

ASSIGNMENT 1

Write Terraform script to do perform following tasks on AWS cloud Platform

First Terraform is intalled and configured in our machine.

The successful installation of terraform is checked using the ``terraform version`` command.

```
ishita@ishita-VirtualBox:~/project-terraform$ sudo mv terraform_0.13.5_linux_amd64 /usr/local/bin  
[sudo] password for ishita:  
ishita@ishita-VirtualBox:~/project-terraform$ terraform version  
Terraform v0.13.5  
ishita@ishita-VirtualBox:~/project-terraform$
```

```
see "snap info <snapname>" for additional versions.  
  
ishita@ishita-VirtualBox:~/terraform$ terraform init  
  
Initializing the backend...  
  
Initializing provider plugins...  
- Finding latest version of hashicorp/aws...  
- Installing hashicorp/aws v3.16.0...  
- Installed hashicorp/aws v3.16.0 (signed by HashiCorp)  
  
The following providers do not have any version constraints in configuration,  
so the latest version was installed.  
  
To prevent automatic upgrades to new major versions that may contain breaking  
changes, we recommend adding version constraints in a required_providers block  
in your configuration, with the constraint strings suggested below.  
  
* hashicorp/aws: version = "~> 3.16.0"  
  
Terraform has been successfully initialized!  
  
You may now begin working with Terraform. Try running "terraform plan" to see  
any changes that are required for your infrastructure. All Terraform commands  
should now work.  
  
If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.  
ishita@ishita-VirtualBox:~/terraform$
```

Then terraform is configured to aws by adding the necessary files and access key id and password in the terraform configuration file, `config.tf`.

```
commands will detect it and remind you to do so if necessary.  
ishita@ishita-VirtualBox:~/terraform$ cat config.tf  
# main.tf  
provider "aws" {  
  aws_access_key_id=ASIARIKMM3ZDU4UUHAPC  
  aws_secret_access_key=odMtQtJNJoUISYX0aLzOenTIyqnyb0nU0cyyZN53  
}  
ishita@ishita-VirtualBox:~/terraform$
```

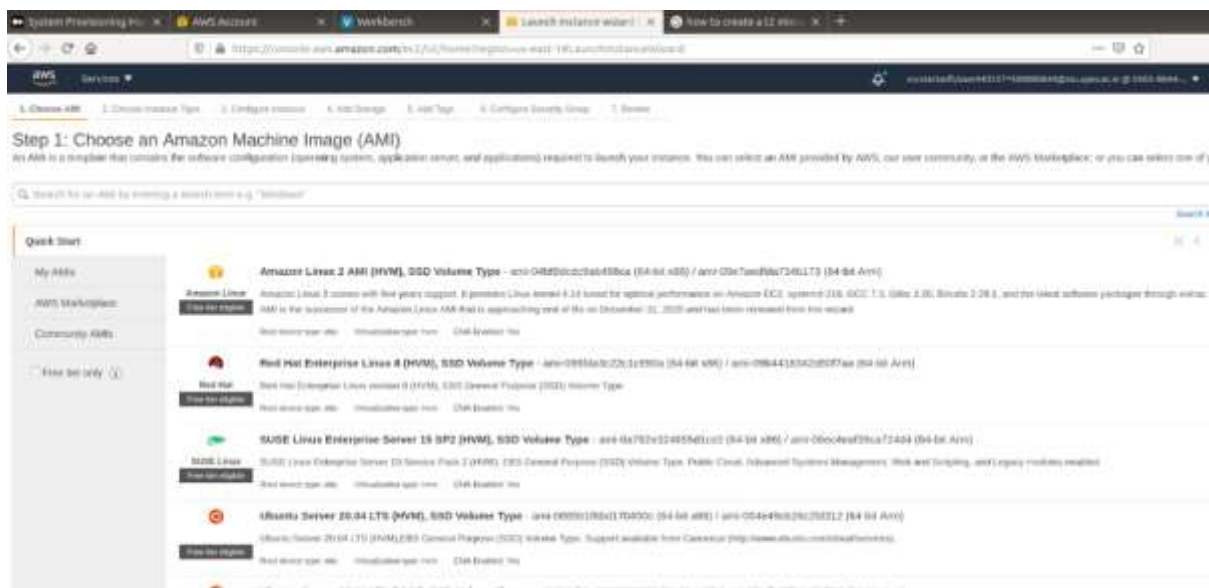
Step 1: Create two T2 Micro EC2 Instances.

Terraform script for creating two EC2 instances:

A screenshot of a code editor window titled "Instances.tf" with a subtitle "~/.terraform". The editor shows a Terraform script with two provider blocks and two resource blocks. The first provider block is for the "aws" provider in the "ap-south-1" region. The second provider block is for the "aws" provider in the "us-east-1" region with an alias of "usa". The first resource block is for an "aws_instance" named "first-instance-india" using AMI "ami-0e306788ff2473ccb" and instance type "t2.micro". The second resource block is for an "aws_instance" named "second-instance-USA" using AMI "ami-0947d2ba12ee1ff75", instance type "t2.micro", and provider "aws.usa".

```
provider "aws" {  
  region = "ap-south-1"  
}  
  
provider "aws" {  
  region = "us-east-1"  
  alias = "usa"  
}  
  
resource "aws_instance" "first-instance-india" {  
  ami = "ami-0e306788ff2473ccb"  
  instance_type = "t2.micro"  
}  
  
resource "aws_instance" "second-instance-USA" {  
  ami = "ami-0947d2ba12ee1ff75"  
  instance_type = "t2.micro"  
  provider = aws.usa  
}
```

An AMI is chosen from the AWS management console.



Instances created.



Step2: Create a VPN on AWS

Terraform script for creating VPN on AWS.

```
shita@ishita-VirtualBox:~/terraform$ cat vpn.tf
resource "aws_vpc" "vpc" {
  cidr_block = "10.0.0.0/16"

}

resource "aws_vpn_gateway" "vpn_gateway" {
  vpc_id = aws_vpc.vpc.id

}

resource "aws_customer_gateway" "customer_gateway" {
  bgp_asn = 65000
  ip_address = "172.0.0.1"
  type = "ipsec.1"

}

resource "aws_vpn_connection" "main" {
  vpn_gateway_id = aws_vpn_gateway.vpn_gateway.id
  customer_gateway_id = aws_customer_gateway.customer_gateway.id
  type = "ipsec.1"
  static_routes_only = true

}
```

VPN by the name ishita-vpc created.



The screenshot shows the AWS Management Console 'Your VPCs' page. It lists two VPCs: a default VPC and a custom VPC named 'ishita-vpc'. The 'ishita-vpc' VPC is highlighted with an orange background. The table columns are Name, VPC ID, State, IPv4 CIDR, IPv6 CIDR (Network border group), and IPv6 pool.

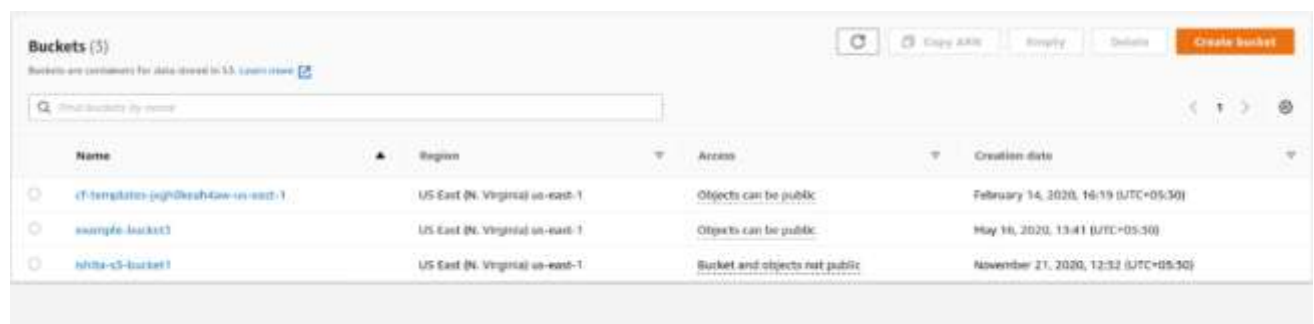
Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR (Network border group)	IPv6 pool
--	vpc-6384d1f9	Available	172.31.0.0/16	--	--
ishita-vpc	vpc-0be73dd0f4ade5c157	Available	10.0.0.0/16	--	--

Step 3: Create a S3 Bucket

Terraform script for creating an S3 bucket.

```
ishita@ishita-VirtualBox:~/terraform$ cat s3.tf
resource "aws_s3_bucket" "ishita-s3-bucket" {
  bucket = "ishita-s3-bucket1234"
  acl    = "public-read"
  tags = {
    Name = "ishita-s3-bucket1"
  }
  versioning {
    enabled = true
  }
}
```

S3 bucket by the name ishita-s3-bucket1 created.



The screenshot shows the AWS Management Console 'Buckets' page. At the top, there are buttons for 'Refresh', 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. Below the buttons is a search bar and a table of buckets. The table has columns for Name, Region, Access, and Creation date. Three buckets are listed: 'cf-templates-jgph0ksh4aw-us-east-1', 'example-bucket', and 'ishita-s3-bucket1'. The 'ishita-s3-bucket1' bucket is highlighted, showing its access level as 'Bucket and objects not public'.

Name	Region	Access	Creation date
cf-templates-jgph0ksh4aw-us-east-1	US East (N. Virginia) us-east-1	Objects can be public	February 14, 2020, 16:19 (UTC+05:30)
example-bucket	US East (N. Virginia) us-east-1	Objects can be public	May 16, 2020, 13:41 (UTC+05:30)
ishita-s3-bucket1	US East (N. Virginia) us-east-1	Bucket and objects not public	November 21, 2020, 12:52 (UTC+05:30)

Step 4: Terraform commands to run the terraform files:

- Terraform init: This command initializes the working directory which contains all the terraform configuration files.

```
ishita@ishita-VirtualBox:~/terraform$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.16.0...
- Installed hashicorp/aws v3.16.0 (signed by HashiCorp)

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* hashicorp/aws: version = "~> 3.16.0"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

- Terraform plan: This command is used to create an execution plan.
- Terraform apply: This command applies the changes which were required to perform the task assigned.

```
ishita@ishita-VirtualBox:~/terraform$ terraform apply

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
ishita@ishita-VirtualBox:~/terraform$ gedit config.tf
```

- Terraform validate: This validates the configuration file in a directory.