declarative and immutable

Terraform is a reliable, reusable and predictable Infrastructure as Code tool.