

# ASSIGNMENT 1

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

**Step 1: Create two T2 Micro EC2 Instances.**

**Step2: Create a VPN on AWS**

**Step 3: Create a S3 Bucket**

**Step 4: Write the code for step 1,2 and 3 in a IaC terraform file and run terraform commands to execute these steps.**

- Codes for step 1, step 2 and step 3 :** These codes will create all the three AWS entities in one go.....

```
provider "aws" {
  region      = "ap-south-1"
  access_key  = "AKIAWJZR2MO2CWOZHTIA"
  secret_key  = "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"
}

# Step 1: Create 2 EC2 Instances
resource "aws_instance" "web" {
  count        = 2
  ami          = "ami-0c02fb55956c7d316" # Amazon Linux 2 AMI (us-east-1)
  instance_type = "t2.micro"

  tags = {
    Name = "EC2Instance-${count.index + 1}"
  }
}

# Step 2: Create a VPN Gateway and Customer Gateway Simulation
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main.id
}

resource "aws_customer_gateway" "cg" {
  bgp_asn    = 65000
  ip_address = "1.2.3.4" # Placeholder IP for on-premises
}
```

**Purpose:** Creates **2 t2.micro EC2 instances** using a basic Amazon Linux 2 AMI.

**count = 2** means it will create two instances.

Each instance gets a different name using \${count.index} (e.g., EC2Instance-1, EC2Instance-2).

The screenshot shows the Terraform Lab interface. The left sidebar is titled 'EXPLORER' and lists files: '.terraform', 'Assignment-1', 'main.tf', 'terraform.lock.hcl', 'terraform.tfstate', and 'terraform.tfstate.backup'. The right pane is titled 'Welcome' and shows the content of 'main.tf'. The code in 'main.tf' is as follows:

```
Assignment-1 > main.tf
23 }
24
25 resource "aws_internet_gateway" "igw" {
26   vpc_id = aws_vpc.main.id
27 }
28
29 resource "aws_customer_gateway" "cg" {
30   bgp_asn    = 65000
31   ip_address = "1.2.3.4" # Placeholder IP for on-premise
32   type       = "ipsec.1"
33 }
34
35 resource "aws_vpn_gateway" "vgw" {
36   vpc_id = aws_vpc.main.id
37 }
38
39 resource "aws_vpn_connection" "vpn" {
40   customer_gateway_id = aws_customer_gateway.cg.id
41   vpn_gateway_id      = aws_vpn_gateway.vgw.id
42   type                = "ipsec.1"
43 }
44
45 # Step 3: Create an S3 Bucket
46 resource "aws_s3_bucket" "bucket" {
47   bucket = "my-terraform-bucket-${random_id.bucket_suffix.hex}"
48   force_destroy = true
49 }
50
51 resource "random_id" "bucket_suffix" {
52   byte_length = 4
```

## EXPLANATION OF CODE –

### Cloud Step 1: EC2 Instances

hcl

CopyEdit

```
resource "aws_instance" "web" {
  count      = 2
  ami        = "ami-0c02fb55956c7d316"
  instance_type = "t2.micro"
```

tags = {

Name = "EC2Instance-\${count.index + 1}"

}

}

- **Purpose:** Creates **2 t2.micro EC2 instances** using a basic Amazon Linux 2 AMI.
  - **count = 2** means it will create two instances.
  - Each instance gets a different name using \${count.index} (e.g., EC2Instance-1, EC2Instance-2).
- 

## Step 2: VPN Setup

### Create a VPC (Virtual Private Cloud)

hcl

CopyEdit

```
resource "aws_vpc" "main" {  
    cidr_block = "10.0.0.0/16"  
}
```

- **Purpose:** A private network in AWS where your resources (like EC2, VPN) will live.

### Internet Gateway

h

CopyEdit

```
resource "aws_internet_gateway" "igw" {  
    vpc_id = aws_vpc.main.id  
}
```

- **Purpose:** Allows the VPC to connect to the internet.

### Customer Gateway (Simulated)

hcl

CopyEdit

```
resource "aws_customer_gateway" "cg" {  
    bgp_asn  = 65000  
    ip_address = "1.2.3.4" # Fake on-prem IP  
    type      = "ipsec.1"  
}
```

- **Purpose:** Represents your **on-premises network gateway**.
- This IP is usually from your office/home router but is simulated here.

## VPN Gateway

hcl

CopyEdit

```
resource "aws_vpn_gateway" "vgw" {  
    vpc_id = aws_vpc.main.id  
}
```

- **Purpose:** AWS-side VPN device to connect with the customer gateway.

## VPN Connection

hcl

CopyEdit

```
resource "aws_vpn_connection" "vpn" {  
    customer_gateway_id = aws_customer_gateway.cg.id  
    vpn_gateway_id     = aws_vpn_gateway.vgw.id  
    type              = "ipsec.1"  
}
```

- **Purpose:** Establishes the **VPN tunnel** between AWS and the customer gateway.
- 

### Step 3: S3 Bucket

hcl

CopyEdit

```
resource "random_id" "bucket_suffix" {  
    byte_length = 4  
}  
  
resource "aws_s3_bucket" "bucket" {  
    bucket = "my-terraform-bucket-${random_id.bucket_suffix.hex}"  
    force_destroy = true  
}
```

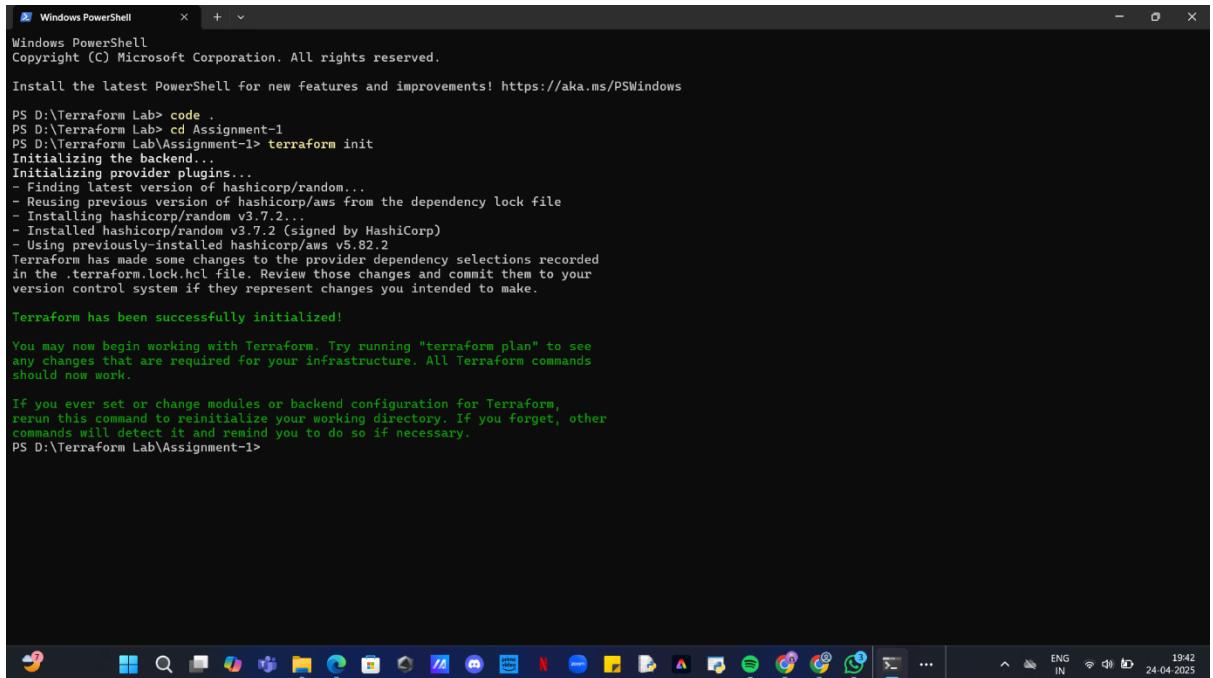
- **Purpose:** Creates a **unique S3 bucket** using a random 4-byte hex suffix.
  - `force_destroy = true` ensures the bucket can be deleted even if it has files.
- 

### Step 4: Running Terraform

- `terraform init` – Sets up Terraform and downloads plugins.
- `terraform plan` – Previews what will be created.
- `terraform apply` – Actually provisions the resources on AWS.

## Terraform commands to run :

- `terraform init` # Initialize Terraform



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

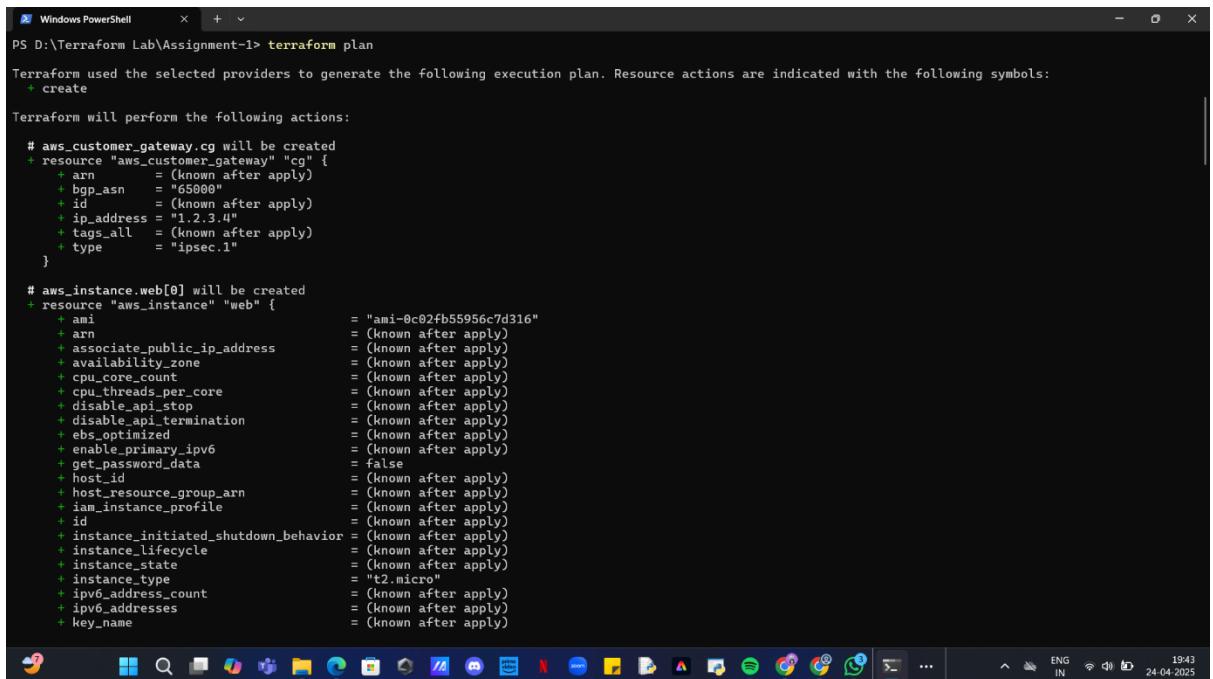
PS D:\Terraform Lab> code .
PS D:\Terraform Lab> cd Assignment-1
PS D:\Terraform Lab\Assignment-1> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/random...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Installing hashicorp/random v3.7.2...
- Installed hashicorp/random v3.7.2 (signed by HashiCorp)
- Using previously-installed hashicorp/aws v5.82.2
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

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.
PS D:\Terraform Lab\Assignment-1>
```

- `terraform plan` # Preview the changes



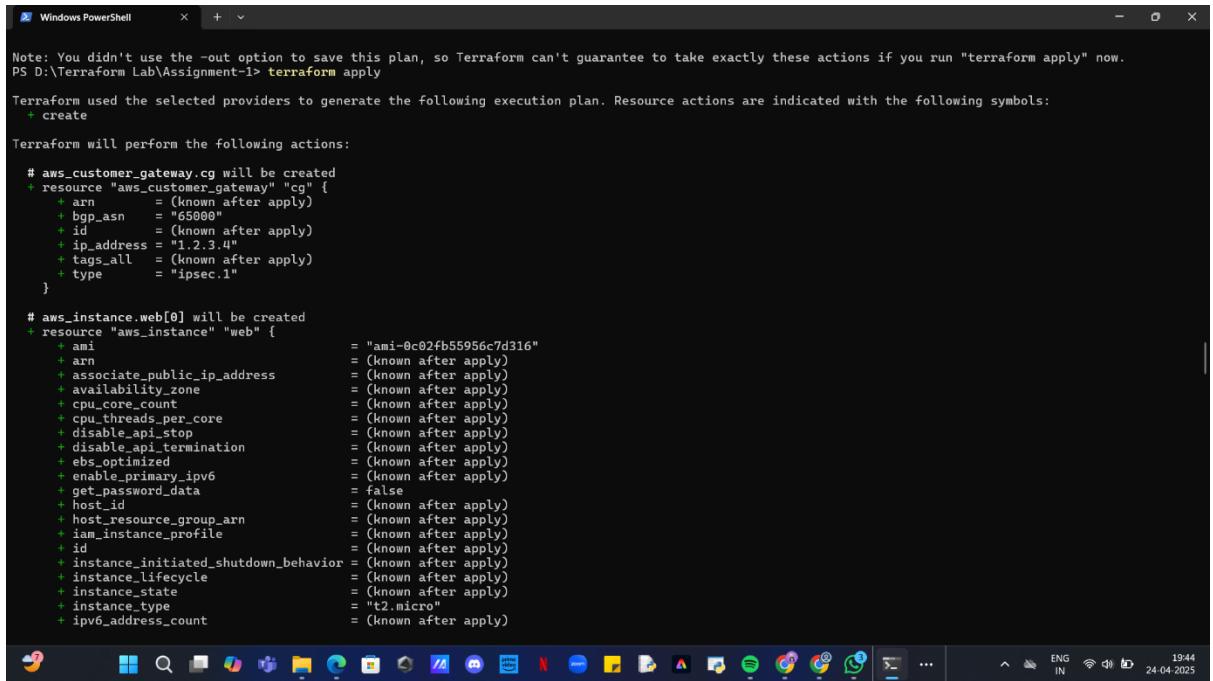
```
Windows PowerShell
PS D:\Terraform Lab\Assignment-1> terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_customer_gateway.cg will be created
+ resource "aws_customer_gateway" "cg" {
    + arn      = (known after apply)
    + bgp_asn  = "65000"
    + id       = (known after apply)
    + ip_address = "1.2.3.4"
    + tags_all = (known after apply)
    + type     = "ipsec.1"
}

# aws_instance.web[0] will be created
+ resource "aws_instance" "web" {
    + ami           = "ami-0c02fb55956c7d316"
    + arn          = (known after apply)
    + associate_public_ip_address = (known after apply)
    + availability_zone        = (known after apply)
    + cpu_core_count          = (known after apply)
    + cpu_threads_per_core    = (known after apply)
    + disable_api_stop         = (known after apply)
    + disable_api_termination = (known after apply)
    + ebs_optimized           = (known after apply)
    + enable_primary_ipv6      = (known after apply)
    + get_password_data        = false
    + host_id                 = (known after apply)
    + host_resource_group_arn = (known after apply)
    + iam_instance_profile     = (known after apply)
    + id                      = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance.lifecycle        = (known after apply)
    + instance.state           = (known after apply)
    + instance_type             = "t2.micro"
    + ipv6_address_count       = (known after apply)
    + ipv6_addresses            = (known after apply)
    + key_name                = (known after apply)
```

- **terraform apply # Apply and create resources**



```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS D:\Terraform Lab\Assignment-> terraform apply
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_customer_gateway.cg will be created
+ resource "aws_customer_gateway" "cg" {
    + arn          = (known after apply)
    + bgp_asn     = "65000"
    + id          = (known after apply)
    + ip_address   = "1.2.3.4"
    + tags_all    = (known after apply)
    + type         = "ipsec.1"
}

# aws_instance.web[0] will be created
+ resource "aws_instance" "web" {
    + ami           = "ami-0c02fb55956c7d316"
    + arn          = (known after apply)
    + associate_public_ip_address = (known after apply)
    + availability_zone      = (known after apply)
    + cpu_core_count        = (known after apply)
    + cpu_threads_per_core = (known after apply)
    + disable_api_stop      = (known after apply)
    + disable_api_termination = (known after apply)
    + ebs_optimized        = (known after apply)
    + enable_primary_ipv6   = (known after apply)
    + get_password_data     = false
    + host_id            = (known after apply)
    + host_resource_group_arn = (known after apply)
    + iam_instance_profile = (known after apply)
    + id                 = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance.lifecycle   = (known after apply)
    + instance.state       = (known after apply)
    + instance_type        = "t2.micro"
    + ipv6_address_count   = (known after apply)
```

**ALL THE RESOURCES SUCCESSFULLY CREATED**