Terraform with AWS

. Terraform is an orchestration tool, it is used to provision AWS resources through command line

. Terraform is agnostic which means it can be used to provision AWS/Azure/GCP Cloud resources

. Terraform uses the access key and secret key in order to provision the resources through CLI

. Cloudformation can be used only with AWS, as mentioned earlier Terraform can be used with AWS/Azure/GCP as well.

. Terraform is easy to write declarative templates in HCL (Hashi corp configuration Language)

. Terraform includes an array of modules, built in functions which can be used in Cloud and Onprem as well

. Before stepping into Terraform we need to be familiar with AWS or Azure or GCP and Cloud CLI

. We need to set up Terraform Binary in our OS to proceed with Terraform installation.

. The methods to install Terraform Binary for various OS is given in this link – Downloads | Terraform by HashiCorp

. In my case it is Ubuntu Server OS so I will follow the below steps to install Terraform Binary in my OS.

The below commands needs to be executed as a Ubuntu/Linux user in home directory

```
sudo apt-get update -y


sudo apt-get install wget unzip -y
```

sudo wget https://releases.hashicorp.com/terraform/1.1.7/terraform_1.1.7_linux_amd64.zip

```
sudo unzip terraform_1.1.7_linux_amd64.zip
```

.

```
sudo mv terraform /usr/local/bin/
```

```
shiva@hypo-cloudeva:/usr/local/bin$ ls
docker-machine-driver-kvm  flask  kubectl  minikube  terraform  v1.25.1
shiva@hypo-cloudeva:/usr/local/bin$ terraform -v
Terraform v1.1.7
on linux_amd64
shiva@hypo-cloudeva:/usr/local/bin$
```

We have successfully installed Terraform latest version is Ubuntu Server.

Next step is to install AWS – CLI

sudo apt-get install python3-pip (this command installs the pip manager) – in Linux

sudo pip3 install awscli –user (this command installs the aws CLI in your system) – in Linux

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

The above steps to be done in Ubuntu

To check the aws version … type aws –version

```
shiva@hypo-cloudeva:~$ aws --version
aws-cli/2.4.27 Python/3.8.8 Linux/5.13.0-35-generic exe/x86_64.ubuntu.20 prompt/
off
shiva@hypo-cloudeva:~$
```

We need to configure the access key and secret key of IAM administrator user in this System

. type aws configure

Prior to the above step… create an IAM user with administrator privileges and download the credentials… Ie Access key and Secret key

When you type aws configure … it will ask for your access key and secret key

Copy paste the access key and secret key in that field

Now create a empty directory in your system as below

```
shiva@hypo-cloudeva:~$ mkdir aravind_tfproject
shiva@hypo-cloudeva:~$ cd aravind_tfproject/
shiva@hypo-cloudeva:~/aravind_tfproject$ terraform init
Terraform initialized in an empty directory!

The directory has no Terraform configuration files. You may begin working
with Terraform immediately by creating Terraform configuration files.
shiva@hypo-cloudeva:~/aravind_tfproject$
```

Let’s create a s3 bucket using terraform

```
.       aws s3api create-bucket --bucket yourbucketnamehere --region ap-south-1 -
-create-bucket-configuration LocationConstraint=ap-south-1
```

Let’s see how can we create a VPC , two public subnets and an internet gateway in AWS through terraform

Inside the directory you created … create a file with .tf extension

shiva@hypo-cloudeva:~/aravind_tfproject$ sudo nano aravindterraformnetwork.tf

Type the below templates according to your requirement inside the file

```
#to create a vpc
resource "aws_vpc" "terraformshivavpc" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "terraformshivavpc"
  }
}



#to create a public subnet1

resource "aws_subnet" "public" {
  vpc_id     = aws_vpc.terraformshivavpc.id
  cidr_block = "10.0.2.0/24"
  availability_zone = "ap-south-1a"
}

#to create a public subnet2

resource "aws_subnet" "private" {
```

```
  vpc_id = aws_vpc.terraformshivavpc.id
  cidr_block = "10.0.3.0/24"
  availability_zone = "ap-south-1b"
}


# to create a internet gateway

resource "aws_internet_gateway" "terraformawsgateway" {
 vpc_id = aws_vpc.terraformshivavpc.id
}

# to create a route table for - IGW

resource "aws_route_table" "my_table" {
 vpc_id = aws_vpc.terraformshivavpc.id


 route {
  cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.terraformawsgateway.id
 }
```

Save the file and exit

Type terraform plan and it throws error if any in your template else it displays the resources to be provisioned… PFB

```
shiva@hypo-cloudeva:~/aravind_tfproject$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_internet_gateway.terraformawsgateway will be created
  + resource "aws_internet_gateway" "terraformawsgateway" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags_all = (known after apply)
      + vpc_id   = (known after apply)
    }

  # aws_route_table.my_table will be created
  + resource "aws_route_table" "my_table" {
      + arn              = (known after apply)
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + route            = [
          + {
              + carrier_gateway_id        = ""
              + cidr_block                = "0.0.0.0/0"
              + destination_prefix_list_id = ""
              + egress_only_gateway_id    = ""
              + gateway_id                = (known after apply)
              + instance_id               = ""
              + ipv6_cidr_block           = ""
              + local_gateway_id          = ""
              + nat_gateway_id            = ""
              + network_interface_id      = ""
              + transit_gateway_id        = ""
              + vpc_endpoint_id           = ""
              + vpc_peering_connection_id  = ""
            },
        ]
      + tags_all         = (known after apply)
      + vpc_id           = (known after apply)
    }
```

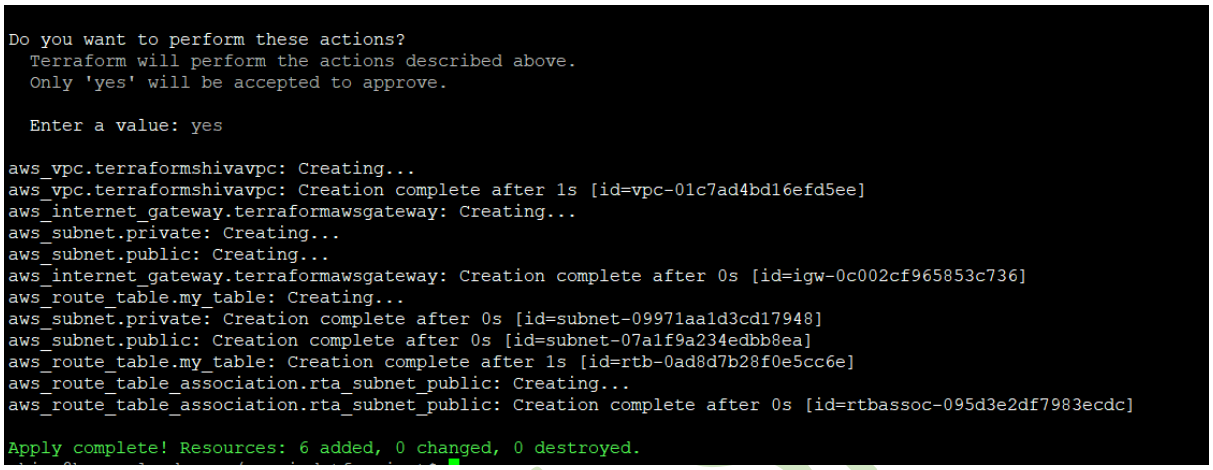
```
  # aws_route_table_association.rta_subnet_public will be created
  + resource "aws_route_table_association" "rta_subnet_public" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = (known after apply)
    }

  # aws_subnet.private will be created
  + resource "aws_subnet" "private" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "ap-south-1b"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.3.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags_all                                       = (known after apply)
      + vpc_id                                         = (known after apply)
    }

  # aws_subnet.public will be created
  + resource "aws_subnet" "public" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "ap-south-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.2.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags_all                                       = (known after apply)
      + vpc_id                                         = (known after apply)
    }
```

As the terraform plan shows the resources to be provisioned in AWS its good to

use the command

. Terraform apply

and provision the plan in AWS console



Happy Learning more depth of terraform commands with usage to be continued in my medium blog soon - [Aravind KumarTS – Medium]