**Terraform – infrastructure as code (IAC)**

**Terraform installation in Linux:**

1. **Sudo su**
2. **wget** [**https://releases.hashicorp.com/terraform/1.0.0/terraform\_1.0.0\_linux\_amd64.zip**](https://releases.hashicorp.com/terraform/1.0.0/terraform_1.0.0_linux_amd64.zip)
3. **unzip terraform\_1.0.0\_linux\_amd64.zip**
4. **mv terraform /usr/bin/**
5. **check „terraform --version“**

**How to install terraform in windows:**

1. Go to Browser “ <https://www.terraform.io/downloads.html> ”
2. Select OS of system “where you want to install terraform”
3. Select folder -> unzip war file
4. set “path for Terraform in environmental variables”

Type “edit the system environmental variables” -> select “advanced” -> click on “environmental variables” -> select “system variables” -> click on “path” -> click on “New” -> enter the “Terraform path” -> click on “Ok”

1. Go to visual studio code -> install “ HashiCorp Terraform ”
2. In Visual studio code open folder where you installed terraform -> open new Terminal (bash)
3. Type: terraform

**Main commands:**

init Prepare your working directory for other commands

validate Check whether the configuration is valid

plan Show changes required by the current configuration

apply Create or update infrastructure

destroy Destroy previously-created infrastructure

**All other commands:**

Console Try Terraform expressions at an interactive command prompt

Fmt Reformat your configuration in the standard style

force-unlock Release a stuck lock on the current workspace

get Install or upgrade remote Terraform modules

graph Generate a Graphviz graph of the steps in an operation

import Associate existing infrastructure with a Terraform resource

login Obtain and save credentials for a remote host

logout Remove locally-stored credentials for a remote host

output Show output values from your root module

providers Show the providers required for this configuration

refresh Update the state to match remote systems

show Show the current state or a saved plan

state Advanced state management

taint Mark a resource instance as not fully functional

test Experimental support for module integration testing

untaint Remove the 'tainted' state from a resource instance

version Show the current Terraform version

workspace Workspace management

**Global options (use these before the subcommand, if any):**

-chdir=DIR Switch to a different working directory before executing the

given subcommand.

-help Show this help output, or the help for a specified subcommand.

-version An alias for the "version" subcommand.

1. Check version “terraform --version”
2. terraform –help
3. go to aws and create user and provider programmatic access
4. make directory “mkdir demo01” ===> This folder is called terraform configuration folder.
5. cd demo01
6. Under this folder we need to create below files
7. terraform.tfvars

AWS\_ACCESS\_KEY="AKIAY4AQH2IWERNGCPXW"

AWS\_SECRET\_KEY="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"

1. providers.tf

provider "aws" {

access\_key=var.AWS\_ACCESS\_KEY

If you don’t have “terraform.tfvars” you need to provide details in “providers.tf” file.

   secret\_key=var.AWS\_SECRET\_KEY

  region=var.AWS\_REGION

}

1. terraform init ===> by using above credentials terraform downloads “AWS Plugin”
2. now I need to create resources for this I created file **“instance.tf”**

resource "aws\_instance" "demo01" {

   ami           = "0aeeebd8d2ab47354"

   instance\_type = "t2.micro"

 tags = {

     Name = "HelloWorld"

   }

}

1. Create **“vars.tf”**

variable "AWS\_ACCESS\_KEY" { }

variable "AWS\_SECRET\_KEY" { }

variable "AWS\_REGION" {

  default = "us-east-1"

}

1. terraform fmt
2. terraform validate
3. terraform plan
4. terraform apply
5. check and enter “yes”
6. terraform destroy

**note:** If you don’t have default vpc,subnet so create in exiting subnet.

resource "aws\_instance" "demo01" {

  ami               = "ami-0aeeebd8d2ab47354"

  instance\_type     = "t2.micro"

  availability\_zone = "us-east-1a"

  subnet\_id         = "subnet-0d8d68f3ac0b29605"

  tags = {

    Name = "HelloWorld"

  }

}

**Terraform script to create AWS resources:**

**In “terraform.tfvars”**

AWS\_ACCESS\_KEY = "AKIAY4AQH2IWERNGCPXW"

AWS\_SECRET\_KEY = "yZtdNVuPFjdjXPKqs25SdADATFgngHyFYs5voyxx"

**In “resource.tf”**

# variable details

variable "AWS\_ACCESS\_KEY" {}

variable "AWS\_SECRET\_KEY" {}

# AWS Provider

provider "aws" {

  access\_key = var.AWS\_ACCESS\_KEY

  secret\_key = var.AWS\_SECRET\_KEY

  region     = "us-east-2"

}

# creating Resources like VPC , SUBNET , IGW

resource "aws\_vpc" "vpc\_dev" {

  cidr\_block = "172.16.0.0/16"

  tags = {

    Name = "vpc\_dev"

  }

}

resource "aws\_subnet" "public" {

  vpc\_id            = aws\_vpc.vpc\_dev.id

  cidr\_block        = "172.16.0.0/24"

  availability\_zone = "us-east-2a"

  tags = {

    Name = "public"

  }

}

resource "aws\_subnet" "private" {

  vpc\_id            = aws\_vpc.vpc\_dev.id

  cidr\_block        = "172.16.10.0/24"

  availability\_zone = "us-east-2b"

  tags = {

    Name = "private"

  }

}

resource "aws\_internet\_gateway" "IGW" {

  vpc\_id = aws\_vpc.vpc\_dev.id

  tags = {

    Name = "IGW"

  }

}

# Creating EC2 Instance

resource "aws\_instance" "myinstance" {

  ami           = "ami-0d8d212151031f51c"

  instance\_type = "t2.micro"

  subnet\_id     = aws\_subnet.public.id

  tags = {

    Name = "terraform instance"

  }

}

# Security details

resource "aws\_security\_group" "sg\_allow\_ssh\_jenkins" {

  name        = "allow\_ssh\_jenkins"

  description = "Allow SSH and Jenkins inbound traffic"

  vpc\_id      = "${aws\_vpc.vpc\_dev.id}"

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 8080

    to\_port     = 8080

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }