Terraform

Infrastructure as a Code

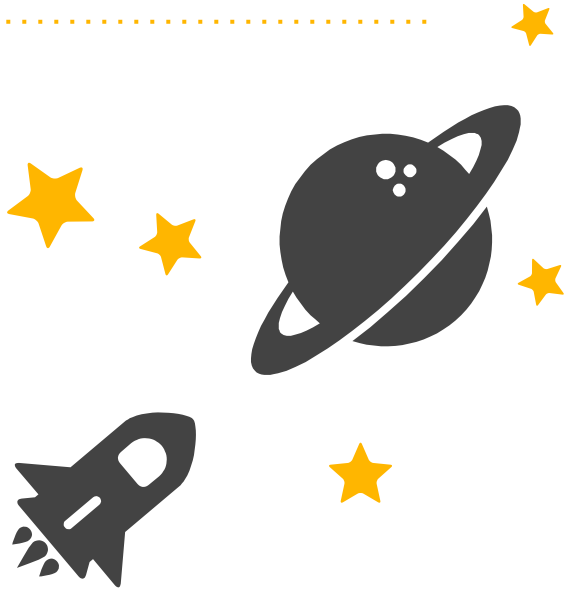
Introduction

- IAAC | Automate Infrastructure
- Define Infrastructure State
- Ansible, puppet or chef automates mostly OS related tasks.
  - Defines machines state
- Terraform automates infra itself
  - Like AWS, GCP, Azure, digital ocean etc

Introduction

- Terraform works with automation softwares like ansible after infra is setup and ready.
- No Programming, its own syntax similar to JSON.

Everything Needs Automation



Infrastructure automation centralized.

Installation

Download Terraform binary from its website

- Linux
- Mac
- Windows

Store binary in exported PATH

e:g: Linux => /usr/local/bin

Launch ec2 instance

- AWS Account
- IAM User with access keys
- Terraform file to launch instance
- Run terraform apply

Exercise

- ➔ Write instance.tf file
- ➔ Launch instance
- ➔ Make some changes to instance.tf file
- ➔ Apply changes.

Summarizing

instance.tf

```
provider "aws" {
  access_key = "ACCESS_KEY"
  secret_key = "SECRET_KEY"
  region = "ap-south-1"
}

resource "aws_instance" "intro" {
  ami = "ami-009110a2bf8d7dd0a"
  instance_type = "t2.micro"
}
```

Summarizing

terraform plan

+ create

Terraform will perform the following actions:

```
  # aws_instance.intro will be created
  + resource "aws_instance" "intro" {
      + ami                          = "ami-009110a2bf8d7dd0a"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)
      + availability_zone            = (known after apply)
```

Summarizing

terraform apply

aws_instance.intro: Creating...

aws_instance.intro: Still creating... [10s elapsed]

aws_instance.intro: Still creating... [20s elapsed]

aws_instance.intro: Still creating... [30s elapsed]

aws_instance.intro: Creation complete after 31s [id=i-047d7ea789e081807]

Summarizing

terraform destroy

Plan: 0 to add, 0 to change, 1 to destroy.

aws_instance.intro: Destroying... [id=i-047d7ea789e081807]

aws_instance.intro: Still destroying... [id=i-047d7ea789e081807, 10s elapsed]

aws_instance.intro: Still destroying... [id=i-047d7ea789e081807, 20s elapsed]

aws_instance.intro: Destruction complete after 29s

Destroy complete! Resources: 1 destroyed.

Variables

- Move secrets to another file
- Values that change
  - AMI, tags, keypair etc
- Reuse your code

instance.tf

```
provider "aws" {
  #access_key = "ACCESS_KEY"
  #secret_key = "SECRET_KEY"
  region = "ap-south-1"
}

resource "aws_instance" "intro" {
  ami = "ami-009110a2bf8d7dd0a"
  instance_type = "t2.micro"
}
```

providers.tf

```
provider "aws" {
  region = var.REGION
}
```

vars.tf

```
variable REGION {
  default = "us-west-1"
}
```

terraform.tfvars

```
AWS_ACCESS_KEY = ""
AWS_SECRET_KEY = ""
```

instance.tf

```
resource "aws_instance" "intro" {
  ami = "ami-009110a2bf8d7dd0a"
  instance_type = "t2.micro"
}
```

providers.tf

```
provider "aws" {
  region = var.REGION
}
```

instance.tf

```
resource "aws_instance" "intro" {
  ami = var.AMIS[var.REGION]
  instance_type = "t2.micro"
}
```

vars.tf

```
variable AWS_ACCESS_KEY {}
variable AWS_SECRET_KEY {}
variable REGION {
  default = "us-west-1"
}
variable AMIS {
  type = "map"
  default {
    us-west-1 = "ami-06397100adf427136"
    us-west-2 = "ami-a042f4d8"
  }
}
```

Exercise

- ➔ Write providers.tf file
- ➔ Write vars.tf file
- ➔ Write instance.tf file
- ➔ Apply Changes
- ➔ Make some changes to instance.tf file
- ➔ Apply changes.

Provisioning

- Build Custom Images with tools like **packer**

- Use standard Image and use provisioner to setup softwares and files.
  - File uploads
  - remote_exec
  - Ansible, Puppet or Chef

Provisioner Connection

Requires Connection for provisioning.

SSH

```
provisioner "file" {
  source      = "files/test.conf"
  destination = "/etc/test.conf"

  connection {
    type     = "ssh"
    user     = "root"
    password = var.root_password
  }
}
```

WinRM

```
provisioner "file" {
  source      = "conf/myapp.conf"
  destination = "C:/App/myapp.conf"

  connection {
    type     = "winrm"
    user     = "Administrator"
    password = var.admin_password
  }
}
```

More Provisioner

- The **file** provisioner is used to copy files or directories
- **remote-exec** invokes a command/script on remote resource.
- **local-exec** provisioner invokes a local executable after a resource is created

More Provisioner

- The **puppet** provisioner installs, configures and runs the Puppet agent on a remote resource.
  - Supports both ssh and winrm type connections.

- The **chef** provisioner installs, configures and runs the Chef Client on a remote resource.
  - Supports both ssh and winrm type connections.

- Ansible : run terraform, Output IP address, run playbook with **local-exec**

Variables

```
variable "PRIV_KEY_PATH" {
  default = "infi-inst_key"
}
variable "PUB_KEY_PATH" {
  default = "infi-inst_key.pub"
}
variable "USER" {
  default = "ubuntu"
}
```

Key Pair & instance Resources

```
resource "aws_key_pair" "dove-key" {
  key_name = "dovekey"
  public_key = file("dovekey.pub")
}

resource "aws_instance" "intro" {
  ami = var.AMIS[var.REGION]
  instance_type = "t2.micro"
  availability_zone = var.ZONE1
  key_name = aws_key_pair.dove-key.key_name
  vpc_security_group_ids = ["sg-833e24fd"]
  }
```

File Provisioner

```
provisioner "file" {
    source = "web.sh"
    destination = "/tmp/web.sh"
   connection {
    user = var.USER
    private_key =
file(var.PRIV_KEY_PATH)
    host = self.public_ip
  }
```

```
variable "PRIV_KEY_PATH" {
  default = "infi-inst_key"
}
variable "PUB_KEY_PATH" {
  default = "infi-inst_key.pub"
}
variable "USER" {
  default = "ubuntu"
}
```

Remote-exec Provisioner

```
provisioner "remote-exec" {
   inline = [
     "chmod u+x /tmp/web.sh",
     "sudo /tmp/web.sh"
   ]
 }
```

Exercise

- Generate key pair
- Write script
- Write providers.tf
- Write vars.tf
- Write instances.tf
  - key pair resource
  - aws_instance resource
    - provisioners
      - file
      - remote-exec
- Apply changes.

Output Information

- Terraform stores returned value of all resources created.
  - e:g **aws_instance** resource has the **attribute public_ip**
- Use **output** block to print these attributes.
- local-exec to save info to a file

Output Attributes

```
output "instance_ip_addr" {
  value = aws_instance.server.public_ip
}
```

Elements => resourceType.resourceName.attributeName

resourceType => aws_instance
resourceName => server
attributeName => public_ip

Store Output in File

```
resource "aws_instance" "out_inst" {
 ami = var.AMIS[var.REGION]
 instance_type = "t2.micro"
 key_name = aws_key_pair.dino-key.key_name

 provisioner "local-exec" {
    command = "echo aws_instance.out_inst.private_ip   >>
private_ips.txt"
  }
}
```