***Assignment1***

**Create a file called main.tf.**

**Create a docker image resource and call it nginx\_image.**

**Set the name of the image to nginx:latest.**

**Save and exit the file.**

**Main.tf**

terraform {

required\_providers {

docker = {

source = "kreuzwerker/docker"

version = "2.12.0"

}

}

}

provider "docker" {

host = "npipe:////.//pipe//docker\_engine"

}

# Pulls the image

resource "docker\_image" "nginx\_image" {

name = "nginx:latest"

}

***Assignment2***

**Initialize Terraform.**

**Plan the deploy and output a terraform plan called tf\_image\_plan.**

**Apply the plan using tf\_image\_plan.**

**Create a new Terraform file called main.tf.**

**Create three variables.**

**The first variable, called image\_name, needs to be set to ghost:latest.**

**The second variable is called container\_name with a default of ghost\_blog.**

**The final variable is called ext\_port and set the default to port 80.**

**Create a Docker image resource called ghost\_image that uses the image\_name variable.**

**Create a Docker container resource called ghost\_container.**

**The name will use the container\_name variable.**

**The image will use the ghost\_image resource.**

**The internal port will be set to 2368.**

**The external port will use ext\_port variable.**

**Initialize Terraform.**

**Create a Terraform plan that uses the following variables:**

**container\_name = ghost\_blog1**

**image\_name = ghost:alpine**

**ext\_port = 8080**

**Output the plan to a file called tfplan.**

**Then apply the plan using tfplan and make sure that the apply doesn’t prompt for inpu**t.

**Main.tf**

variable "image\_name" {}

variable "container\_name" {}

variable "ext\_port" {}

terraform {

required\_providers {

docker = {

source = "kreuzwerker/docker"

version = "2.12.0"

}

}

}

provider "docker" {

host = "npipe:////.//pipe//docker\_engine"

}

# Pulls the image

resource "docker\_image" "ghost\_image" {

name = "${var.image\_name}"

}

#creating the container

resource "docker\_container" "ghost\_container" {

name = "${var.container\_name}"

image = "${docker\_image.ghost\_image.latest}"

ports {

internal = "2368"

external = "${var.ext\_port}"

}

}

**Variables.tfvar**

image\_name = "ghost:latest"

container\_name = "ghost\_blog"

ext\_port = "80"

***Assignment 3***

**Adding Maps and Lookups in your Terraform files**

**The lab files can be found @ https://github.com/satyensingh/terraform-assignment-resources.git**

**In the repo you will find main.tf, outputs.tf, and variables.tf.**

**Add a new variable called env. Set a description to “env: dev or prod”.**

**Convert the type from image\_name to map.**

**Change the default to use key/value pairs. Set dev to ghost:latest and prod to ghost:alpine.**

**Convert container\_name to a map. Change the default to use key/value pairs. Set dev to blog\_dev and prod to blog\_prod.**

**Convert ext\_port to a map. Change the default to use key/value pairs. Set dev to 8080 and prod to 80.**

**Now initialize Terraform.**

**Setup the Development environment**

**Create a workspace called dev.**

**Generate a Terraform plan. Output the plan and call it tfdev\_plan. Pass in a variable called env and set it to dev.**

**Apply tfdev\_plan.**

**Setup the Production environment**

**Create a workspace called prod.**

**Generate a Terraform plan. Output the plan and call it tfprod\_plan. Pass in a variable called env and set it to prod.**

**Apply tfprod\_plan.**

**Verify both environments work**

**Open a browser and navigate to the public IP. This should pull up the production environment.**

**Open a browser tab and navigate to the public IP on port 8080. This should pull up the development environment.**

**Main.tf**

terraform {

required\_providers {

docker = {

source = "kreuzwerker/docker"

version = "2.12.0"

}

}

}

provider "docker" {

host = "npipe:////.//pipe//docker\_engine"

}

# Download the latest Ghost Image

resource "docker\_image" "image\_id" {

name = "${lookup(var.image\_name, var.env)}"

}

# Start the Container

resource "docker\_container" "container\_id" {

name = "${lookup(var.container\_name, var.env)}"

image = "${docker\_image.image\_id.latest}"

ports {

internal = "2368"

external = "${lookup(var.ext\_port, var.env)}"

}

}

**Variables.tf**

variable "env" {

description = "env: dev or prod"

default = "dev"

}

variable "image\_name" {

description = "Image for container."

type = map(string)

default = {

dev : "ghost:latest"

prod : "ghost:alpine"

}

}

variable "container\_name" {

description = "Name of the container."

type = map(string)

default = {

dev : "blog\_dev"

prod : "blog\_prod"

}

}

variable "ext\_port" {

description = "External port for container."

type = map(string)

default = {

dev : 9191

prod : 90

}

}

**Output.tf**

**Assignment 4**

**Create Ghost module**

**Create a directory called ghost.**

**Your modules will be made up of three files: main.tf, variables.tf and outputs.tf.**

**main.tf**

**In main.tf you will deploy out two resources docker\_image and docker\_container. The docker\_image resource name will be ghost\_image.**

**The name will use th image\_name variable.**

**The docker\_container resource name will be ghost\_container.**

**The name will be set using a varialbe called container\_name. The image will be set usingdocker\_image.ghost\_image.latest. Set the external port to use the ext\_port variable.**

**variables.tf**

**In variables.tf create three varialbes: image\_name, container\_name and ext\_port.**

**outputs.tf**

**In outputs.tf create two outputs: ip and container\_name. The ip output the ghost\_container's ip\_address attribute.**

**The container\_name output the ghost\_container's name attribute.**

**Create root module**

**main.tf**

**In cloud\_user directory create main.tf, variables.tf and outputs.tf. In main.tf will use the ghost module.**

**Set image\_name using a variable called image\_name.**

**Set container\_name using a variable called container\_name.**

**Set ext\_port using a variable called ext\_port.**

**In variables.tf create three image\_name, container\_name and ext\_port.**

**variables.tf**

**The image\_name will have a default value of ghost:latest with a description of Image for container.**

**The container\_name will have a default value of blogwith a description ofName of the container.**

**Theext\_portwill have adefaultvalue of 80 with a description of External port for container.**

**outputs.tf**

**In outputs.tf create two outputs: ip and container\_name. The ip output the ghost\_container's ip\_address attribute.**

**The container\_name output the ghost\_container's name attribute.**

**Deploy the infrastructure**

**Initialize Terraform.**

**Generate a Terraform plan and output a plan file.**

**Deploy the infrastructure using the plan file.**

**Root module**

**Main.tf**

module "ghost" {

source = "./ghost"

image\_name = "${var.image\_name}"

container\_name = "${var.container\_name}"

ext\_port = "${var.ext\_port}"

}

**Variables.tf**

#Define variables

variable "image\_name" {

default = "ghost:latest"

}

variable "container\_name" {

default = "blog"

}

variable "ext\_port" {

default = "9090"

}

**Outputs.tf**

#Output the IP Address of the Container

output "ip" {

value = "${module.ghost.ip}"

}

output "container\_name" {

value = "${module.ghost.container\_name}"

}

**ghost module**

**Main.tf**

terraform {

required\_providers {

docker = {

source = "kreuzwerker/docker"

version = "2.12.0"

}

}

}

provider "docker" {

host = "npipe:////.//pipe//docker\_engine"

}

# Download the Ghost Image

resource "docker\_image" "ghost\_image" {

name = "${var.image\_name}"

}

# Start the Ghost Container

resource "docker\_container" "ghost\_container" {

name = "${var.container\_name}"

image = "${docker\_image.ghost\_image.latest}"

ports {

internal = "2368"

external = "${var.ext\_port}"

}

}

**Variables.tf**

#Define variables

variable "image\_name" {}

variable "container\_name" {}

variable "ext\_port" {}

**Output.tf**

#Output the IP Address of the Container

output "ip" {

value = "${docker\_container.ghost\_container.ip\_address}"

}

output "container\_name" {

value = "${docker\_container.ghost\_container.name}"

}

***Assignment 5***

**Create the variables file (variables.tf) and add four variables with these default values:**

**1. container\_name: mysql.**

**2. mysql\_root\_password: P4sSw0rd0!.**

**3. mysql\_network\_name: mysql\_internal\_network.**

**4. mysql\_volume\_name: mysql\_data.**

**Create the images file (images.tf)**

**1. Add the docker\_image resource and call it mysql\_image.**

**2. Set the name to mysql:5.7.**

**Create the networks file (networks.tf):-**

**1. Add the docker\_network resource and call it private\_bridge\_network.**

**2. Set the name to use the mysql\_network\_name variable.**

**3. Set the driver to bridge.**

**4. Set internal to true.**

**Create the volumes file (volume.tf):-**

**1. In volumes.tf add the docker\_volume resource and call it mysql\_data\_volume.**

**2. Set the name to use the mysql\_volume\_name variable.**

**Create the main file (main.tf):-**

**1. In main.tf add the docker\_container resource and call it mysql\_container.**

**2. Set the name to use the container\_name variable.**

**3. Set the image to use the name of the image coming from docker\_image.**

**4. Create an environment variable for MYSQL\_ROOT\_PASSWORD and set it to the mysql\_root\_password variable.**

**5. Configure the container volume to use the volume created by docker\_volume, and make sure the container\_path is set to /var/lib/mysql.**

**7. The container needs to use the network created by docker\_network.**

**Deploy the infrastructure**

**1. Initialize Terraform.**

**2. Validate the files.**

**3. Generate a Terraform plan.**

**4. Deploy the infrastructure using the plan file.**

**Image.tf**

resource "docker\_image" "mysql\_image" {

name = "mysql:5.7.33"

}

**Networks.tf**

resource "docker\_network" "private\_bridge\_network" {

name = "${var.mysql1\_network\_name}"

driver = "bridge"

internal = "true"

}

**Variables.tf**

variable "container\_name" {

default = "mysql"

}

variable "mysql\_root\_password" {

default = "P4sSw0rd0!"

}

variable "mysql1\_network\_name" {

default = "mysql1\_internal\_network"

}

variable "mysql\_volume\_name" {

default = "mysql\_data"

}

**Volumes.tf**

resource "docker\_volume" "mysql\_data\_volume" {

name = "${var.mysql\_volume\_name}"

}

**Main.tf**

terraform {

required\_providers {

docker = {

source = "kreuzwerker/docker"

version = "2.12.0"

}

}

}

provider "docker" {

host = "npipe:////.//pipe//docker\_engine"

}

resource "docker\_container" "mysql\_container" {

name = "${var.container\_name}"

image = "${docker\_image.mysql\_image.latest}"

env = [

"MYSQL\_ROOT\_PASSWORD = ${var.mysql\_root\_password}"

]

volumes {

volume\_name = "${docker\_volume.mysql\_data\_volume.name}"

container\_path = "/var/lib/mysql"

}

networks\_advanced {

name = docker\_network.private\_bridge\_network.name

}

}

***Assignment 6***

**Using Terraform to Create a RandomID and S3 Buckets**

**Create the Main file:-**

**Create the main.tf Terraform file.**

**Add a provider, aws.**

**Set the region to use a variable called aws\_region.**

**Add a random\_id resource and name it tf\_bucket\_id.**

**Set the byte\_length to 2.**

**Add a resource, aws\_s3\_bucket, and name it tf\_code.**

**The bucket name will be set using a variable called project\_name, followed by a -, and will use the dec attribute from tf\_bucket\_id.**

**Set the acl to private.**

**Set force\_destroy to true.**

**Create a tag with a name to tf\_bucket.**

**Create the Variables File:-**

**Create the variables.tf Terraform file.**

**Add a variable called aws\_region.**

**Set the default to us-east-1. Add a variable called project\_name.**

**Set the default to la-terraform.**

**Create the outputs file:-**

**Create the outputs.tf Terraform file.**

**Add a output called bucketname.**

**The value should be set to id, coming from tf\_code.**

**Deploy the infrastructure:-**

**Initialize Terraform.**

**Validate the files.**

**Deploy the S3 bucket.**

**Main.tf**

provider "aws" {

  access\_key = var.aws\_access\_key

  secret\_key = var.aws\_secret\_key

  region = var.aws\_region

}

resource "random\_id" "tf\_bucket\_id" {

  byte\_length = 2

  count = var.number\_of\_instances

}

resource "aws\_s3\_bucket" "tf\_code" {

    #count = 2

    bucket        = "${var.project\_name}-${random\_id.tf\_bucket\_id[count.index]}"

    acl           = "private"

    force\_destroy =  true

    tags = {

      Name = "tf\_bucket-${count.index}"

    }

}

**Variable.t**f

variable "aws\_access\_key" { default = ""}

variable "aws\_secret\_key" {default = ""}

variable "key\_name" {default ="MyFirstKey"}

variable "private\_key\_path" {default = "D:\\AWS\\MyFirstKey.pem"}

variable "aws\_region" {

  default = "us-east-1"

}

variable "project\_name" {

  default = "la-terraform"

}

variable "number\_of\_instances" {

  default = 2

}

**Output.tf**

output "bucketname" {

  value = join(",", [aws\_s3\_bucket.tf\_code.id, aws\_s3\_bucket.tf\_code.id])

}

***Assignment 7***

**Using Join and Count to Create Multiple S3 Buckets**

**Find three files as below:-**

**main.tf :-**

**---------**

**provider "aws" {**

**region = "${var.aws\_region}"**

**}**

**resource "random\_id" "tf\_bucket\_id" {**

**byte\_length = 2**

**}**

**resource "aws\_s3\_bucket" "tf\_code" {**

**bucket = "${var.project\_name}-${random\_id.tf\_bucket\_id.dec}"**

**acl = "private"**

**force\_destroy = true**

**tags {**

**Name = "tf\_bucket"**

**}**

**}**

**variables.tf**

**variable "aws\_region" {**

**default = "us-east-1"**

**}**

**variable "project\_name" {**

**default = "la-terraform"**

**}**

**outputs.tf:-**

**-------------**

**output "bucketname" {**

**value = "${aws\_s3\_bucket.tf\_code.id}"**

**}**

**Update the Variables File:-**

**Edit variables.tf.**

**Add a new variable number\_of\_instances.**

**Set the the default to 2.**

**Update the Main File:-**

**Update random\_id and add a count.**

**Set the value count to use the number\_of\_instances variable.**

**Update aws\_s3\_bucket and add a count.**

**Update random\_id.tf\_bucket\_id.dec so it iterates through the count. Update the Name tag so that tf\_bucket is appended with the count index plus one.**

**Update the Outputs File:-**

**Update the bucketname output value to use the join function so that it returns a comma delimited list of bucket names.**

**Deploy the Infrastructure:-**

**Initialize Terraform.**

**Validate the files.**

**Deploy the S3 buckets.**

**main.tf**

provider "aws" {

access\_key = var.aws\_access\_key

secret\_key = var.aws\_secret\_key

region = var.aws\_region

}

resource "random\_id" "tf\_bucket\_id" {

byte\_length = 2

count = "${var.number\_of\_instances}"

}

resource "aws\_s3\_bucket" "tf\_code" {

count = "${var.number\_of\_instances}"

bucket = "${var.project\_name}-${random\_id.tf\_bucket\_id.\*.dec[count.index]}"

acl = "private"

force\_destroy = true

# tags = {

# Name = "tf\_bucket-${count.index}"

# }

tags = {

Name = "${format("tf\_bucket-%d", count.index + 1)}"

}

}

**Variables.tf**

variable "aws\_access\_key" { default = ""}

variable "aws\_secret\_key" {default = ""}

variable "key\_name" {default ="MyFirstKey"}

variable "private\_key\_path" {default = "D:\\AWS\\MyFirstKey.pem"}

variable "aws\_region" {

default = "us-east-1"

}

variable "project\_name" {

default = "la-terraform"

}

variable "number\_of\_instances" {

default = 2

}

**Output.tf**

output "bucketname" {

value = "${join(", ", aws\_s3\_bucket.tf\_code.\*.id)}"

}