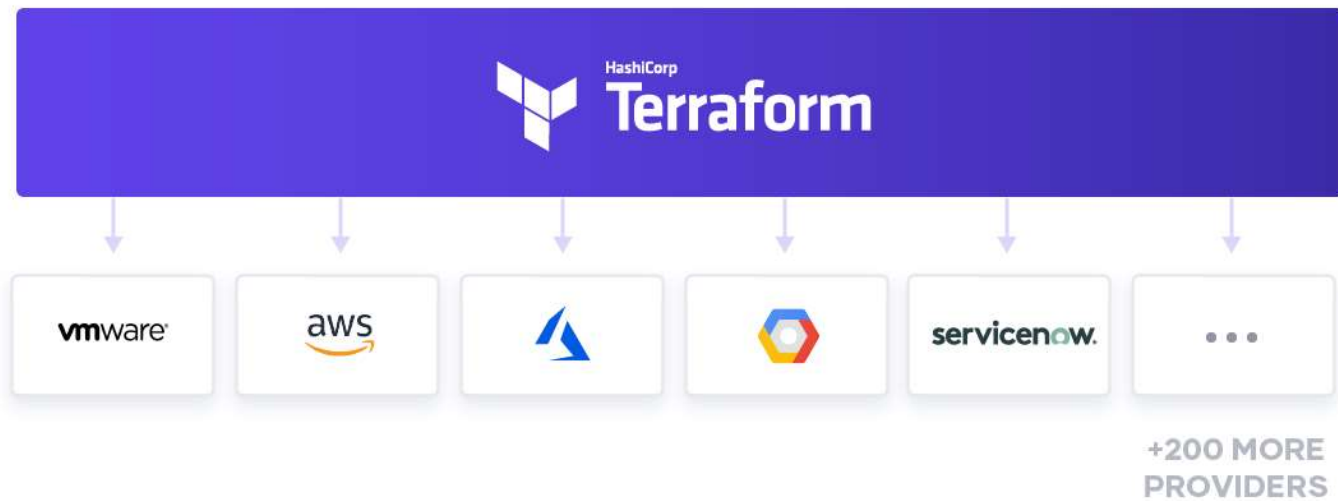


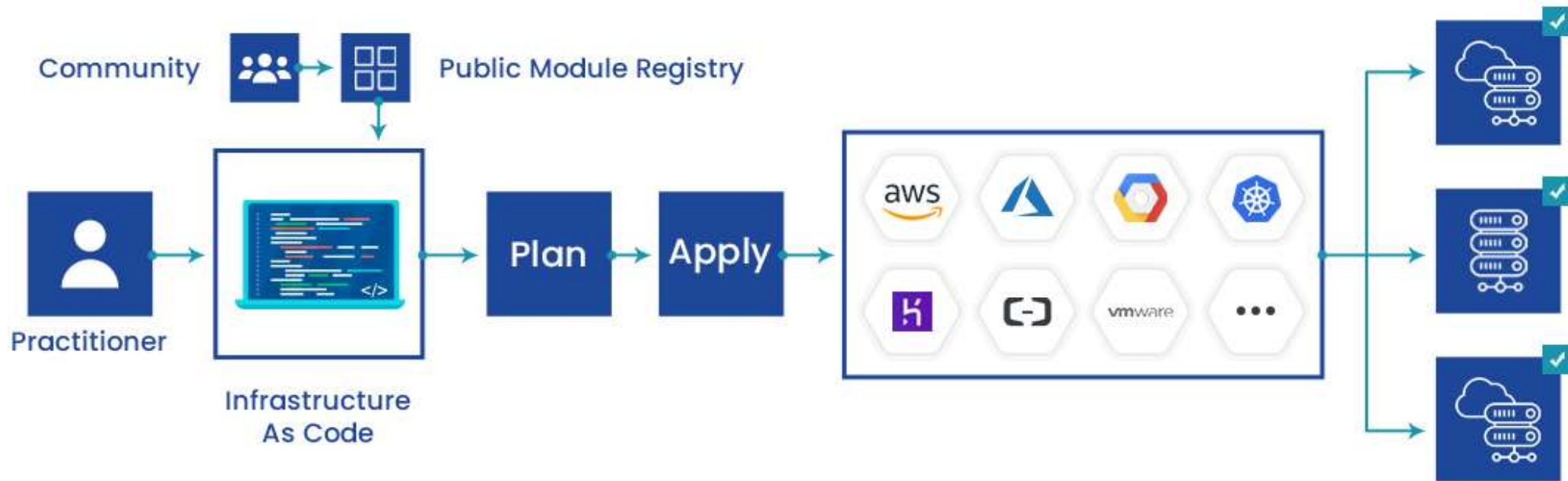
HashiCorp

Terraform

Introduction



Introduction



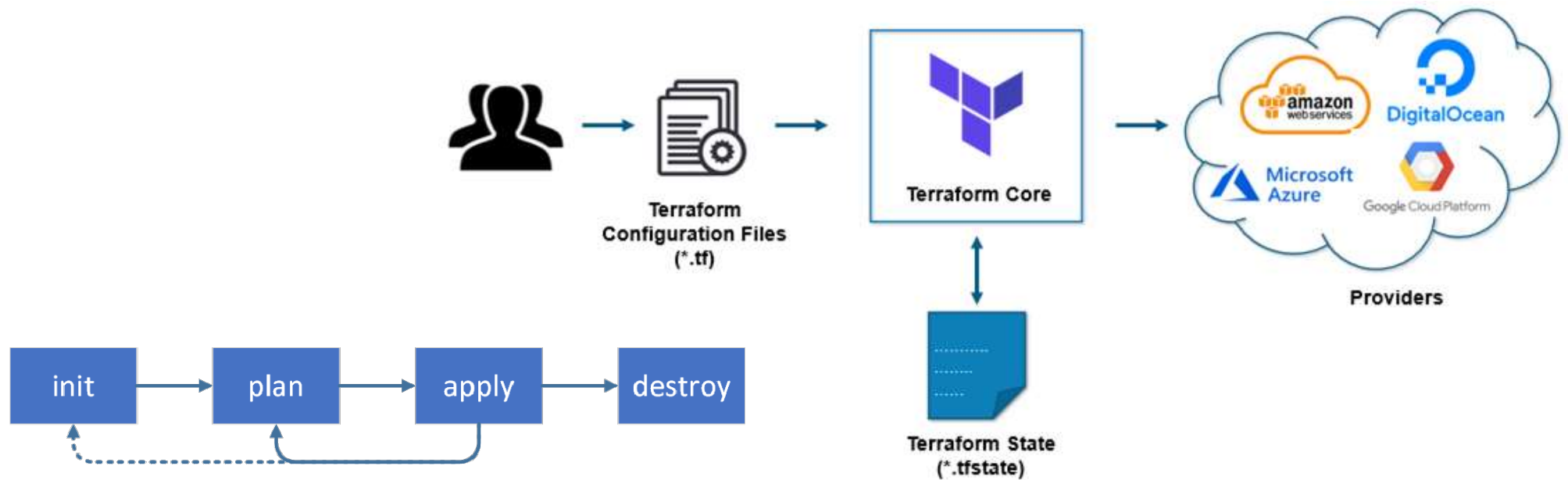
What is Terraform?

- Tool for
 - Building,
 - Changing, and
 - Versioning infrastructure safely and efficiently
- Can manage
 - Existing and popular service providers as well as
 - Custom in-house solutions.

The key features of Terraform

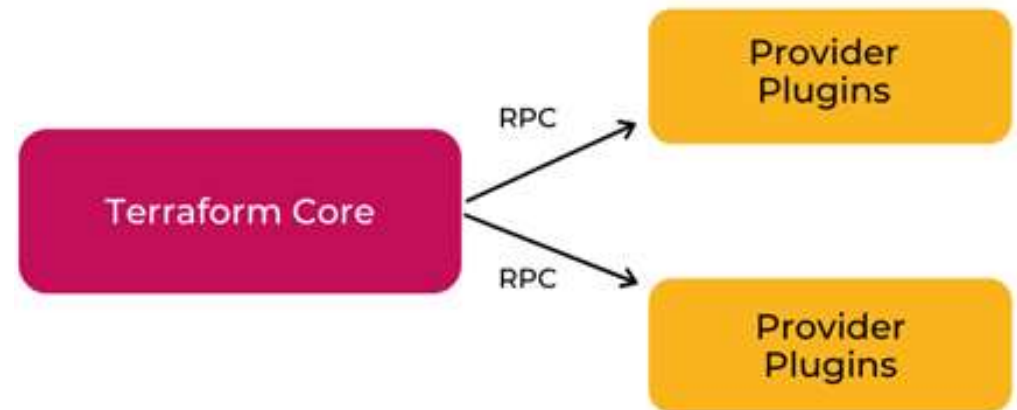
- Infrastructure as Code
- Execution Plans
- Resource Graph
- Change Automation

How Terraform works?



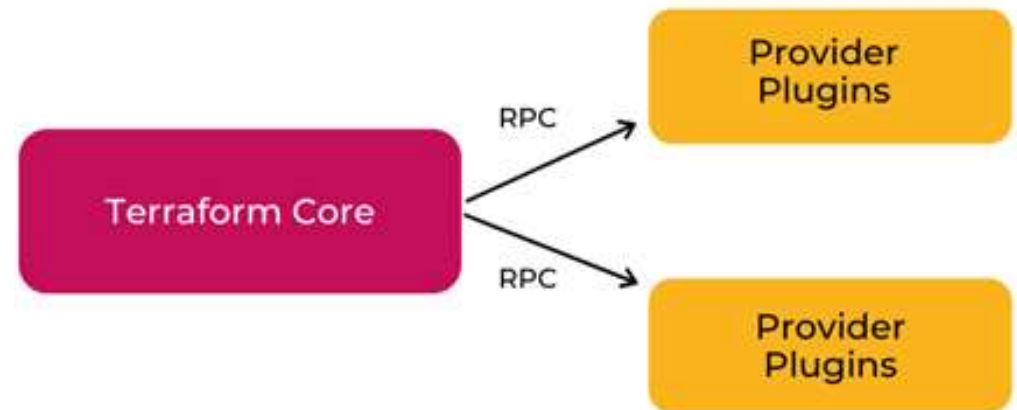
Terraform Core

- Written in the Go programming language
- The entry-point for anyone using Terraform
- Primary responsibilities
 - Reading configuration files
 - Resource state management
 - Construction of the Resource Graph
 - Plan execution
 - Communication with plugins



Terraform Plugins

- Each plugin exposes an implementation for a specific service
 - such as AWS, Azure
- Plugin Locations
 - ~/.terraform.d/plugins



Install Terraform

- `curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add –`
- `sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"`
- `sudo apt-get update && sudo apt-get install terraform`
- `terraform –help`
- `terraform -install-autocomplete`
- Refer: 0-Install-Terraform.md

```
aws_vpc.main: Creating...
arn:                               "" => "<computed>"
assign_generated_ipv6_cidr_block:  "" => "false"
cidr_block:                         "" => "10.11.0.0/16"
default_network_acl_id:             "" => "<computed>"
default_route_table_id:             "" => "<computed>"
default_security_group_id:          "" => "<computed>"
dhcp_options_id:                   "" => "<computed>"
enable_classiclink:                 "" => "<computed>"
enable_classiclink_dns_support:     "" => "<computed>"
enable_dns_hostnames:              "" => "true"
enable_dns_support:                 "" => "true"
instance_tenancy:                  "" => "default"
ipv6_association_id:               "" => "<computed>"
ipv6_cidr_block:                   "" => "<computed>"
main_route_table_id:               "" => "<computed>"
owner_id:                          "" => "<computed>"
tags.%:                            "" => "2"
tags.Environment:                  "" => "Test"
tags.Name:                         "" => "Test-VPC"
aws_vpc.main: Still creating... (10s elapsed)
aws_vpc.main: Creation complete after 17s (ID: vpc-0e94a7d72c02dab2b)
aws_subnet.test: Creating...
arn:                               "" => "<computed>"
assign_ipv6_address_on_creation:    "" => "false"
availability_zone:                  "" => "us-east-1a"
cidr_block:                         "" => "10.11.1.0/24"
ipv6_cidr_block:                   "" => "<computed>"
ipv6_cidr_block_association_id:     "" => "<computed>"
map_public_ip_on_launch:           "" => "false"
owner_id:                          "" => "<computed>"
tags.%:                            "" => "1"
tags.Name:                         "" => "Test_subnet1"
vpc_id:                            "" => "vpc-0e94a7d72c02dab2b"
aws_subnet.test: Creation complete after 6s (ID: subnet-0ad06c2e86542ddc1)
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS C:\Users\Administrator\projects\terraform>
```

Procedural vs Declarative

Ansible

```
# Add 5 more

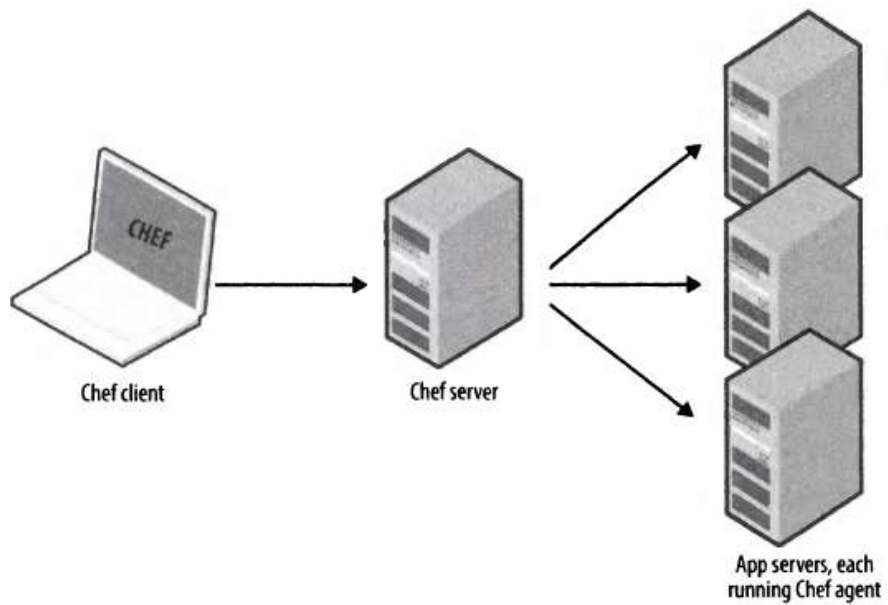
- ec2:
    count: 5
    image: aws-ami
    Instance_type: t2.micro
```

Terraform

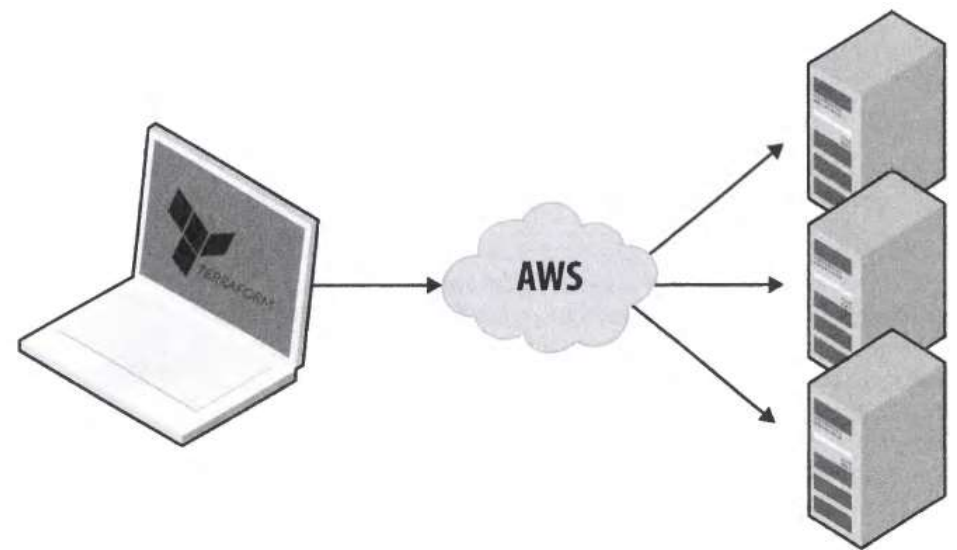
```
# Make sure that we have 5

resource "aws_instance"
"ec2ins" {
    count = 5
    ami = "aws-ami"
    instance_type = "t2.micro"
}
```

Agent vs Agentless



15 February 2021



Terraform

11

An Introduction to Terraform Syntax

- Called HashiCorp Configuration Language (HCL)
- Human readable as well as machine-friendly
 - `# An AMI`
 - `variable "ami" {`
 - `description = "the AMI to use"`
 - `}`

 - `resource "aws_instance" "web" {`
 - `ami = "${var.ami}"`
 - `count = 2`
 - `source_dest_check = false`
 - `connection {`
 - `user = "root"`
 - `}`
 - `}`

Thanks

How to create reusable infrastructure

Module basics

Module inputs

Module outputs

Versioned modules

Module basics

- Any set of Terraform configuration files in a folder is a module.
 - `$ tree minimal-module/`
 - `.`
 - `|— README.md`
 - `|— main.tf`
 - `|— variables.tf`
 - `|— outputs.tf`
- A module can call other modules

Calling a Child Module

- `module "servers" {`
- `source = "../app-cluster"`
- `servers = 5`
- `}`

Module inputs

- Parameters for a Terraform module
- Like function arguments
 - `variable "image_id" {`
 - `type = string`
 - `}`

 - `variable "availability_zone_names" {`
 - `type = list(string)`
 - `default = ["us-west-1a"]`
 - `}`

Using Input Variable Values

- `var.<NAME>`
- `resource "aws_instance" "example" {`
- `instance_type = "t2.micro"`
- `ami = var.image_id`
- `}`

Variable Definitions (.tfvars) Files

- To set lots of variables, it is more convenient to specify their values in a variable definitions file
- And then specify that file on the command line with -var-file
 - `terraform apply -var-file="testing.tfvars"`

Module outputs

- Like the return values of a Terraform module
 - `output "instance_ip_addr" {`
 - `value = aws_instance.server.private_ip`
 - `}`

Loops

- `resource "aws_lam_user" "example" {`
- `count = 3`
- `name = "neo.${count.Index}"`
- `}`

Getting Started & Setting Up Labs

- Install Terraform
- Choosing Right IDE for Terraform
 - - VSCode
- Use AWS account

Thanks