**Terraform**

Cloud automation

Languages: HashiCorp Configuration Language(HCL),JSON

Stateful

Version V1.1.5

Hashicorp Terraform(plugin) – Visual studio

Terraform

Terraform Script

//code//

terraform

{

required\_providers

{

aws= {

source =”hashicorp/aws”

version = “~> 3.0”

}

provider “aws” {

r egion =”us-east-1”

access\_key =””

secret\_key =””

}

//code//

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 3.0"

}

}

}

provider "aws" {

region = "ap-south-1"

}

resource "aws\_instance" "myinstance01" {

ami = "ami-0c6615d1e95c98aca"

instance\_type = "t2.small"

count = "1"

tags = {

Name = "Server\_ONE"

}

}

resource "aws\_instance" "myinstance02" {

ami = "ami-0c6615d1e95c98aca"

instance\_type = "t2.small"

count = "1"

tags = {

Name = "Server\_TWO"

}

}

Commands

Step1: terraform init //initialization

Step2: terraform validate //checking the configuration

Step3:terraform plan //like dry run checking with aws

Step4:terraform apply

Terraform.tfstate- has all the information of instances

Step5:Again terraform apply

Step6: say apply

Launching EC2 Instance

* Infrastructure as a code

**Prerequistes required for Terraform**

AWS Account

IAM User with access keys

Terraform file to launch instance

RUN Terraform Apply

**Installation on Windows machine**

Step1: install choco and then install choco install terraform

**first\_instance.tf**

provider "aws" {

  region     = "us-east-1"

  shared\_credentials\_file = "~/.aws/credentials"

    }

resource "aws\_instance" "Demo2" {

  ami           = "ami-0c4f7023847b90238"

  instance\_type = "t2.micro"

  count = 2

  key\_name = "awstestkey"

  tags = {

    Name = "Demo1"

  }

}

Provider indicates which cloud

Under we can keeo region,shared\_credentials\_file

Resource

type aws instance

**Terraform Using Variables**

**Providers.tf**

Provider “aws” {

region = var.REGION

}

**Vars.tf**

variable REGION {

default = “us-west-1” // “us-east-1”

}

**Terraform.tfvars**

AWS\_ACCESS\_KEY = ‘’

AWS\_SECRET\_KEY **= ‘’**

**Instance.tf**

resource “aws\_instance” “intro” {

ami = “ami-009110a2b

instance\_type = “t2.micro”

}

Provisiononing

* 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

**For Linux machine**

SSH

provisioner “file” {

source = “files/test.conf”

destination = “/etc/test.conf”

}

connection {

type =”ssh”

user =”root”

password =var.root\_password

}

}

**For windows Machine**

**WinRM**

provisioner “file” {

source = “conf/myapp.conf”

destination = “C:/App/myapp.conf”

}

connection {

type =”winrm”

user =”Administrator”

password =var.admin\_password

}

The file provisioner is used to copy files files or directories

* Remote-exec invokes a command/script on remote resource]
* Local-exec provisioner invokes a local executable after a resource is created

**Create private and publickey and push to aws**

**Modules**

Ready made packages