Terraform

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

1

What is Terraform?

What is Terraform?

- **Terraform** is the infrastructure as code offering from HashiCorp.
- It is a tool for building, changing, and managing infrastructure in a safe, repeatable way.

2

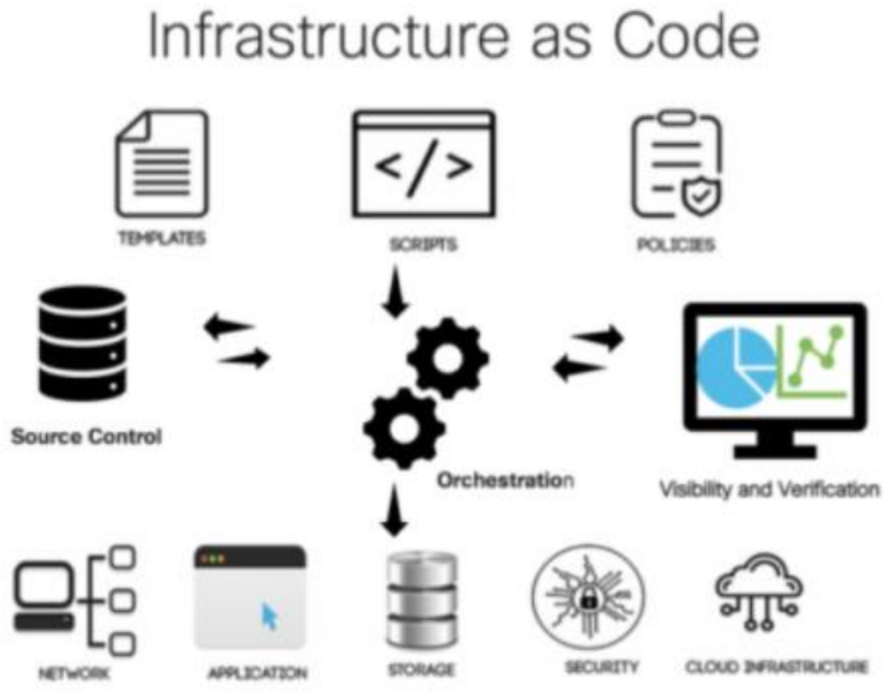
What is Infrastructure as Code?

CLARUSWAY©
WAY TO REINVENT YOURSELF

What is Infrastructure as Code (IaC)?

- **IaC** is the process of managing infrastructure in a file or files rather than manually configuring resources in a user interface.

- A resource in an instance is any piece of infrastructure in a given environment, such as a virtual machine, security group, network interface, etc.



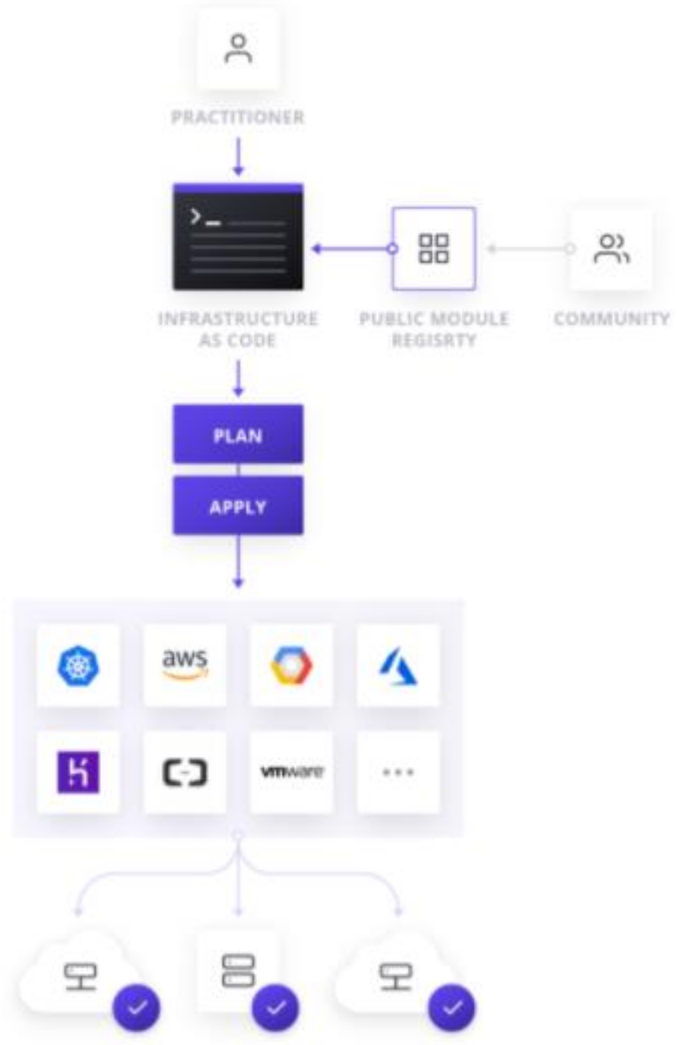
3

How Terraform Works

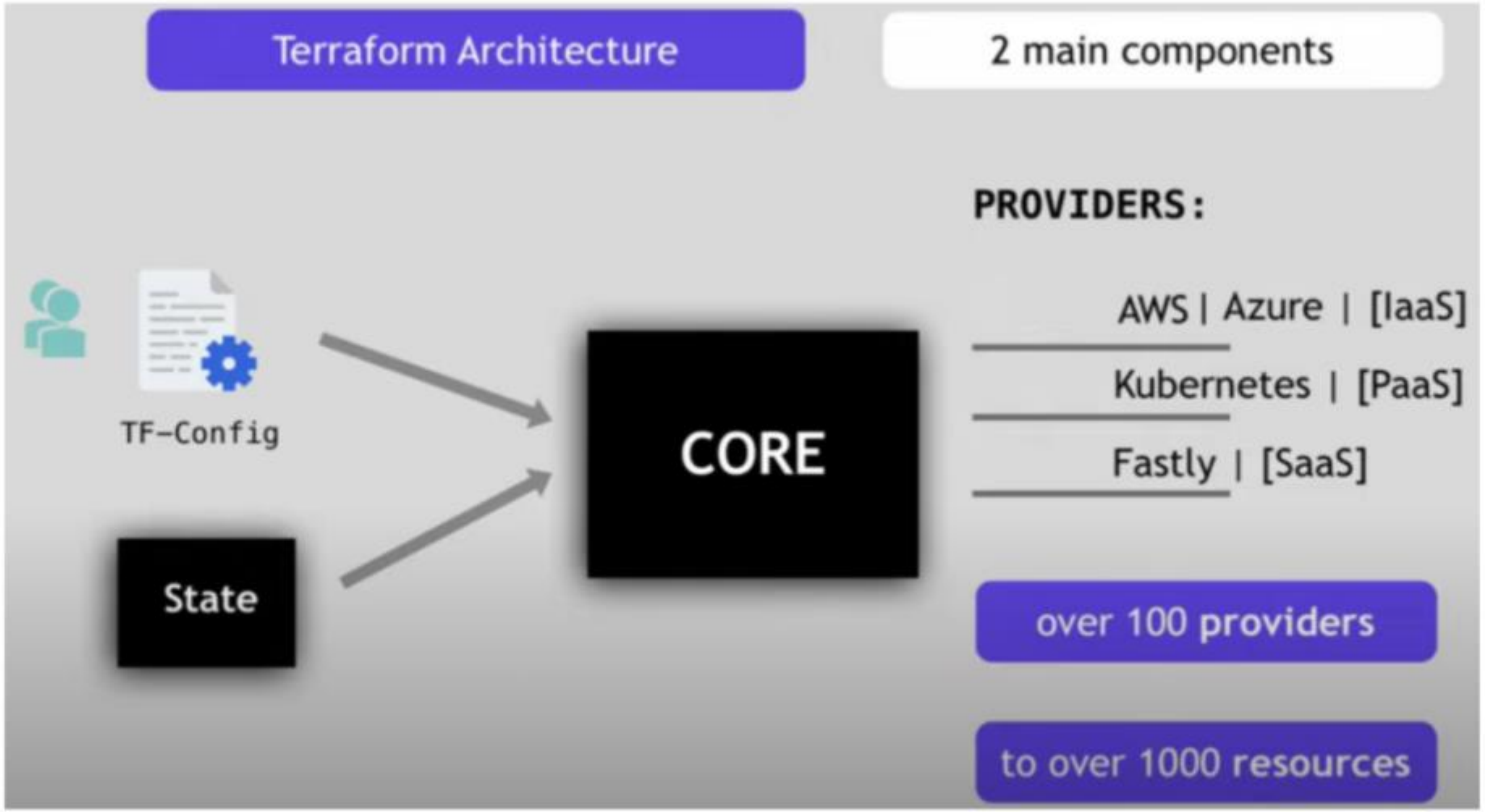
How Terraform Works

- Terraform allows infrastructure to be expressed as code in a simple, human readable language called **HCL** (HashiCorp Configuration Language).

- Terraform CLI reads configuration files and provides an execution plan of changes, which can be reviewed for safety and then applied and provisioned.



How Terraform Works



4

Workflows

Workflows

A **simple workflow** for deployment will follow closely to the steps below.

Scope: Confirm what resources need to be created for a given project

Author: Create the configuration file in HCL based on the scoped parameters

Initialize: Run terraform init in the project directory with the configuration files. This will download the correct provider plug-ins for the project.

Plan & Apply: Run terraform plan to verify creation process and then terraform apply to create real resources as well as state file that compares future changes in your configuration files to what actually exists in your deployment environment.

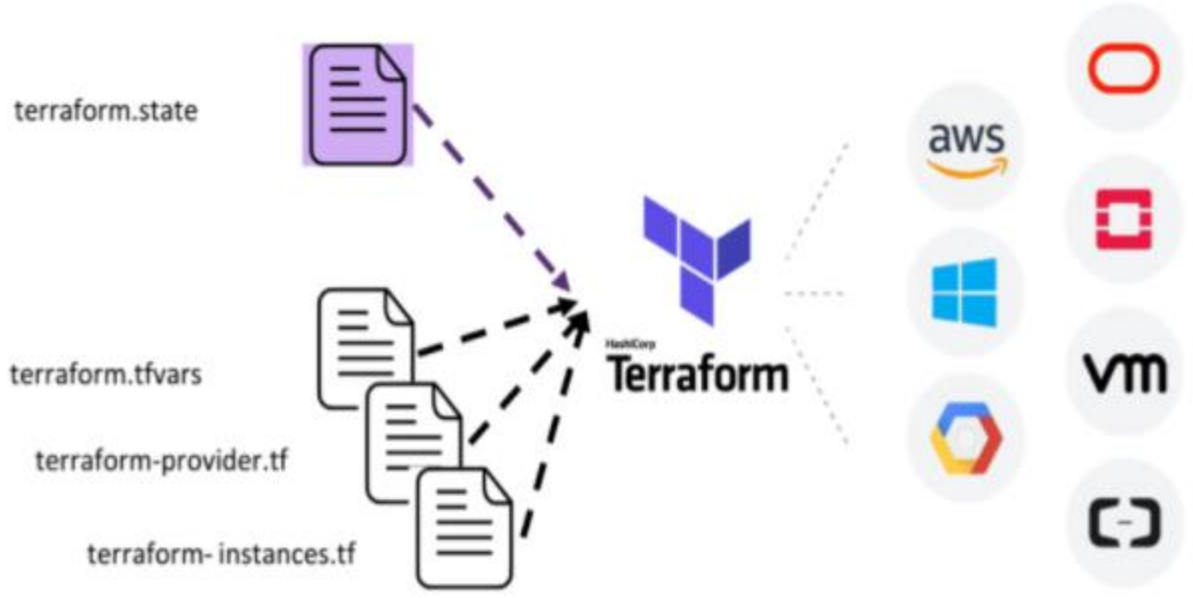
5

Terraform Elements

Terraform Elements

State

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. Terraform uses this local state to create plans and make changes to your infrastructure. State is a necessary requirement for Terraform to function.



Terraform Elements

Providers

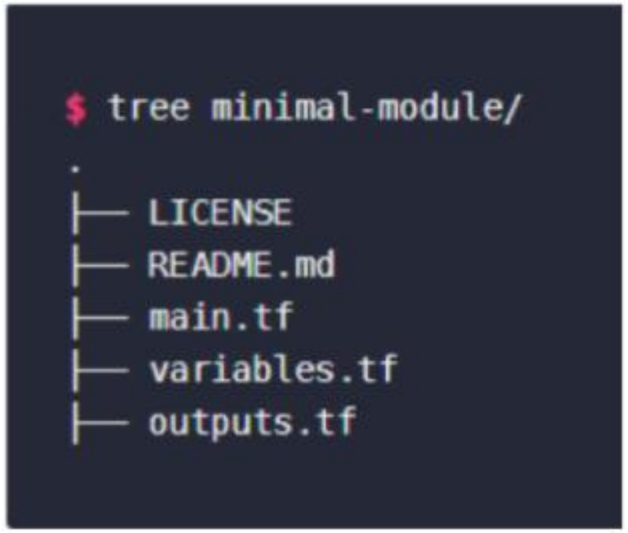
A provider is responsible for understanding API interactions and exposing resources. Every Terraform provider has its own documentation, describing its resource types and their arguments.

amazon web services
Google Cloud Platform
Azure
kubernetes
HashiCorp Nomad
openstack
PostgreSQL
Bitbucket
docker
CONSUL
GitHub
New Relic.
DATADOG
dnsimple
vmware vSphere

Terraform Elements

Modules

A Terraform module is a set of Terraform configuration files in a single directory. Even a simple configuration consisting of a single directory with one or more «.tf» files is a module. When you run Terraform commands directly from such a directory, it is considered the root module. So in this sense, every Terraform configuration is part of a module.



```
$ tree minimal-module/
.
├── LICENSE
├── README.md
├── main.tf
├── variables.tf
├── outputs.tf
```

Terraform Elements

Backends

A "backend" in Terraform determines how state is loaded and how an operation such as <apply> is executed. By default, Terraform uses the "local" backend, which is the normal behavior of Terraform you're used to. Backends are completely optional. You can successfully use Terraform without ever having to learn or use backends. Backends are used for Keeping sensitive information off disk.

HashiCorp Terraform

Terraform
Using AWS S3
Remote Backend

[data]

Amazon S3

6 Advantages of Terraform

Advantages of Terraform

- **Platform Agnostic**
- **State Management**
- **Operator Confidence**

7

Terraform & Ansible

Terraform & Ansible

ANSIBLE

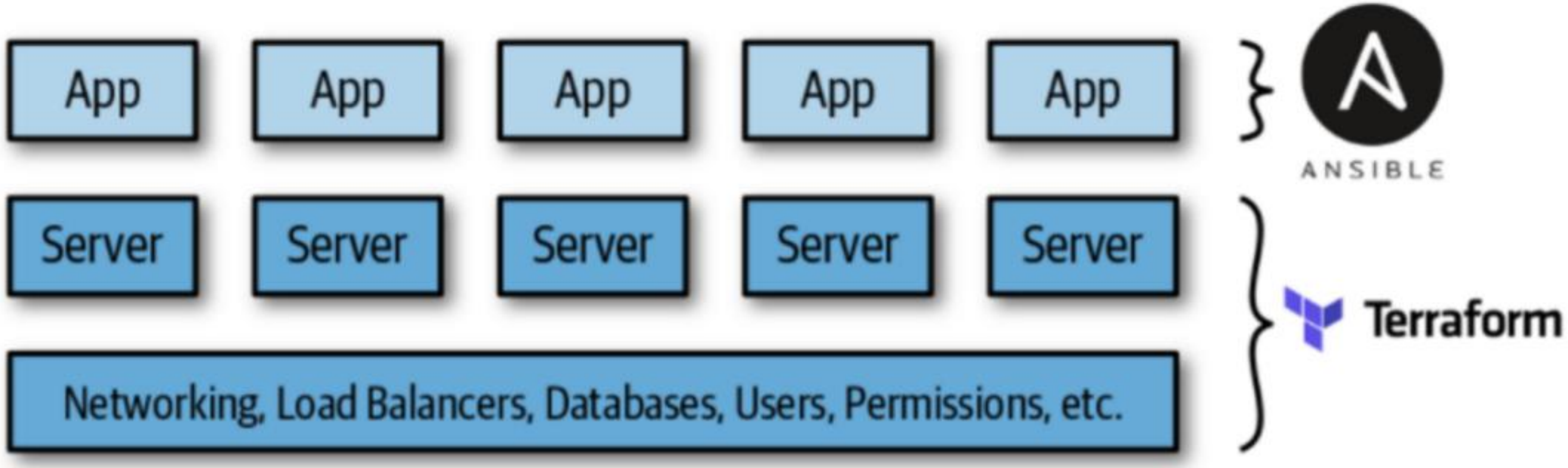
You can use **both**

HashiCorp Terraform

Better: for configuring that infrastructure

Better: for infrastructure

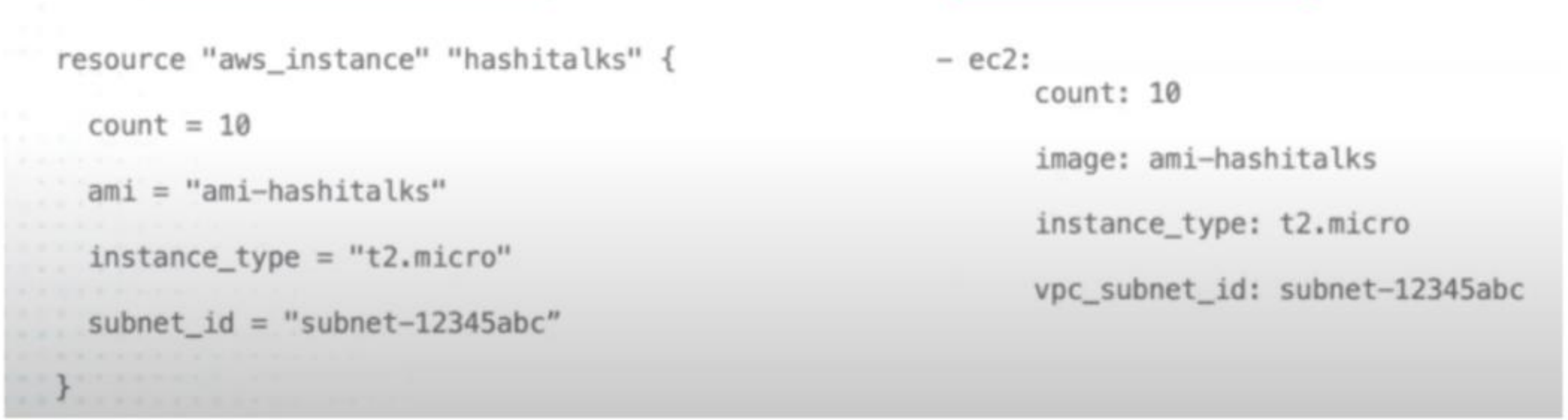
Terraform & Ansible


App
App
App
App
App
ANSIBLE
Server
Server
Server
Server
Server
Terraform
Networking, Load Balancers, Databases, Users, Permissions, etc.

Terraform & Ansible

Sample Terraform Code

Sample Ansible Code



```hcl
resource "aws_instance" "hashitalks" {

  count = 10

  ami = "ami-hashitalks"

  instance_type = "t2.micro"

  subnet_id = "subnet-12345abc”

}
```

```yaml
- ec2:
     count: 10

     image: ami-hashitalks

     instance_type: t2.micro

     vpc_subnet_id: subnet-12345abc
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

8

Learn More Terraform

Learn More Terraform

- Terraform Documentation
- Hashicorp/terraform (Github Page)
- Shuaibiyy/awesome-terraform
- tfutils/tfenv
- gruntwork-io/terragrunt
- 28mm/blast-Radius
- Terraform Registry

THANKS!

Any questions?

You can find me at:

- oliver@clarusway.com