SYSTEM PROVISION AND CONFIGURATION

ASSIGNMENT – 1

Submitted by:

Name	**Roll No**	**Branch**
Darsh Asawa	R171217016	CSE-DevOps

School of Computer Science and Engineering
University of Petroleum & Energy Studies

Dehradun - 248001

2020

TASK DESCRIPTION

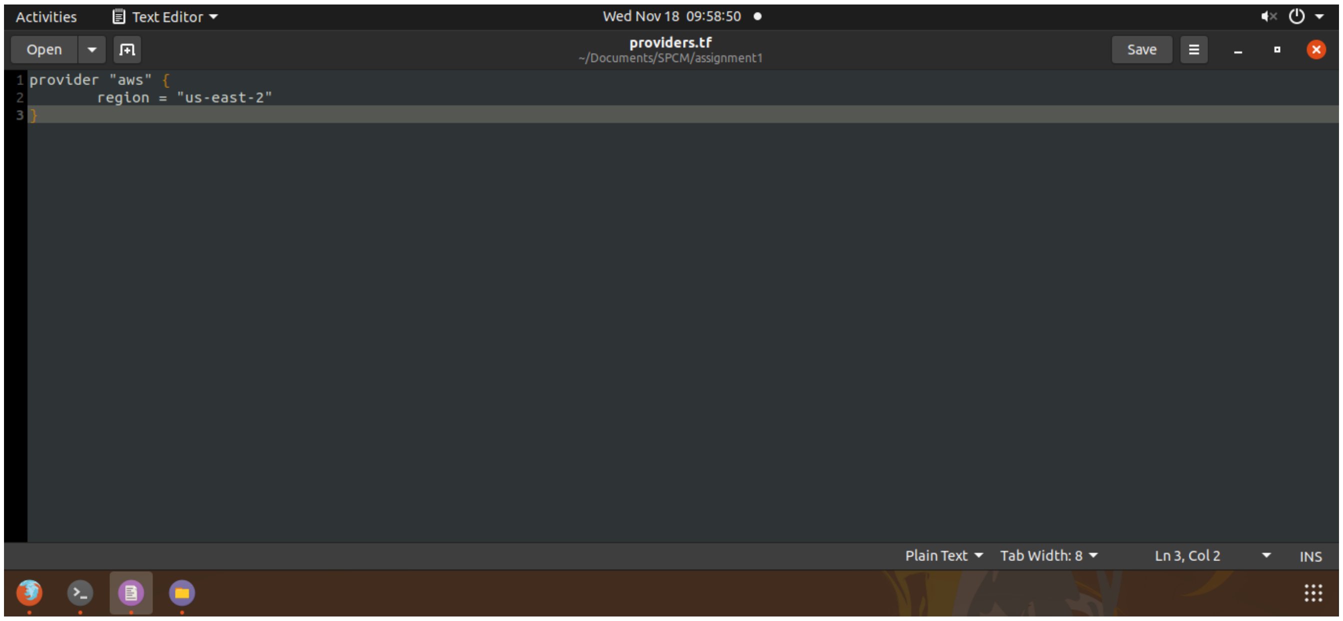
Write Terraform script to do perform following tasks on AWS cloud Platform
Step 1: Create two T2 Micro EC2 Instances.
Step2: Create a VPN on AWS
Step 3: Create a S3 Bucket
Step 4: Write the code for step 1,2 and 3 in a IaC terraform file and run terraform commands to execute these steps.

THEORY:

- **AWS EC2 Instance:** An **EC2 instance** is a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications on the Amazon Web Services (AWS) infrastructure. EC2 is a service that allows business subscribers to run application programs in the computing environment. Instances are created from Amazon Machine Images (AMI). The machine images are like templates that are configured with an operating system and other software, which determine the user's operating environment. Users can select an AMI provided by AWS, the user community, or through the AWS Marketplace. Users can also create their own AMIs and share them.
- **AWS VPN: AWS** Client **VPN** is a fully-managed, elastic **VPN** service that automatically scales up or down based on user demand. Because it is a cloud **VPN** solution, you don't need to install and manage hardware or software-based solutions or try to estimate how many remote users to support at one time.
- **AWS S3:** Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services that provides object storage through a web service interface. Amazon S3 uses the same scalable storage infrastructure that Amazon.com uses to run its global e-commerce network.

IMPLEMENTATION

1. Create a new directory and inside the directory, create a file named providers.tf. This file is used to configure the provider we are going to use (such as AWS).
   Additionally, providing the region in which we want the resources to be created.



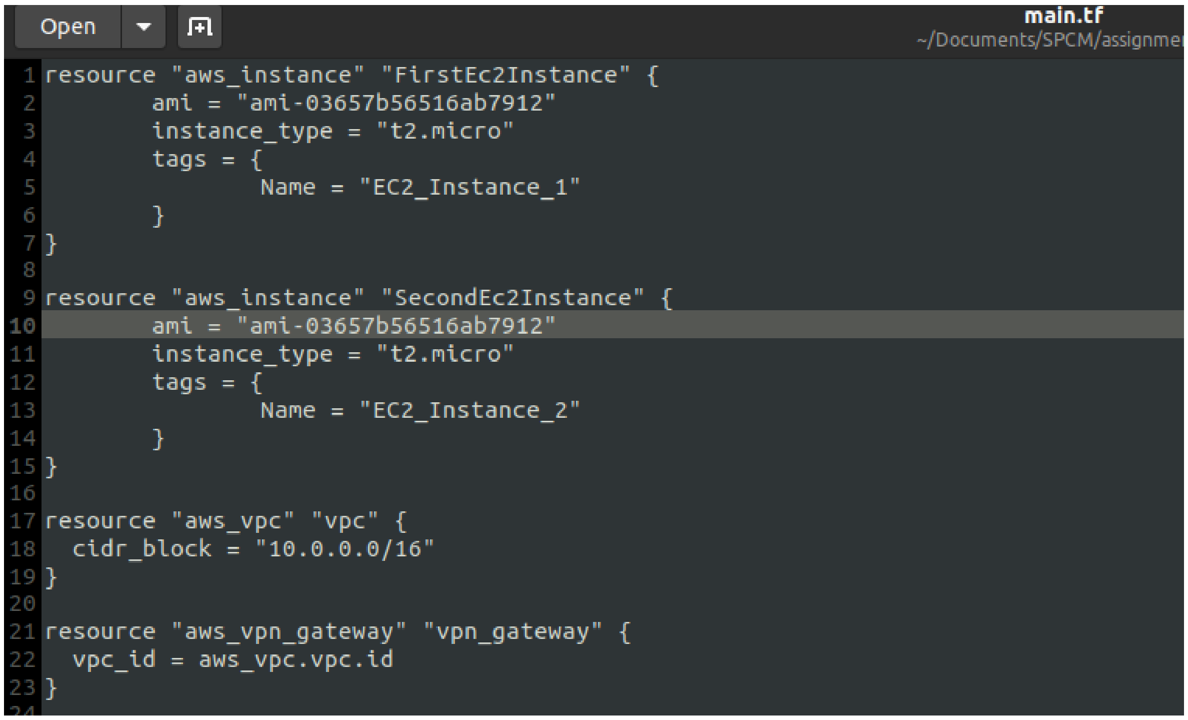
```
provider "aws" {
	region = "us-east-2"
}
```

2. Now we create main.tf file.
   Syntax for creating any resource in terraform is:
   resource "<PROVIDER>_<TYPE>" "NAME" {
   [Configuration…]
   }

   Provider can be aws, azure or gcp.
   Type is the type of resource to create in that provider, for say **Instance.**
   Configuration basically consists of one or more *arguments* that are specific to that resource.

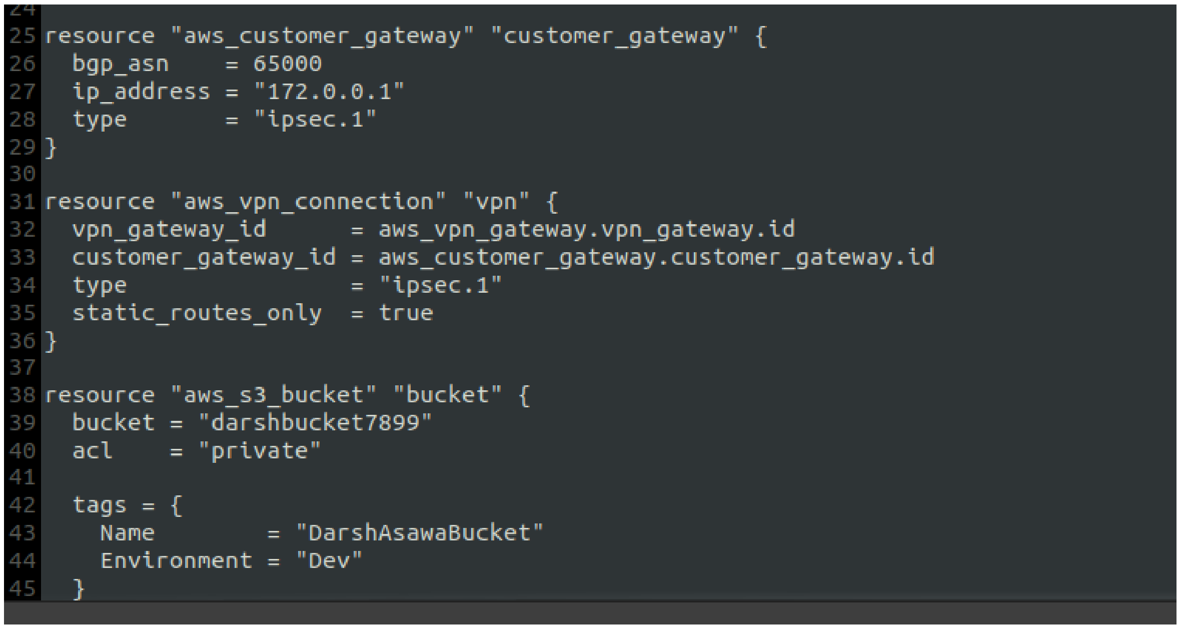
   In my main.tf, I've added the following resources:
   - Two **EC2 instances** with the names "EC2_Instance_1" and "EC2_Instance_2". I have used Amazon Linux 2 AMI and instance type as t2.micro as per the eligibility of free tier AWS account.
   - **VPN (Virtual Private Network)** – In order to create a VPN on AWS, we need to create a VPC (Virtual Private Cloud), a VPN gateway, a Customer Gateway and finally a VPN connection.
     So, started by adding resource VPC named "vpc" and added a CIDR block configuration.
     After that adding VPN Gateway resource named 'vpn_gateway", attaching to the VPC created above.

Followed by the creation of Customer Gateway named "customer_gateway". Giving it an IP address, its type as **ipsec.1**.
Finally, VPN connection resource to be established, naming it as "vpn" and attaching to the vpn gateway and customer gateway.



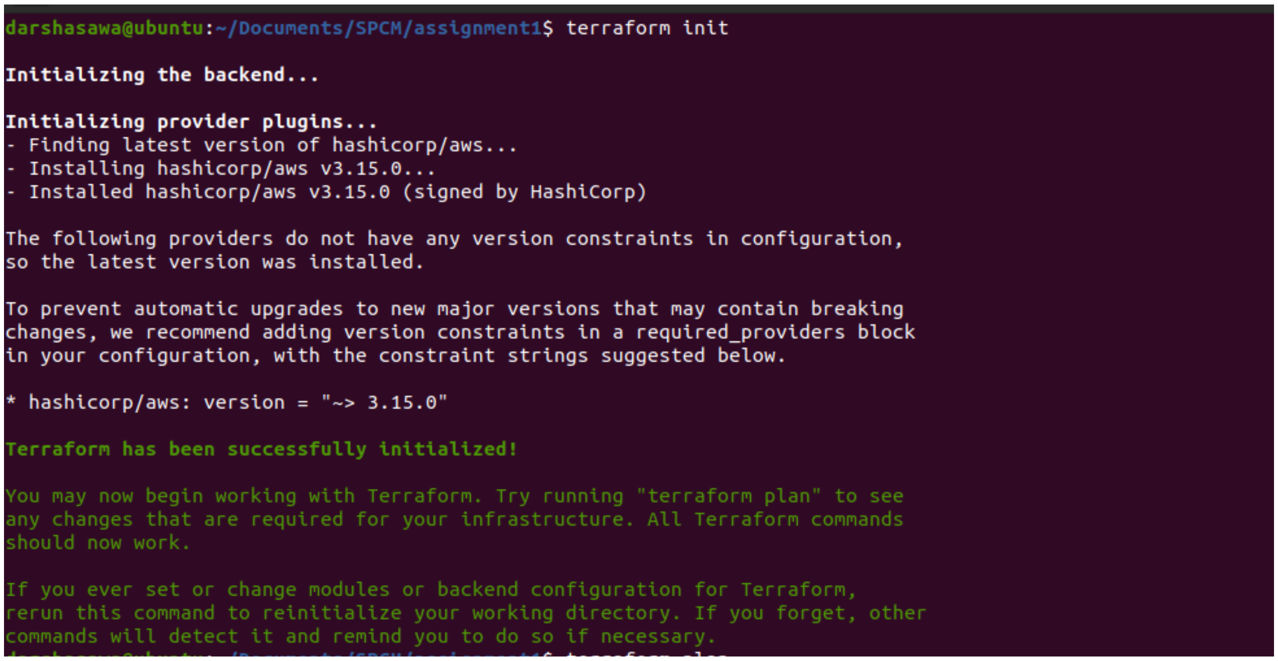
```
resource "aws_instance" "FirstEc2Instance" {
        ami = "ami-03657b56516ab7912"
        instance_type = "t2.micro"
        tags = {
                Name = "EC2_Instance_1"
        }
}

resource "aws_instance" "SecondEc2Instance" {
        ami = "ami-03657b56516ab7912"
        instance_type = "t2.micro"
        tags = {
                Name = "EC2_Instance_2"
        }
}

resource "aws_vpc" "vpc" {
  cidr_block = "10.0.0.0/16"
}

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

- **S3 Bucket:** Creating a private s3 bucket named "darshbucket7899" (As S3 bucket name have to be unique worldwide for sharing purposes).



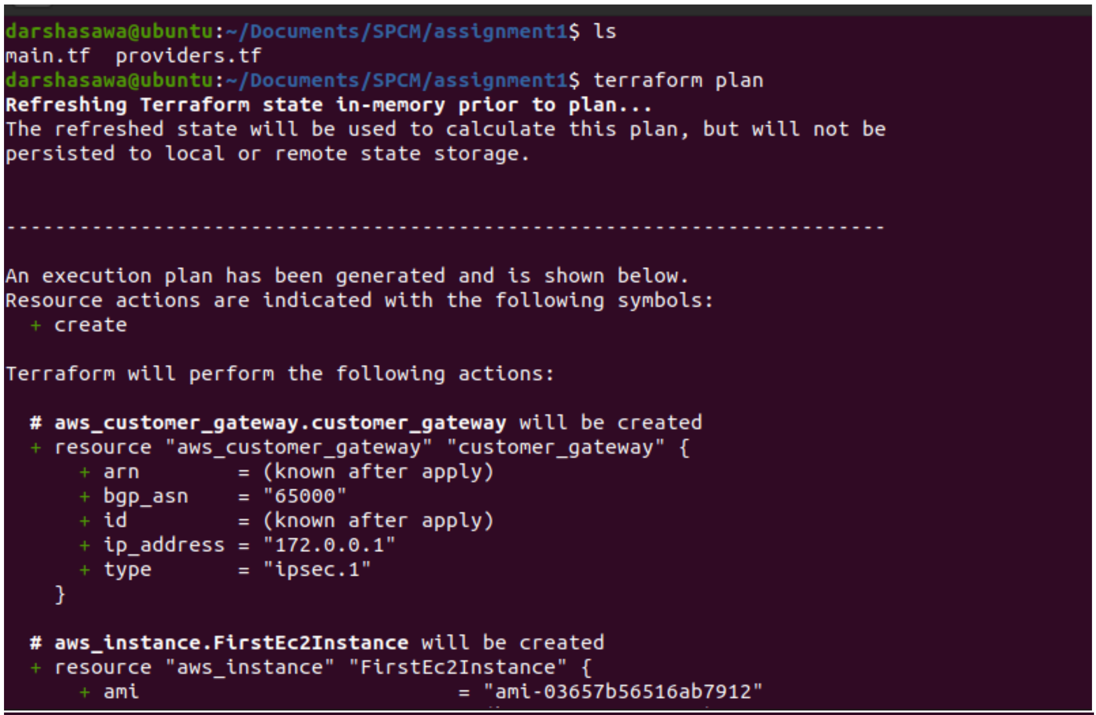
```
resource "aws_customer_gateway" "customer_gateway" {
  bgp_asn    = 65000
  ip_address = "172.0.0.1"
  type       = "ipsec.1"
}

resource "aws_vpn_connection" "vpn" {
  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
}

resource "aws_s3_bucket" "bucket" {
  bucket = "darshbucket7899"
  acl    = "private"

  tags = {
    Name        = "DarshAsawaBucket"
    Environment = "Dev"
  }
```

3. Open terminal in this folder, and execute ‘**terraform init**’.
   Terraform init will tell Terraform to scan the code, figure out which providers you’re using, and download the code for them. By default, the provider code will be downloaded into a .terraform folder, which is Terraform’s scratch directory.

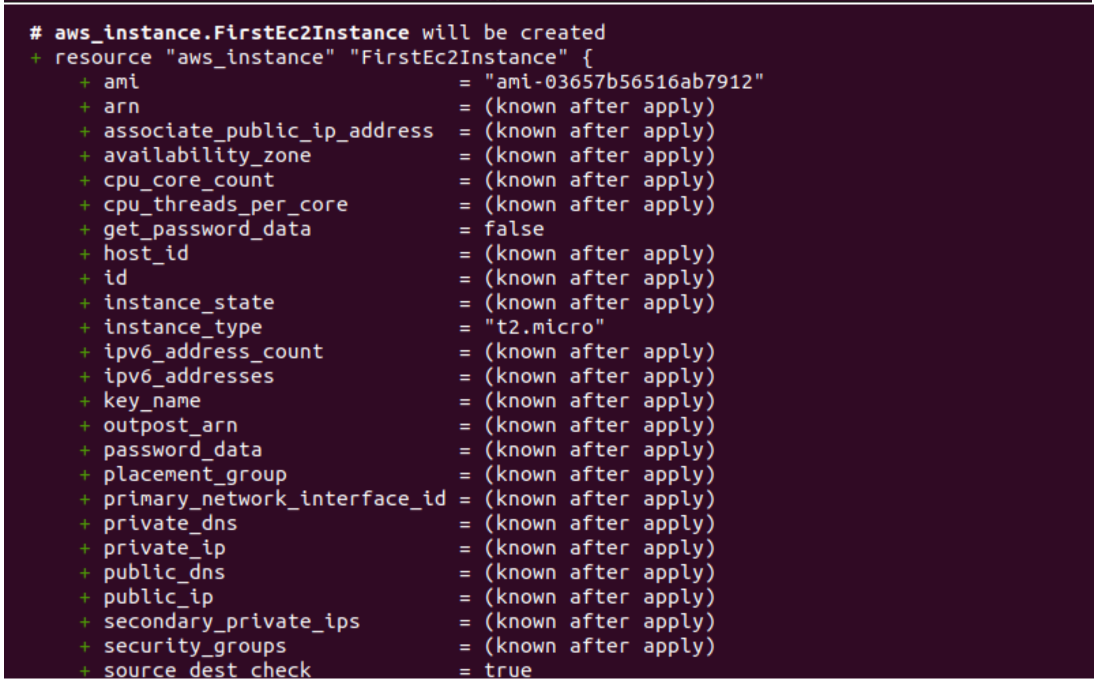


```
darshasawa@ubuntu:~/Documents/SPCM/assignment1$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.15.0...
- Installed hashicorp/aws v3.15.0 (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 = "~> 3.15.0"

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.
```

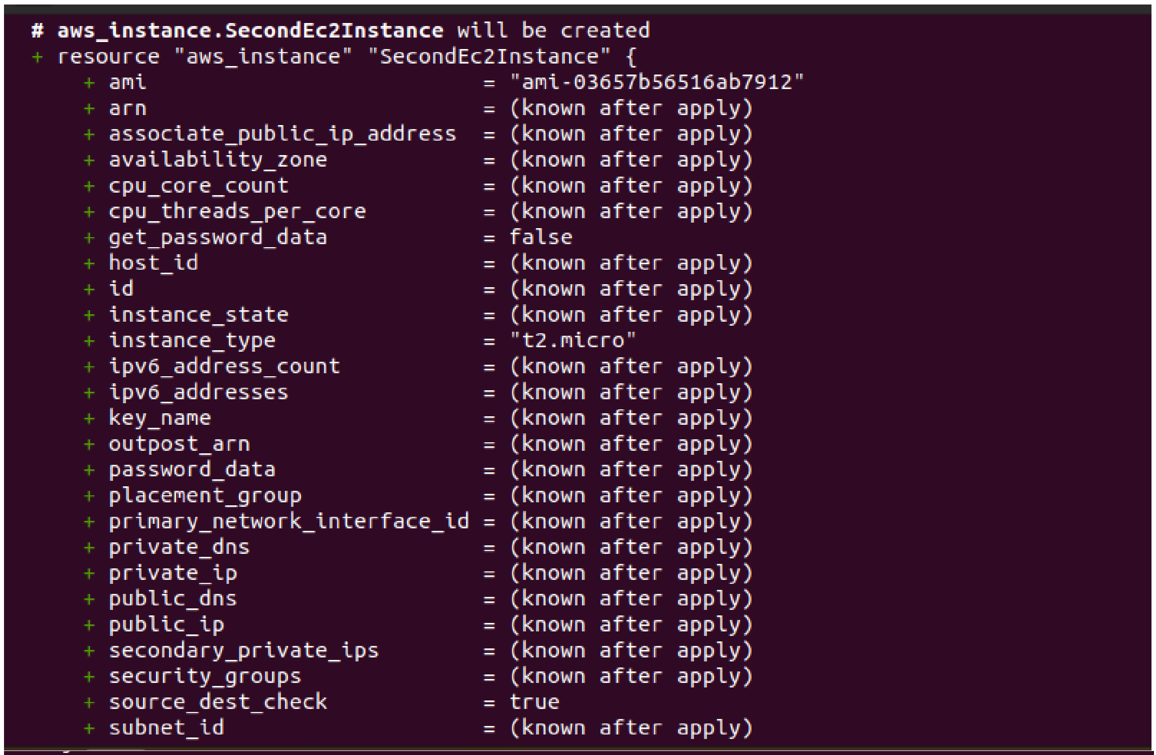
4. Now that you have the provider code downloaded, run the **terraform plan** command. The plan command lets you see what Terraform will do before actually making any changes. This is a great way to sanity check your code before unleashing it onto the world. Anything with a plus sign (+) will be created, anything with a minus sign (–) will be deleted, and anything with a tilde sign (~) will be modified in place.



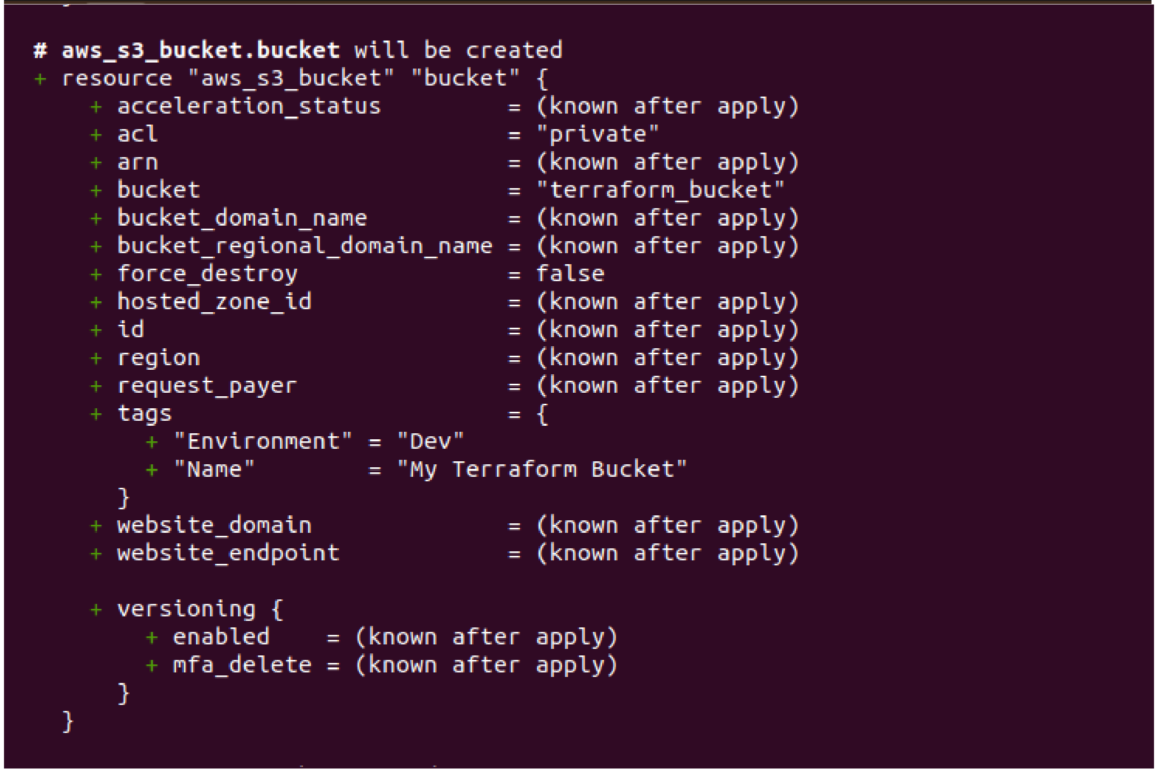
```
darshasawa@ubuntu:~/Documents/SPCM/assignment1$ ls
main.tf  providers.tf
darshasawa@ubuntu:~/Documents/SPCM/assignment1$ terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_customer_gateway.customer_gateway will be created
  + resource "aws_customer_gateway" "customer_gateway" {
      + arn        = (known after apply)
      + bgp_asn    = "65000"
      + id         = (known after apply)
      + ip_address = "172.0.0.1"
      + type       = "ipsec.1"
    }

  # aws_instance.FirstEc2Instance will be created
  + resource "aws_instance" "FirstEc2Instance" {
      + ami                          = "ami-03657b56516ab7912"
```



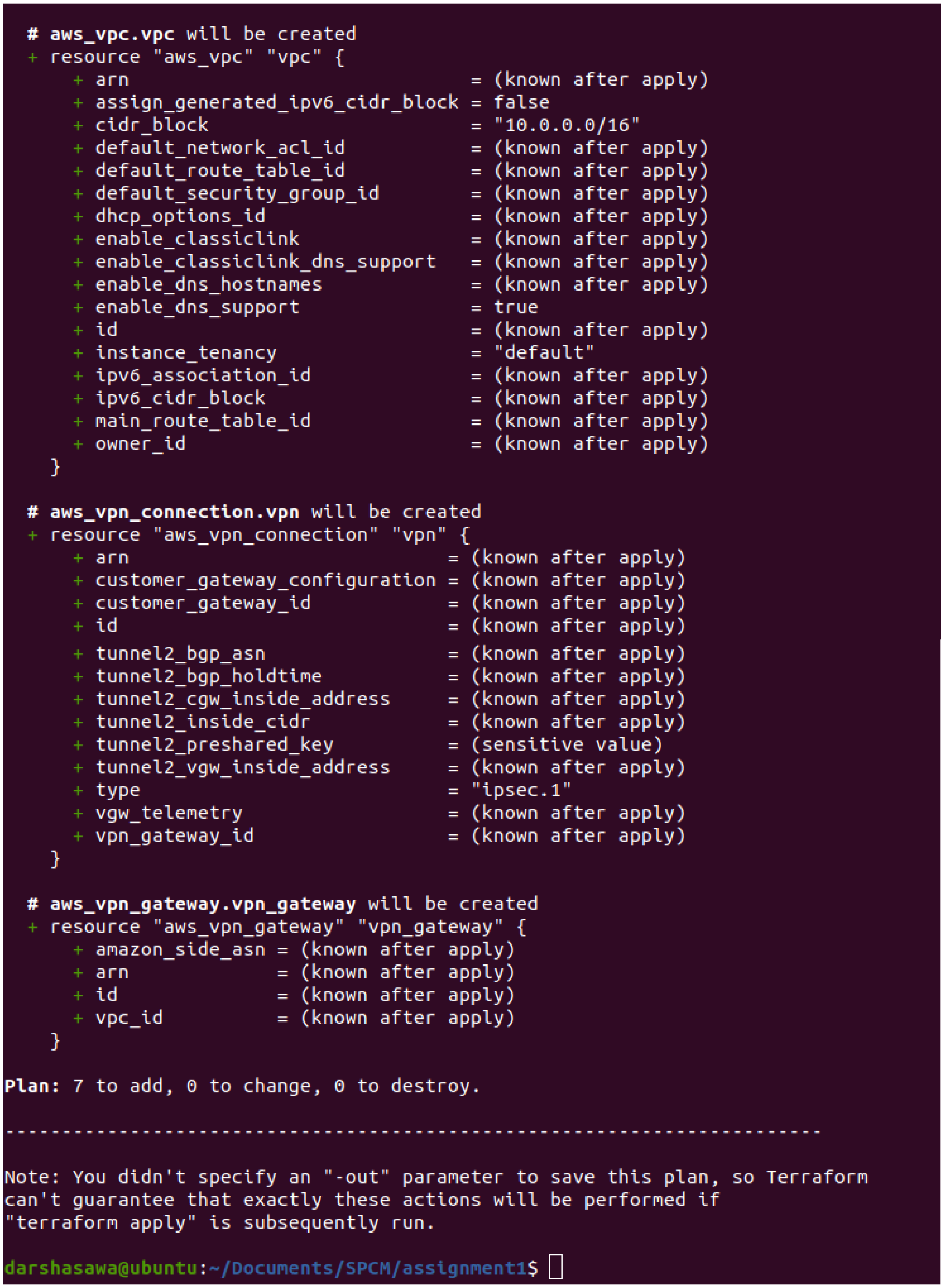
```
  # aws_instance.FirstEc2Instance will be created
  + resource "aws_instance" "FirstEc2Instance" {
      + ami                          = "ami-03657b56516ab7912"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)
      + availability_zone            = (known after apply)
      + cpu_core_count               = (known after apply)
      + cpu_threads_per_core         = (known after apply)
      + get_password_data            = false
      + host_id                      = (known after apply)
      + id                           = (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)
      + outpost_arn                  = (known after apply)
      + password_data                = (known after apply)
      + placement_group              = (known after apply)
      + primary_network_interface_id = (known after apply)
      + private_dns                  = (known after apply)
      + private_ip                   = (known after apply)
      + public_dns                   = (known after apply)
      + public_ip                    = (known after apply)
      + secondary_private_ips        = (known after apply)
      + security_groups              = (known after apply)
      + source_dest_check            = true
```



```
  # aws_instance.SecondEc2Instance will be created
  + resource "aws_instance" "SecondEc2Instance" {
      + ami                          = "ami-03657b56516ab7912"
      + 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)
      + get_password_data            = false
      + host_id                      = (known after apply)
      + id                           = (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)
      + outpost_arn                  = (known after apply)
      + password_data                = (known after apply)
      + placement_group              = (known after apply)
      + primary_network_interface_id = (known after apply)
      + private_dns                  = (known after apply)
      + private_ip                   = (known after apply)
      + public_dns                   = (known after apply)
      + public_ip                    = (known after apply)
      + secondary_private_ips        = (known after apply)
      + security_groups              = (known after apply)
      + source_dest_check            = true
      + subnet_id                    = (known after apply)
```

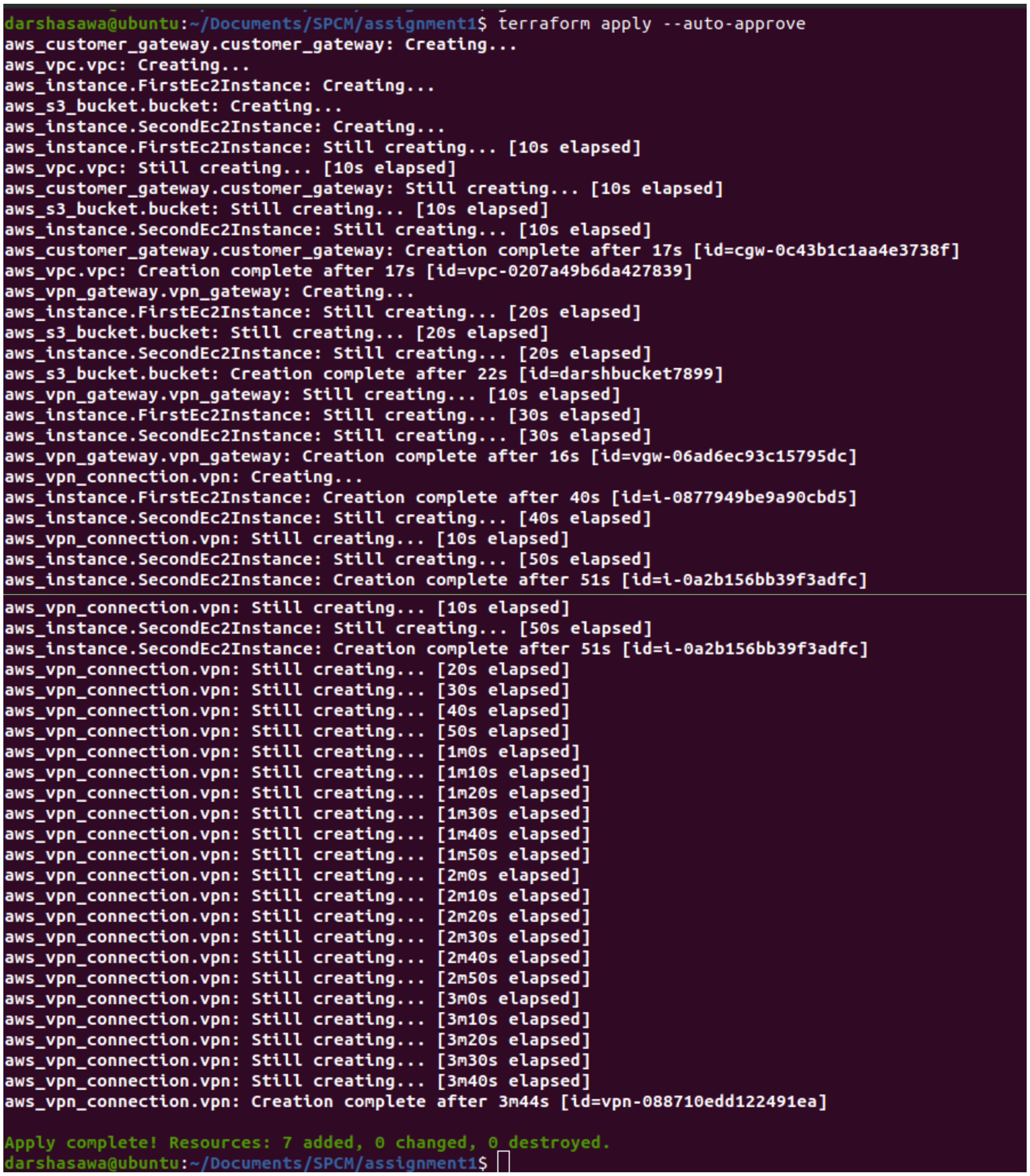


```
  # aws_s3_bucket.bucket will be created
  + resource "aws_s3_bucket" "bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = "private"
      + arn                         = (known after apply)
      + bucket                      = "terraform_bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "My Terraform Bucket"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + versioning {
          + enabled    = (known after apply)
          + mfa_delete = (known after apply)
        }
    }
```



```
  # aws_vpc.vpc will be created
  + resource "aws_vpc" "vpc" {
      + arn                              = (known after apply)
      + assign_generated_ipv6_cidr_block = false
      + cidr_block                       = "10.0.0.0/16"
      + default_network_acl_id           = (known after apply)
      + default_route_table_id           = (known after apply)
      + default_security_group_id        = (known after apply)
      + dhcp_options_id                  = (known after apply)
      + enable_classiclink               = (known after apply)
      + enable_classiclink_dns_support   = (known after apply)
      + enable_dns_hostnames             = (known after apply)
      + enable_dns_support               = true
      + id                               = (known after apply)
      + instance_tenancy                 = "default"
      + ipv6_association_id              = (known after apply)
      + ipv6_cidr_block                  = (known after apply)
      + main_route_table_id              = (known after apply)
      + owner_id                         = (known after apply)
    }

  # aws_vpn_connection.vpn will be created
  + resource "aws_vpn_connection" "vpn" {
      + arn                            = (known after apply)
      + customer_gateway_configuration = (known after apply)
      + customer_gateway_id            = (known after apply)
      + id                             = (known after apply)
      + tunnel2_bgp_asn                = (known after apply)
      + tunnel2_bgp_holdtime           = (known after apply)
      + tunnel2_cgw_inside_address     = (known after apply)
      + tunnel2_inside_cidr            = (known after apply)
      + tunnel2_preshared_key          = (sensitive value)
      + tunnel2_vgw_inside_address     = (known after apply)
      + type                           = "ipsec.1"
      + vgw_telemetry                  = (known after apply)
      + vpn_gateway_id                 = (known after apply)
    }

  # aws_vpn_gateway.vpn_gateway will be created
  + resource "aws_vpn_gateway" "vpn_gateway" {
      + amazon_side_asn = (known after apply)
      + arn             = (known after apply)
      + id              = (known after apply)
      + vpc_id          = (known after apply)
    }

Plan: 7 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

darshasawa@ubuntu:~/Documents/SPCM/assignment1$
```

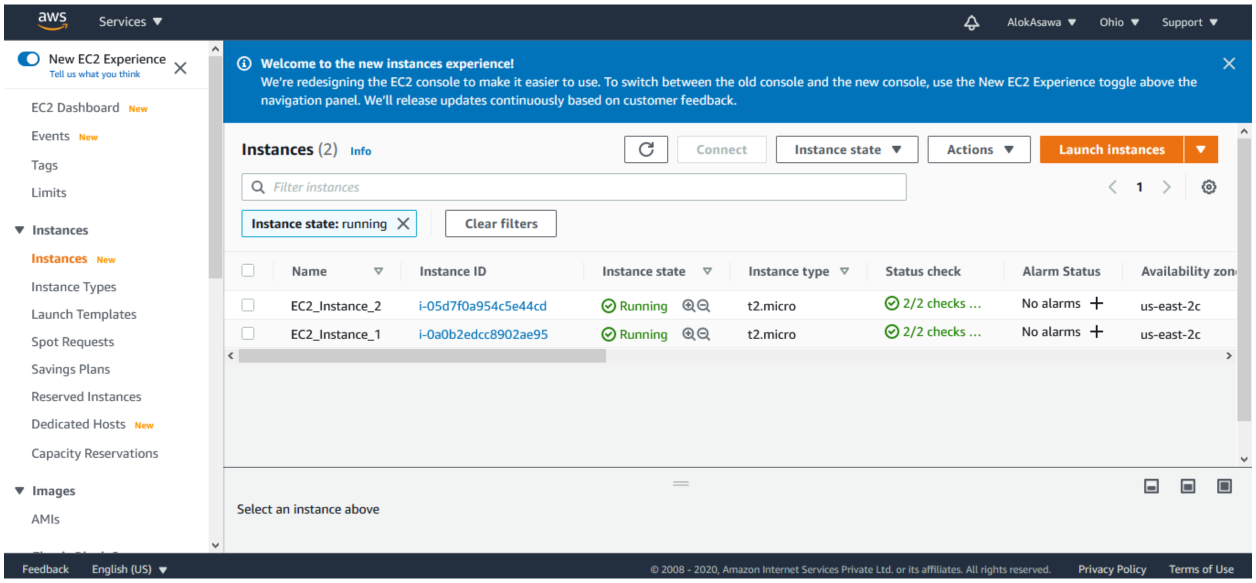
5. To actually create the Instance, run the **terraform apply** command. Adding –auto-approve will eliminate the requirement of entering “yes” after command execution.



```
darshasawa@ubuntu:~/Documents/SPCM/assignment1$ terraform apply --auto-approve
aws_customer_gateway.customer_gateway: Creating...
aws_vpc.vpc: Creating...
aws_instance.FirstEc2Instance: Creating...
aws_s3_bucket.bucket: Creating...
aws_instance.SecondEc2Instance: Creating...
aws_instance.FirstEc2Instance: Still creating... [10s elapsed]
aws_vpc.vpc: Still creating... [10s elapsed]
aws_customer_gateway.customer_gateway: Still creating... [10s elapsed]
aws_s3_bucket.bucket: Still creating... [10s elapsed]
aws_instance.SecondEc2Instance: Still creating... [10s elapsed]
aws_customer_gateway.customer_gateway: Creation complete after 17s [id=cgw-0c43b1c1aa4e3738f]
aws_vpc.vpc: Creation complete after 17s [id=vpc-0207a49b6da427839]
aws_vpn_gateway.vpn_gateway: Creating...
aws_instance.FirstEc2Instance: Still creating... [20s elapsed]
aws_s3_bucket.bucket: Still creating... [20s elapsed]
aws_instance.SecondEc2Instance: Still creating... [20s elapsed]
aws_s3_bucket.bucket: Creation complete after 22s [id=darshbucket7899]
aws_vpn_gateway.vpn_gateway: Still creating... [10s elapsed]
aws_instance.FirstEc2Instance: Still creating... [30s elapsed]
aws_instance.SecondEc2Instance: Still creating... [30s elapsed]
aws_vpn_gateway.vpn_gateway: Creation complete after 16s [id=vgw-06ad6ec93c15795dc]
aws_vpn_connection.vpn: Creating...
aws_instance.FirstEc2Instance: Creation complete after 40s [id=i-0877949be9a90cbd5]
aws_instance.SecondEc2Instance: Still creating... [40s elapsed]
aws_vpn_connection.vpn: Still creating... [10s elapsed]
aws_instance.SecondEc2Instance: Still creating... [50s elapsed]
aws_instance.SecondEc2Instance: Creation complete after 51s [id=i-0a2b156bb39f3adfc]
```

```
aws_vpn_connection.vpn: Still creating... [10s elapsed]
aws_instance.SecondEc2Instance: Still creating... [50s elapsed]
aws_instance.SecondEc2Instance: Creation complete after 51s [id=i-0a2b156bb39f3adfc]
aws_vpn_connection.vpn: Still creating... [20s elapsed]
aws_vpn_connection.vpn: Still creating... [30s elapsed]
aws_vpn_connection.vpn: Still creating... [40s elapsed]
aws_vpn_connection.vpn: Still creating... [50s elapsed]
aws_vpn_connection.vpn: Still creating... [1m0s elapsed]
aws_vpn_connection.vpn: Still creating... [1m10s elapsed]
aws_vpn_connection.vpn: Still creating... [1m20s elapsed]
aws_vpn_connection.vpn: Still creating... [1m30s elapsed]
aws_vpn_connection.vpn: Still creating... [1m40s elapsed]
aws_vpn_connection.vpn: Still creating... [1m50s elapsed]
aws_vpn_connection.vpn: Still creating... [2m0s elapsed]
aws_vpn_connection.vpn: Still creating... [2m10s elapsed]
aws_vpn_connection.vpn: Still creating... [2m20s elapsed]
aws_vpn_connection.vpn: Still creating... [2m30s elapsed]
aws_vpn_connection.vpn: Still creating... [2m40s elapsed]
aws_vpn_connection.vpn: Still creating... [2m50s elapsed]
aws_vpn_connection.vpn: Still creating... [3m0s elapsed]
aws_vpn_connection.vpn: Still creating... [3m10s elapsed]
aws_vpn_connection.vpn: Still creating... [3m20s elapsed]
aws_vpn_connection.vpn: Still creating... [3m30s elapsed]
aws_vpn_connection.vpn: Still creating... [3m40s elapsed]
aws_vpn_connection.vpn: Creation complete after 3m44s [id=vpn-088710edd122491ea]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
darshasawa@ubuntu:~/Documents/SPCM/assignment1$
```

6. Now the resources are deployed to the AWS account using Terraform.



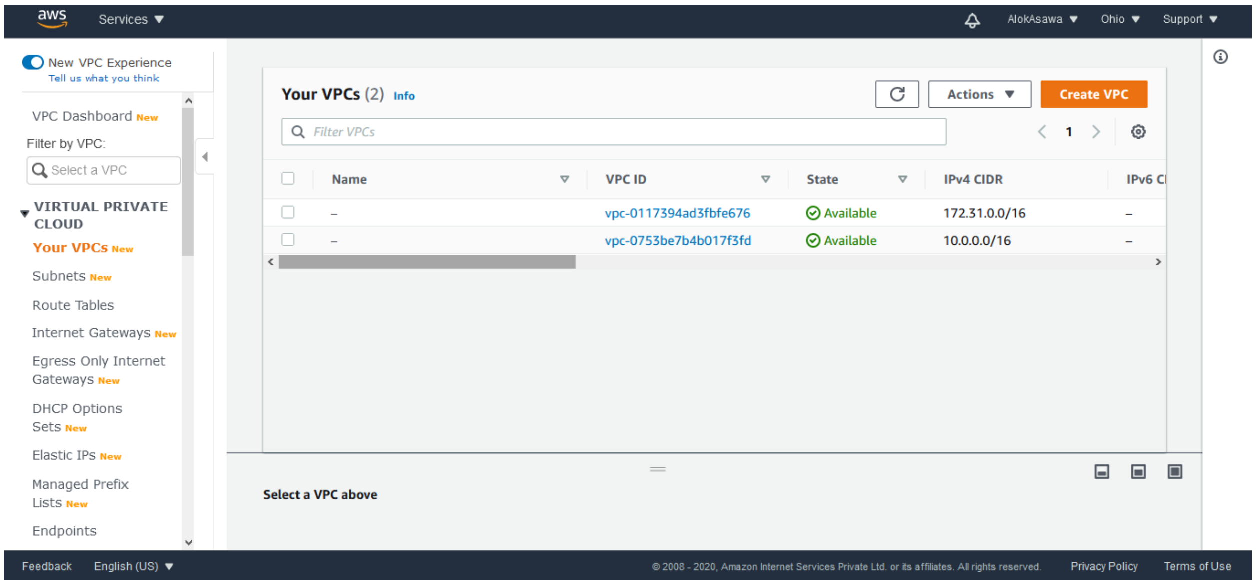
Welcome to the new instances experience!
We're redesigning the EC2 console to make it easier to use. To switch between the old console and the new console, use the New EC2 Experience toggle above the navigation panel. We'll release updates continuously based on customer feedback.

Instances (2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm Status	Availability zone
EC2_Instance_2	i-05d7f0a954c5e44cd	Running	t2.micro	2/2 checks ...	No alarms	us-east-2c
EC2_Instance_1	i-0a0b2edcc8902ae95	Running	t2.micro	2/2 checks ...	No alarms	us-east-2c

Select an instance above

EC2 instances

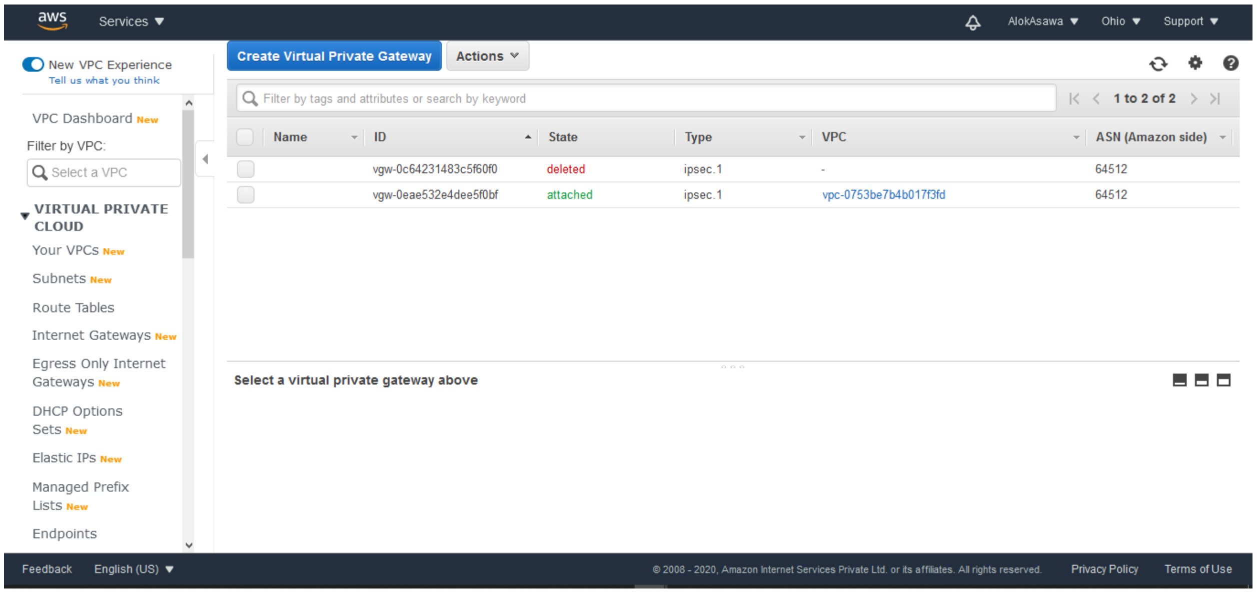


Your VPCs (2) Info

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
–	vpc-0117394ad3fbfe676	Available	172.31.0.0/16	–
–	vpc-0753be7b4b017f3fd	Available	10.0.0.0/16	–

Select a VPC above

Virtual Private Cloud

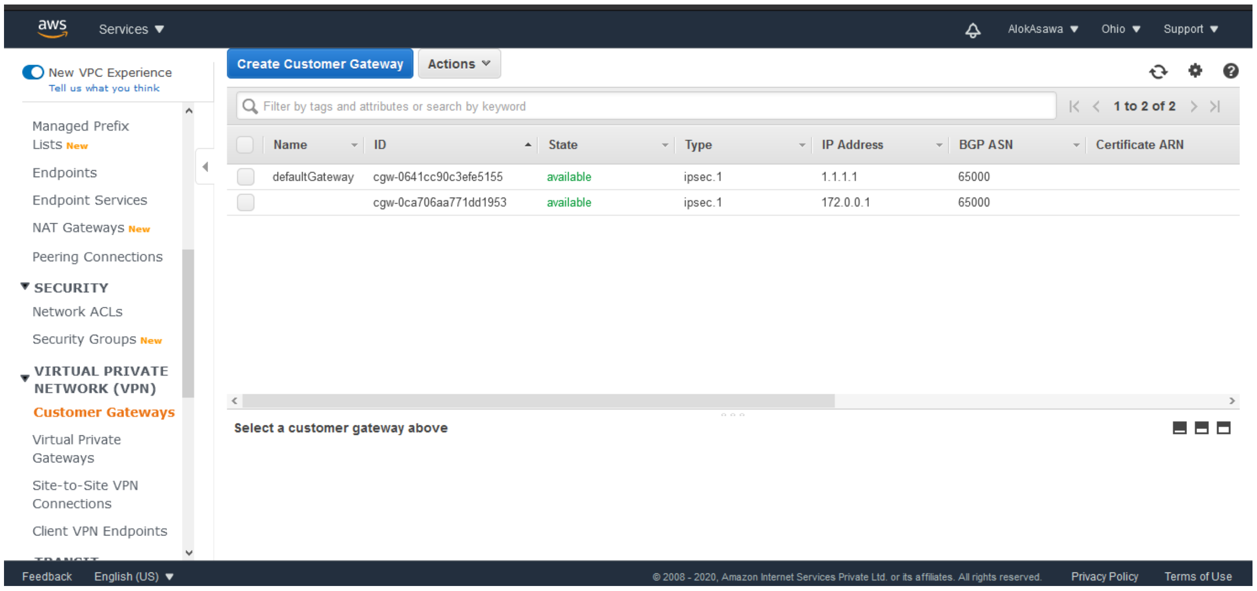


Create Virtual Private Gateway | Actions

Name	ID	State	Type	VPC	ASN (Amazon side)
	vgw-0c64231483c5f60f0	deleted	ipsec.1	-	64512
	vgw-0eae532e4dee5f0bf	attached	ipsec.1	vpc-0753be7b4b017f3fd	64512

Select a virtual private gateway above

Virtual Private Gateway

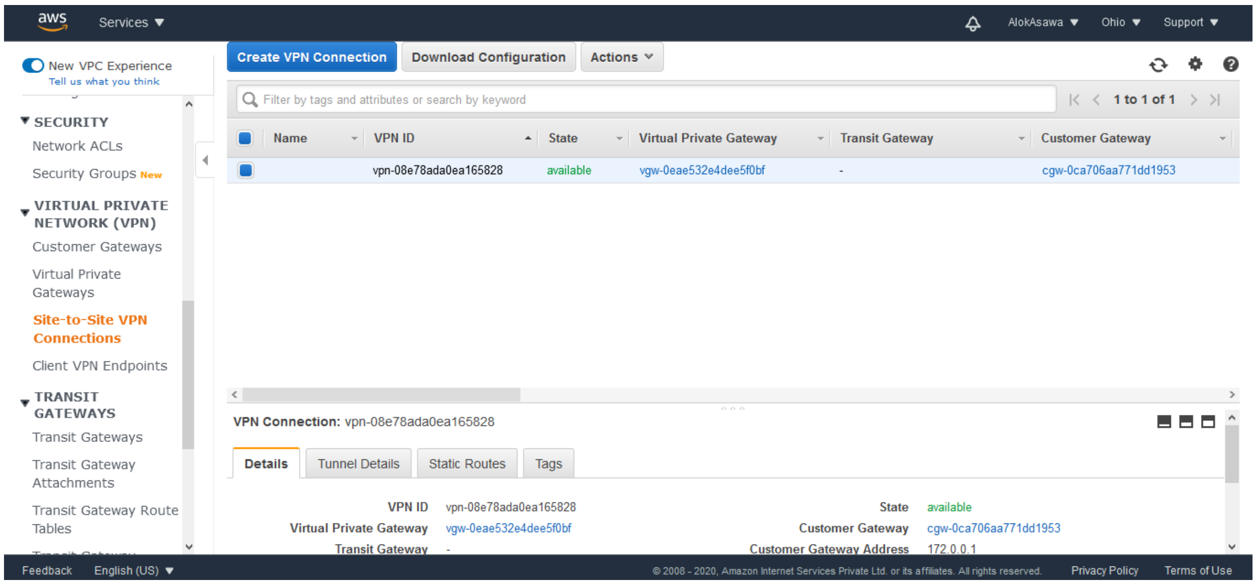


Create Customer Gateway | Actions

Name	ID	State	Type	IP Address	BGP ASN	Certificate ARN
defaultGateway	cgw-0641cc90c3efe5155	available	ipsec.1	1.1.1.1	65000	
	cgw-0ca706aa771dd1953	available	ipsec.1	172.0.0.1	65000	

Select a customer gateway above

Customer Gateway



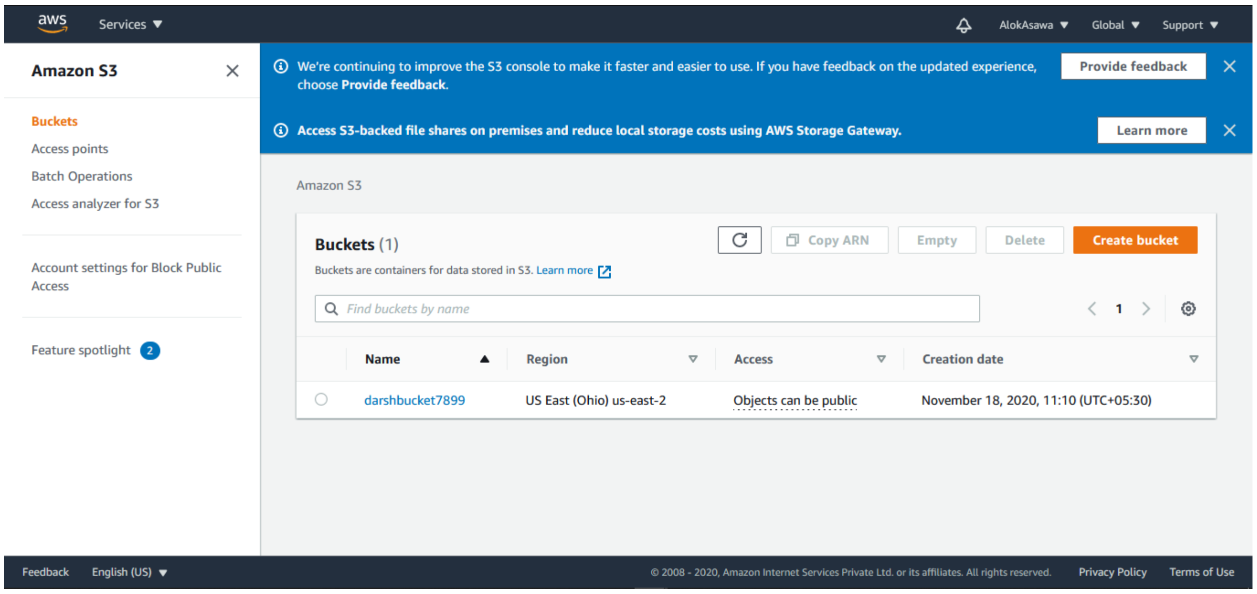
Name	VPN ID	State	Virtual Private Gateway	Transit Gateway	Customer Gateway
	vpn-08e78ada0ea165828	available	vgw-0eae532e4dee5f0bf	-	cgw-0ca706aa771dd1953

VPN Connection: vpn-08e78ada0ea165828

VPN ID	vpn-08e78ada0ea165828	State	available
Virtual Private Gateway	vgw-0eae532e4dee5f0bf	Customer Gateway	cgw-0ca706aa771dd1953
Transit Gateway	-	Customer Gateway Address	172.0.0.1

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Virtual Private Network Connection



We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose **Provide feedback**.

Access S3-backed file shares on premises and reduce local storage costs using AWS Storage Gateway.

Amazon S3

Buckets (1)

Buckets are containers for data stored in S3. Learn more

Name	Region	Access	Creation date
darshbucket7899	US East (Ohio) us-east-2	Objects can be public	November 18, 2020, 11:10 (UTC+05:30)

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

S3 Bucket