# vars.tf

# Input Variables https://www.terraform.io/language/values/variables

variable "aws\_access\_key" {

}

variable "aws\_secret\_key" {

}

variable "apac\_region" {

default = "ap-south-1"

}

variable "cidr\_blocks" {

default = "0.0.0.0/0"

}

#Network Mask - 255.255.255.0 Addresses Available - 256

variable "vpc\_cidr" {

default = "10.0.1.0/24"

}

variable "public\_cidr" {

default = "10.0.1.0/28"

}

variable "private\_cidr" {

default = "10.0.1.16/28"

}

variable "instance\_type" {

default = "t2.micro"

}

\*\*

provider "aws" {

region = "ap-south-1"

}

#Provides an EC2 instance resource.

resource "aws\_instance" "provisionerTestVM" {

ami = "ami-0c6615d1e95c98aca"

instance\_type = "t2.micro"

}

Execute the terraform plan command to verify the execution plan.

\*\* add provisioner block in resource

provider "aws" {

region = "ap-south-1"

}

#Provides an EC2 instance resource.

resource "aws\_instance" "provisionerTestVM" {

ami = "ami-0c6615d1e95c98aca"

instance\_type = "t2.micro"

provisioner "local-exec" {

command = "echo Instance Type=${self.instance\_type}, Instance ID=${self.id}, Public DNS=${self.public\_dns}, AMI ID=${self.ami} >> allinstancedetails"

}

}

Execute the terraform apply command to create an instance in AWS.

\*\* Install packer using chocolatey by referring to https://community.chocolatey.org/packages/packer/1.2.2:

choco install packer

\*\* The following is a script to create an AMI in Amazon EC2 using Packer

{

"variables": {

"aws\_access\_key": "",

"aws\_secret\_key": ""

},

"builders": [

{

"type": "amazon-ebs",

"access\_key": "{{user `aws\_access\_key`}}",

"secret\_key": "{{user `aws\_secret\_key`}}",

"region": "ap-south-1",

"source\_ami": "ami-0851b76e8b1bce90b",

"instance\_type": "t2.micro",

"ssh\_username": "ubuntu",

"ami\_name": "packer-cf-ami-{{timestamp}}"

}

],

"provisioners": [

]

}

Save the file as firstawsami-packer.json.

Execute following packer command to create AMI:

packer build -var "aws\_access\_key=XXXXXXXXXXXXXXXXXXXX" -var "aws\_secret\_key=XXXXXXXXXXXXXXXXXXXX" firstawsami-packer.json

\*\* Let’s configure terraform script in a way that AMI created using Packer is considered based on the latest creation time.

provider "aws" {

region = "ap-south-1"

}

#Get latest AMI ID based on Filter - Here AMI created using Packer

data "aws\_ami" "packeramis" {

owners = ["10xxxxxxxxxx"] #change the owner ID as per your account

most\_recent = true

filter {

name = "name"

values = ["packer-cf\*"]

}

}

#Provides an EC2 instance resource.

resource "aws\_instance" "provisionerTestVM" {

ami = data.aws\_ami.packeramis.id

instance\_type = "t2.micro"

provisioner "local-exec" {

command

= "echo Instance Type=${self.instance\_type}, Instance ID=${self.id}, Public DNS=${self.public\_dns}, AMI ID=${self.ami} >> allinstancedetails"

}

}

\*\* Let’s consider a scenario where we would like to install Nginx using the remote-exec block.

# Configure region in provider block using variable

provider "aws" {

region = var.apac\_region

}

# Query all avilable Availability Zone; we will use specific availability zone using index

data "aws\_availability\_zones" "available" {}

# VPC Creation using cidr block available in vars.tf

resource "aws\_vpc" "provisionerVPC" {

cidr\_block = var.vpc\_cidr

enable\_dns\_hostnames = true

enable\_dns\_support = true

tags = {

Name = "dev-terraform-vpc"

}

}

# Public Subnet public cidr block available in vars.tf and provisionerVPC

resource "aws\_subnet" "public\_subnet" {

cidr\_block = var.public\_cidr

vpc\_id = aws\_vpc.provisionerVPC.id

map\_public\_ip\_on\_launch = true

availability\_zone = data.aws\_availability\_zones.available.names[1]

tags = {

Name = "dev-public-subnet"

}

}

#To access EC2 instance inside a Virtual Private Cloud (VPC) we need an Internet Gateway and a routing table connecting the subnet to the Internet Gateway

# Creating Internet Gateway

resource "aws\_internet\_gateway" "gw" {

vpc\_id = aws\_vpc.provisionerVPC.id

tags = {

Name = "dev-gw"

}

}

# Public Route Table

resource "aws\_route\_table" "public\_route" {

vpc\_id = aws\_vpc.provisionerVPC.id

route {

cidr\_block = var.cidr\_blocks

gateway\_id = aws\_internet\_gateway.gw.id

}

tags = {

Name = "dev-public-route"

}

}

# Associate Public Subnet with Public Route Table

resource "aws\_route\_table\_association" "public\_subnet\_assoc" {

route\_table\_id = aws\_route\_table.public\_route.id

subnet\_id = aws\_subnet.public\_subnet.id

depends\_on = [aws\_route\_table.public\_route, aws\_subnet.public\_subnet]

}

# Security Group Creation for provisionerVPC

resource "aws\_security\_group" "dev\_terraform\_sg\_allow\_ssh\_http" {

name = "dev-sg"

vpc\_id = aws\_vpc.provisionerVPC.id

}

# Ingress Security Port 22 (Inbound)

resource "aws\_security\_group\_rule" "ssh\_ingress\_access" {

from\_port = 22

protocol = "tcp"

security\_group\_id = aws\_security\_group.dev\_terraform\_sg\_allow\_ssh\_http .id

to\_port = 22

type = "ingress"

cidr\_blocks = [var.cidr\_blocks]

}

# Ingress Security Port 80 (Inbound)

resource "aws\_security\_group\_rule" "http\_ingress\_access" {

from\_port = 80

protocol = "tcp"

security\_group\_id = aws\_security\_group.dev\_terraform\_sg\_allow\_ssh\_http .id

to\_port = 80

type = "ingress"

cidr\_blocks = [var.cidr\_blocks]

}

# All egress/outbound Access

resource "aws\_security\_group\_rule" "all\_egress\_access" {

from\_port = 0

protocol = "-1"

security\_group\_id = aws\_security\_group.dev\_terraform\_sg\_allow\_ssh\_http .id

to\_port = 0

type = "egress"

cidr\_blocks = [var.cidr\_blocks]

}

# Instance Configuration

resource "aws\_instance" "provisioner-remoteVM" {

ami = "ami-0c6615d1e95c98aca"

instance\_type = var.instance\_type

key\_name = "terraform"

vpc\_security\_group\_ids = [aws\_security\_group.dev\_terraform\_sg\_allow\_ssh\_http.id]

subnet\_id = aws\_subnet.public\_subnet.id

tags = {

Name = "remote-instance"

}

provisioner "remote-exec" {

inline = [

"sudo yum update -y",

"sudo amazon-linux-extras install -y nginx1",

"sudo service nginx start"

]

}

connection {

type = "ssh"

host = aws\_instance.provisioner-remoteVM.public\_ip

user = "ec2-user"

password = ""

private\_key = file("${path.module}/terraform.pem")

}

}