UPES
UNIVERSITY WITH A PURPOSE

Format No. QSP/7.1/01.F01 (B) **Issue No.05 Rev. No 5 Dated: Jan 1, 2017**

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

School of Computer Science

Dehradun

ASSIGNMENT 1

Programme :	B. Tech in Computer Science and Engineering with Specialization in DevOps
Course :	System Provisioning and Configuration Management
Semester :	VII
Session :	August 2020 - December 2020
Batch :	2017-2021
Faculty :	Dr. Hitesh Kumar Sharma

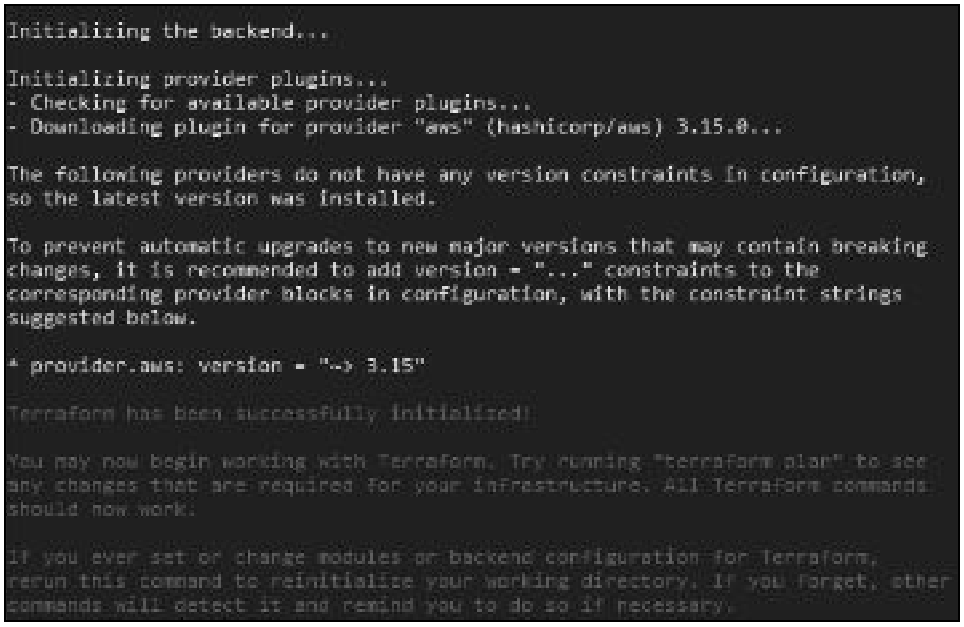
Submitted By:
Roll No- R171217055
SAP ID- 500062194
Name- Shraddha Saini

Faculty Signature

Step 1:

First create a directory project-terraform and initialise terraform which is installed on your system by following command:

>>> Terraform init

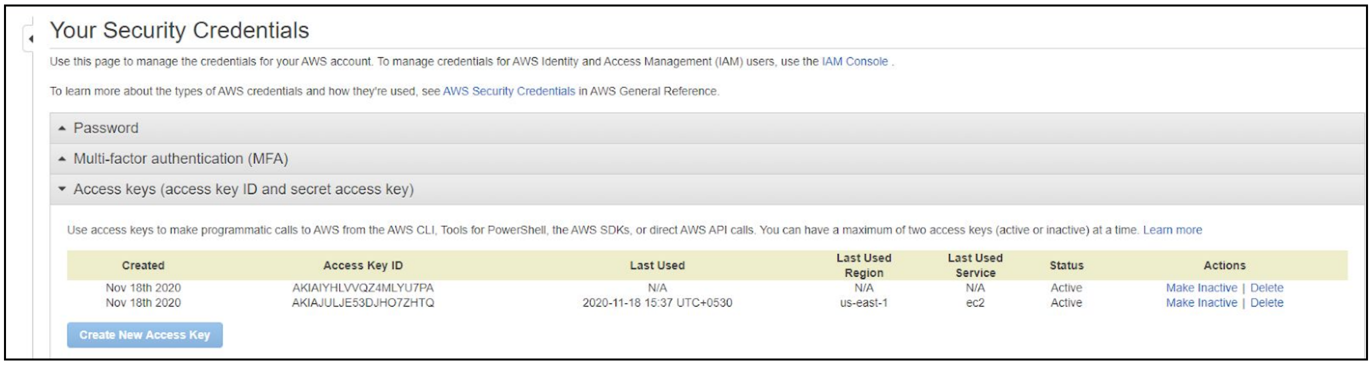


```
Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 3.15.0...

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, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 3.15"

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.
```

Step 2:

Now, setup a connection to aws using the access key and secret key which you can create and download from your aws management console by clicking:

>>> your name -> security credentials -> access keys:



Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the IAM Console .

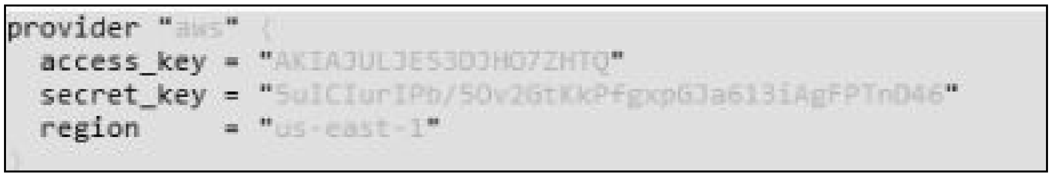
To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

▲ Password
▲ Multi-factor authentication (MFA)
▼ Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, the AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

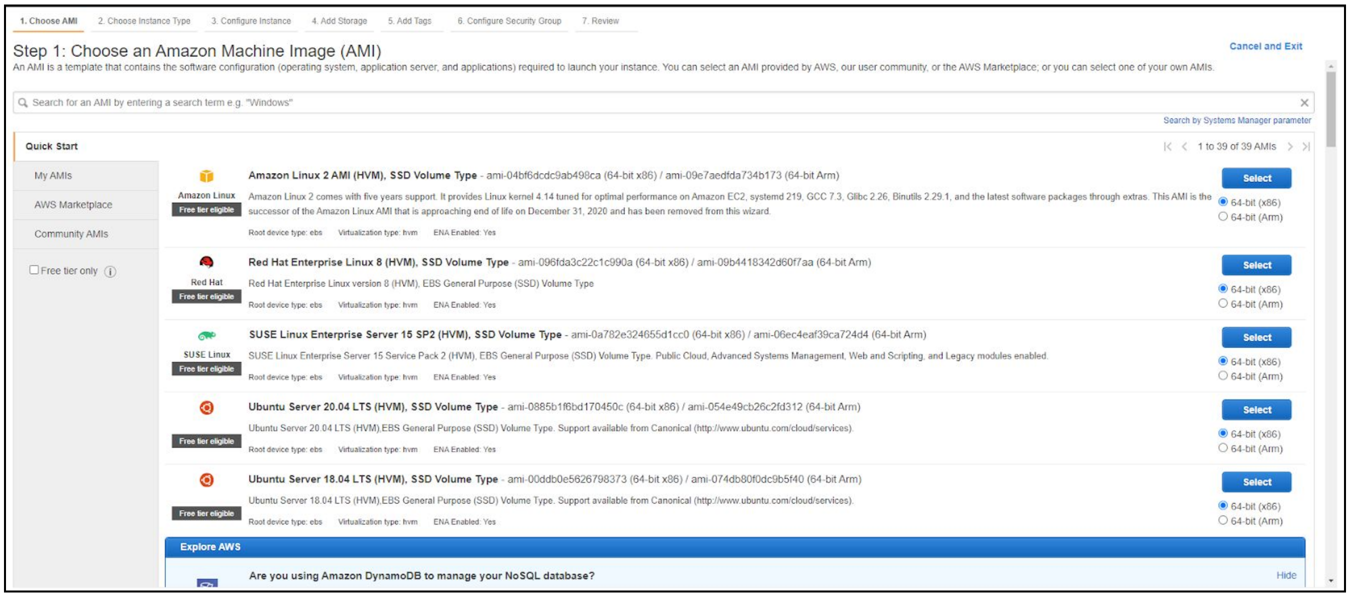
Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
Nov 18th 2020	AKIAIYHLVVQZ4MLYU7PA	N/A	N/A	N/A	Active	Make Inactive | Delete
Nov 18th 2020	AKIAJULJE53DJHO7ZHTQ	2020-11-18 15:37 UTC+0530	us-east-1	ec2	Active	Make Inactive | Delete

Now, create a file using ***vim*** which is will connect to aws and has the access and security key credentials which you have downloaded and enter your region:



```
provider "[redacted]" {
  access_key = "[redacted]"
  secret_key = "[redacted]"
  region     = "[redacted]"
}
```

Then, use ***vim*** to create a file in terraform with ***.tf*** extension and add below commands and set the ami from the screen as shown below and set the instance type as t2 micro and key_name as "mykey":



1. Choose AMI · 2. Choose Instance Type · 3. Configure Instance · 4. Add Storage · 5. Add Tags · 6. Configure Security Group · 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Search by Systems Manager parameter

Quick Start · My AMIs · AWS Marketplace · Community AMIs · Free tier only

1 to 39 of 39 AMIs

Amazon Linux — Free tier eligible
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-04bf6dcdc9ab498ca (64-bit x86) / ami-09e7aedfda734b173 (64-bit Arm)
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.
Root device type: ebs · Virtualization type: hvm · ENA Enabled: Yes
Select · [✓ 64-bit (x86)] 64-bit (Arm)

Red Hat — Free tier eligible
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-096fda3c22c1c990a (64-bit x86) / ami-09b4418342d60f7aa (64-bit Arm)
Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs · Virtualization type: hvm · ENA Enabled: Yes
Select · [✓ 64-bit (x86)] 64-bit (Arm)

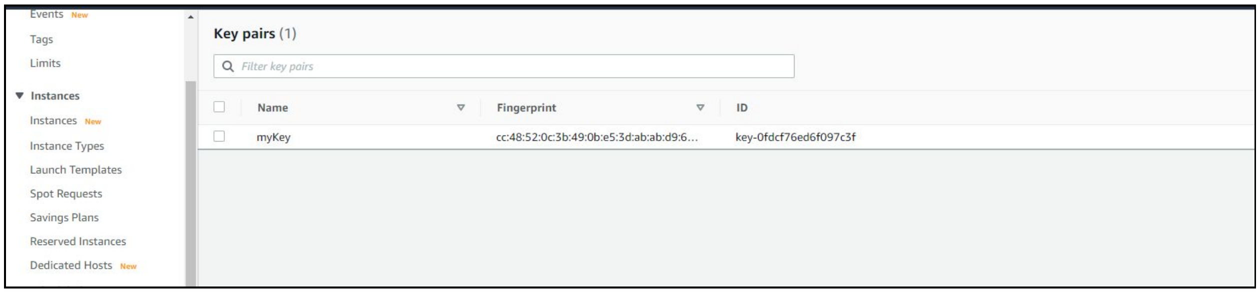
SUSE Linux — Free tier eligible
SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0a782e324655d1cc0 (64-bit x86) / ami-06ec4eaf39ca724d4 (64-bit Arm)
SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs · Virtualization type: hvm · ENA Enabled: Yes
Select · [✓ 64-bit (x86)] 64-bit (Arm)

Free tier eligible
Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0885b1f6bd170450c (64-bit x86) / ami-054e49cb26c2fd312 (64-bit Arm)
Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs · Virtualization type: hvm · ENA Enabled: Yes
Select · [✓ 64-bit (x86)] 64-bit (Arm)

Free tier eligible
Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-00ddb0e5626798373 (64-bit x86) / ami-074db80f0dc9b5f40 (64-bit Arm)
Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs · Virtualization type: hvm · ENA Enabled: Yes
Select · [✓ 64-bit (x86)] 64-bit (Arm)

Explore AWS

Are you using Amazon DynamoDB to manage your NoSQL database? Hide

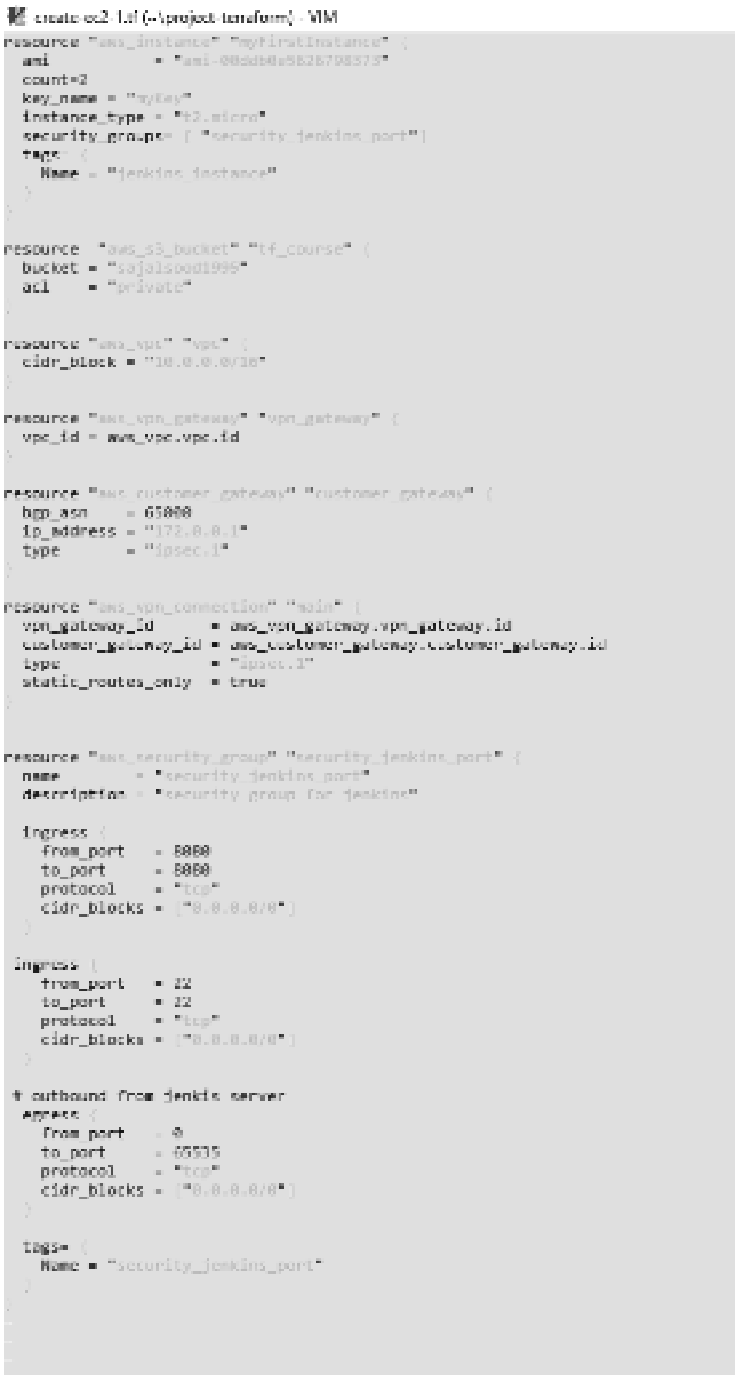


Events New · Tags · Limits · Instances: Instances New · Instance Types · Launch Templates · Spot Requests · Savings Plans · Reserved Instances · Dedicated Hosts New

Key pairs (1)

Filter key pairs

Name	Fingerprint	ID
myKey	cc:48:52:0c:3b:49:0b:e5:3d:ab:ab:d9:6...	key-0fdcf76ed6f097c3f



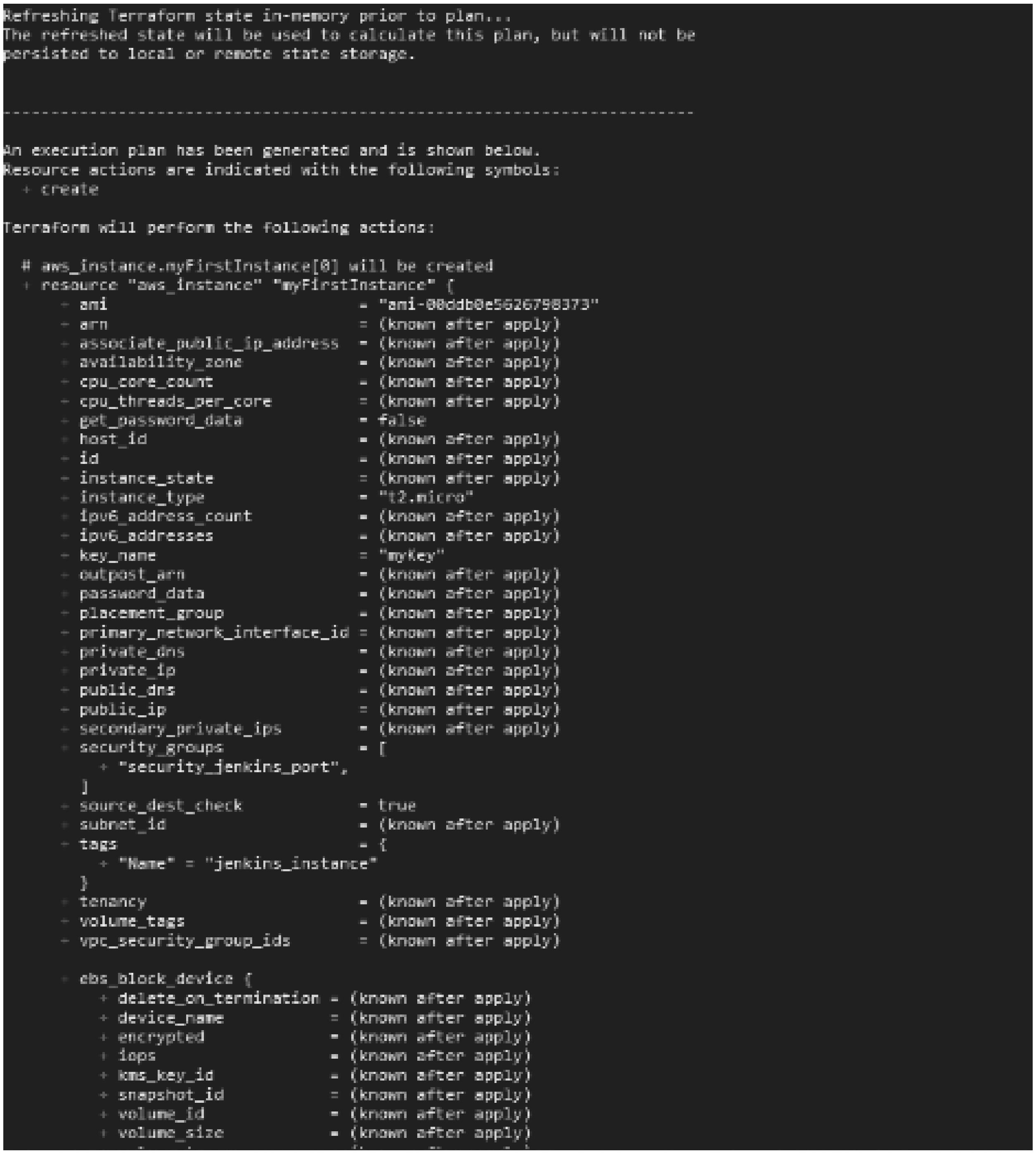
create-ec2-1.tf (~\project-terraform) - VIM

```
resource "aws_instance" "myFirstInstance" {
  ami           = "ami-00ddb0e5626798373"
  count=2
  key_name = "myKey"
  instance_type = "t2.micro"
  security_groups= [ "security_jenkins_port"]
  tags= {
    Name = "jenkins_instance"
  }
}


resource  "aws_s3_bucket" "tf_course" {
  bucket = "sajalsood1995"
  acl    = "private"
}


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
}



resource "aws_security_group" "security_jenkins_port" {
  name        = "security_jenkins_port"
  description = "security group for jenkins"

  ingress {
    from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

 ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

 # outbound from jenkis server
  egress {
    from_port   = 0
    to_port     = 65535
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags= {
    Name = "security_jenkins_port"
  }
}
~
~
~
```

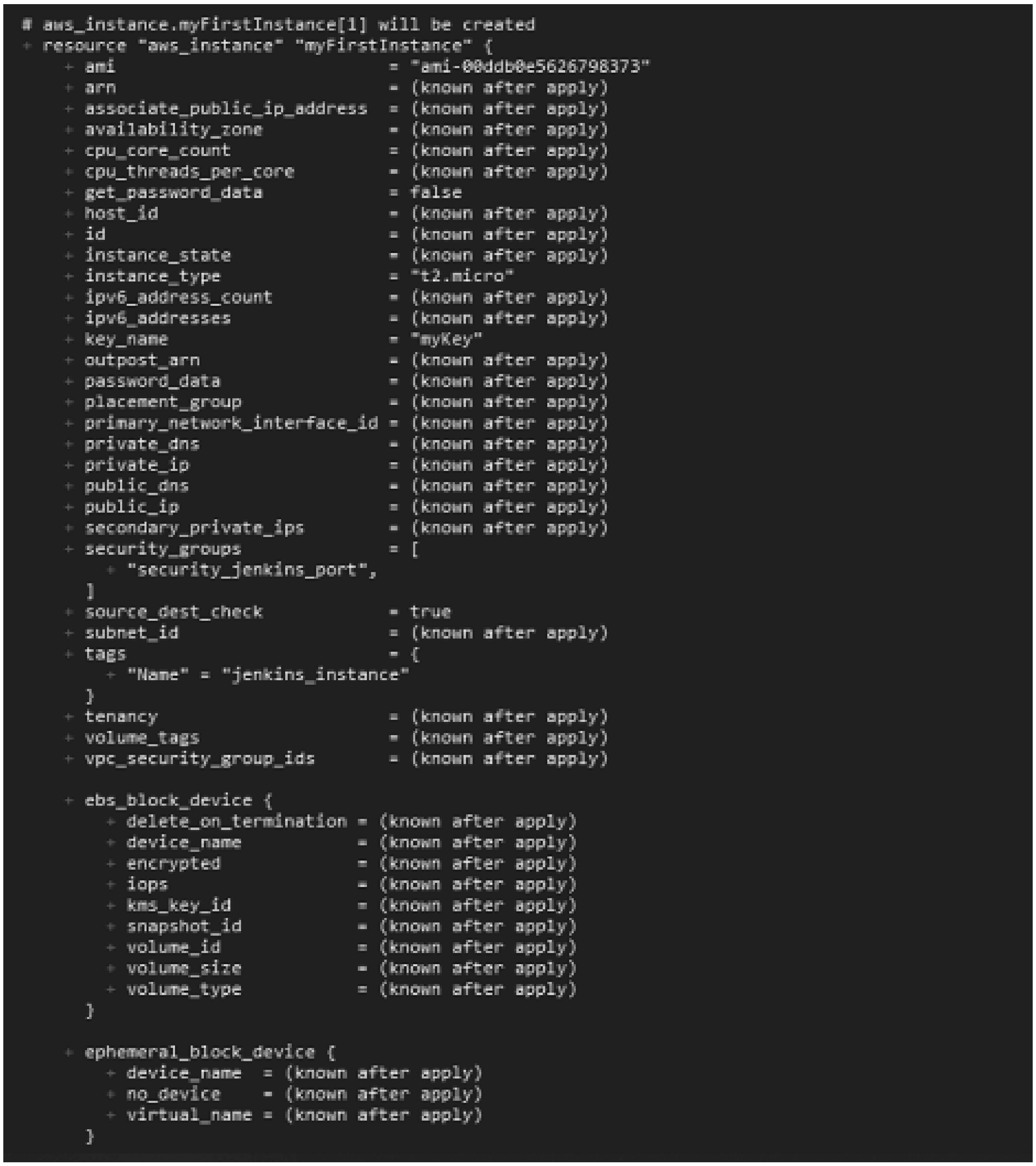
In this file, we add resources like ***instance creation, vpn and S3 bucket***. All these steps to create these 3 added in this file.

Now, apply the following command which depicts all the plans that the file has to perform:

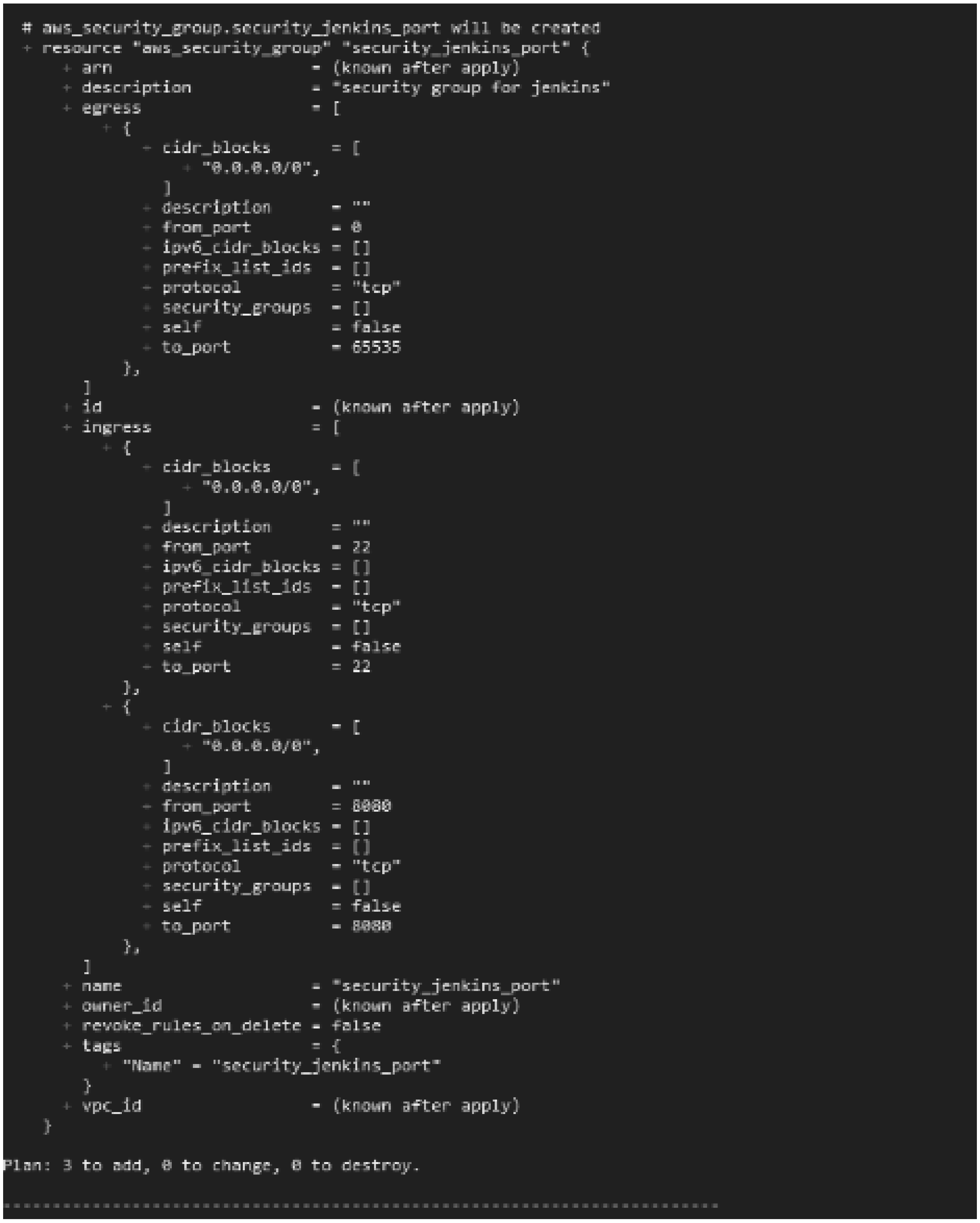
>>> terraform plan



```
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_instance.myFirstInstance[0] will be created
  + resource "aws_instance" "myFirstInstance" {
      + ami                          = "ami-00ddb0e5626798373"
      + 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                     = "myKey"
      + 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              = [
          + "security_jenkins_port",
        ]
      + source_dest_check            = true
      + subnet_id                    = (known after apply)
      + tags                         = {
          + "Name" = "jenkins_instance"
        }
      + tenancy                      = (known after apply)
      + volume_tags                  = (known after apply)
      + vpc_security_group_ids       = (known after apply)

      + ebs_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + snapshot_id           = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
```



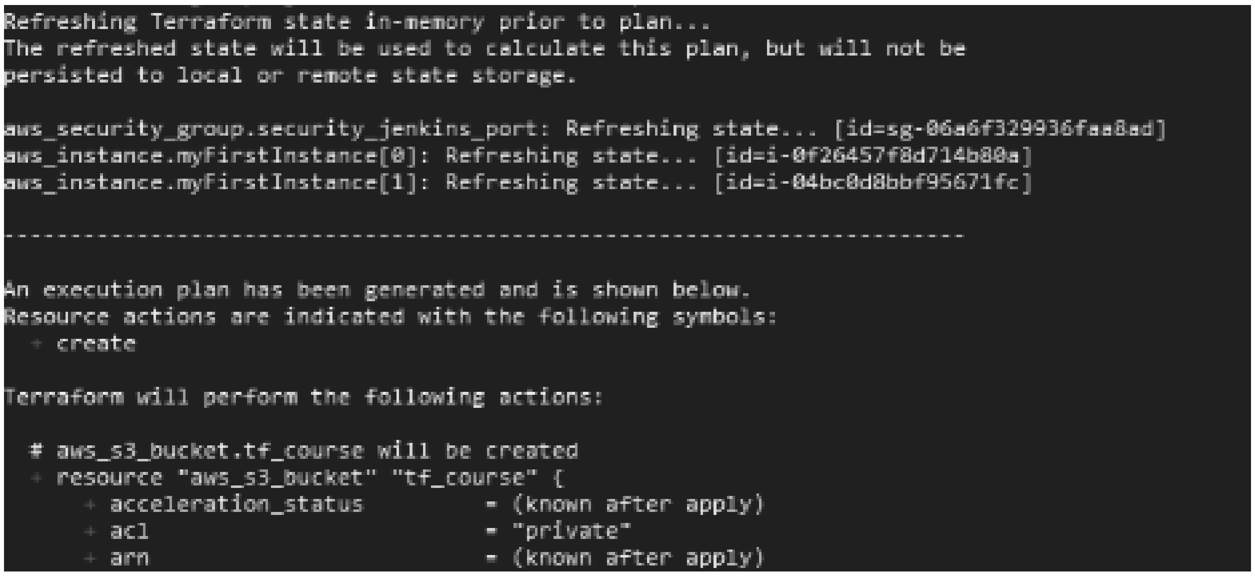
```
# aws_instance.myFirstInstance[1] will be created
+ resource "aws_instance" "myFirstInstance" {
    + ami                          = "ami-00ddb0e5626798373"
    + arn                          = (known after apply)
    + associate_public_ip_address  = (known after apply)
    + availability_zone            = (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                     = "myKey"
    + 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              = [
        + "security_jenkins_port",
      ]
    + source_dest_check            = true
    + subnet_id                    = (known after apply)
    + tags                         = {
        + "Name" = "jenkins_instance"
      }
    + tenancy                      = (known after apply)
    + volume_tags                  = (known after apply)
    + vpc_security_group_ids       = (known after apply)

    + ebs_block_device {
        + delete_on_termination = (known after apply)
        + device_name           = (known after apply)
        + encrypted             = (known after apply)
        + iops                  = (known after apply)
        + kms_key_id            = (known after apply)
        + snapshot_id           = (known after apply)
        + volume_id             = (known after apply)
        + volume_size           = (known after apply)
        + volume_type           = (known after apply)
      }

    + ephemeral_block_device {
        + device_name  = (known after apply)
        + no_device    = (known after apply)
        + virtual_name = (known after apply)
      }
```



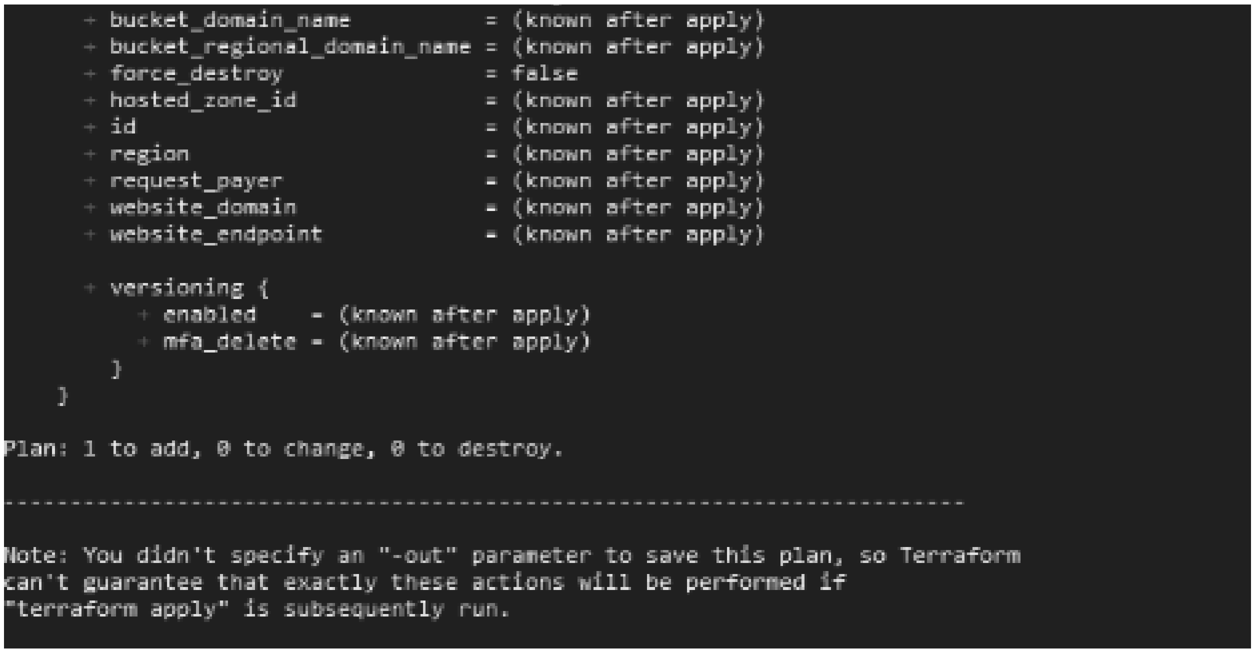
```
  # aws_security_group.security_jenkins_port will be created
  + resource "aws_security_group" "security_jenkins_port" {
      + arn                    = (known after apply)
      + description            = "security group for jenkins"
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 65535
            },
        ]
      + id                     = (known after apply)
      + ingress                = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 22
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 22
            },
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 8080
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 8080
            },
        ]
      + name                   = "security_jenkins_port"
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags                   = {
          + "Name" = "security_jenkins_port"
        }
      + vpc_id                 = (known after apply)
    }

Plan: 3 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------
```

Run the following command to check whether the plans are added :

>>> terraform plan



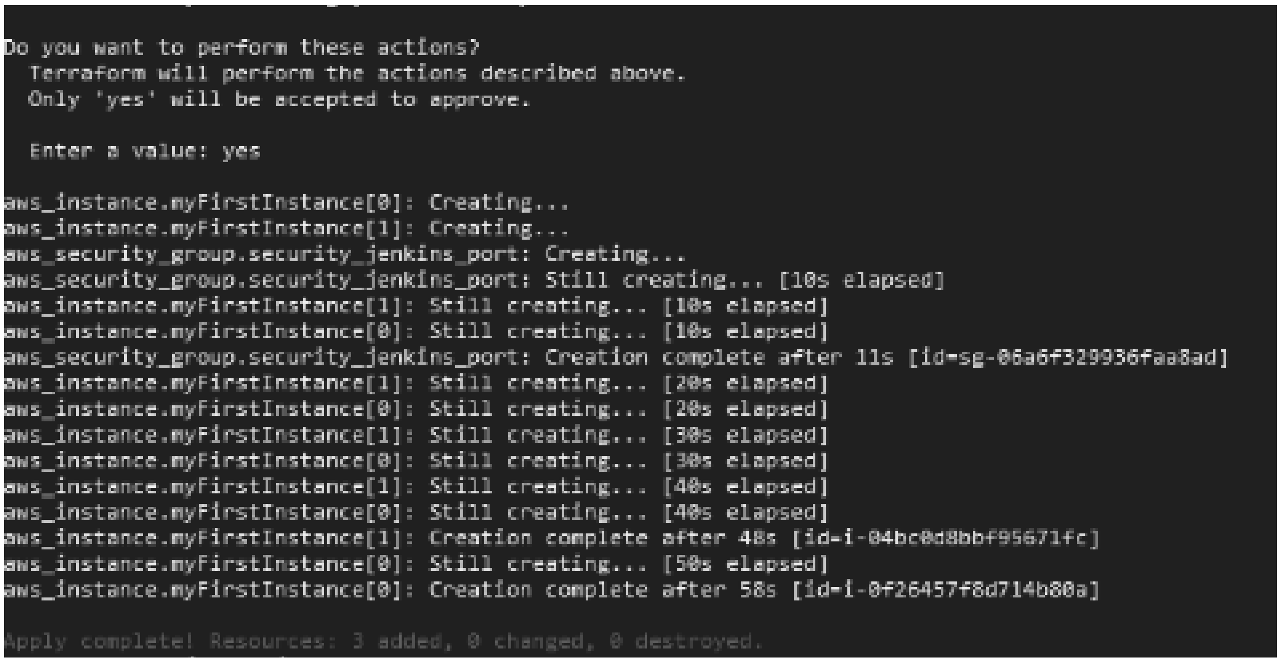
```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

aws_security_group.security_jenkins_port: Refreshing state... [id=sg-06a6f329936faa8ad]
aws_instance.myFirstInstance[0]: Refreshing state... [id=i-0f26457f8d714b80a]
aws_instance.myFirstInstance[1]: Refreshing state... [id=i-04bc0d8bbf95671fc]

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

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_s3_bucket.tf_course will be created
  + resource "aws_s3_bucket" "tf_course" {
      + acceleration_status         = (known after apply)
      + acl                         = "private"
      + arn                         = (known after apply)
```



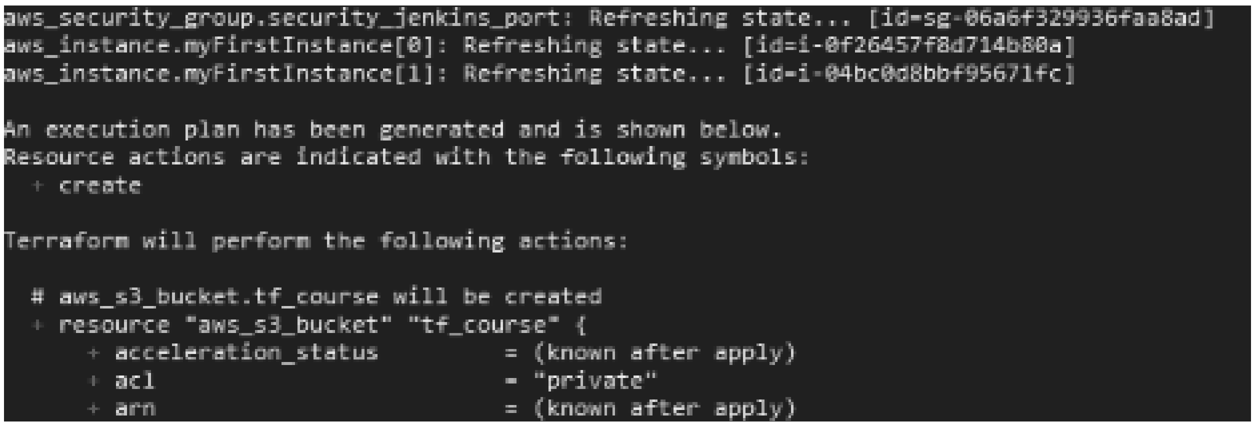
```
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

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

Plan: 1 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.
```

Now Apply the following command through which the script will run:

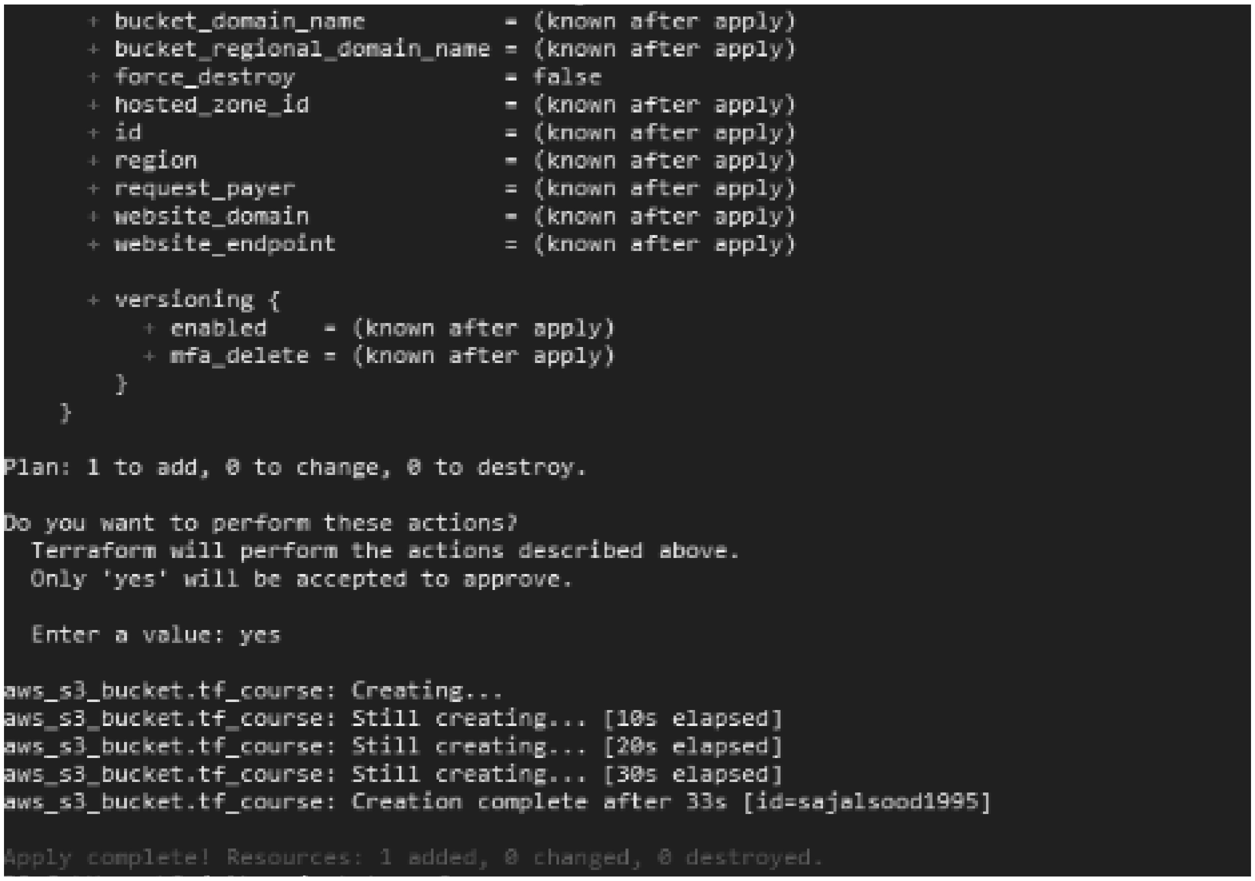
>>> terraform apply



```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.myFirstInstance[0]: Creating...
aws_instance.myFirstInstance[1]: Creating...
aws_security_group.security_jenkins_port: Creating...
aws_security_group.security_jenkins_port: Still creating... [10s elapsed]
aws_instance.myFirstInstance[1]: Still creating... [10s elapsed]
aws_instance.myFirstInstance[0]: Still creating... [10s elapsed]
aws_security_group.security_jenkins_port: Creation complete after 11s [id=sg-06a6f329936faa8ad]
aws_instance.myFirstInstance[1]: Still creating... [20s elapsed]
aws_instance.myFirstInstance[0]: Still creating... [20s elapsed]
aws_instance.myFirstInstance[1]: Still creating... [30s elapsed]
aws_instance.myFirstInstance[0]: Still creating... [30s elapsed]
aws_instance.myFirstInstance[1]: Still creating... [40s elapsed]
aws_instance.myFirstInstance[0]: Still creating... [40s elapsed]
aws_instance.myFirstInstance[1]: Creation complete after 48s [id=i-04bc0d8bbf95671fc]
aws_instance.myFirstInstance[0]: Still creating... [50s elapsed]
aws_instance.myFirstInstance[0]: Creation complete after 58s [id=i-0f26457f8d714b80a]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

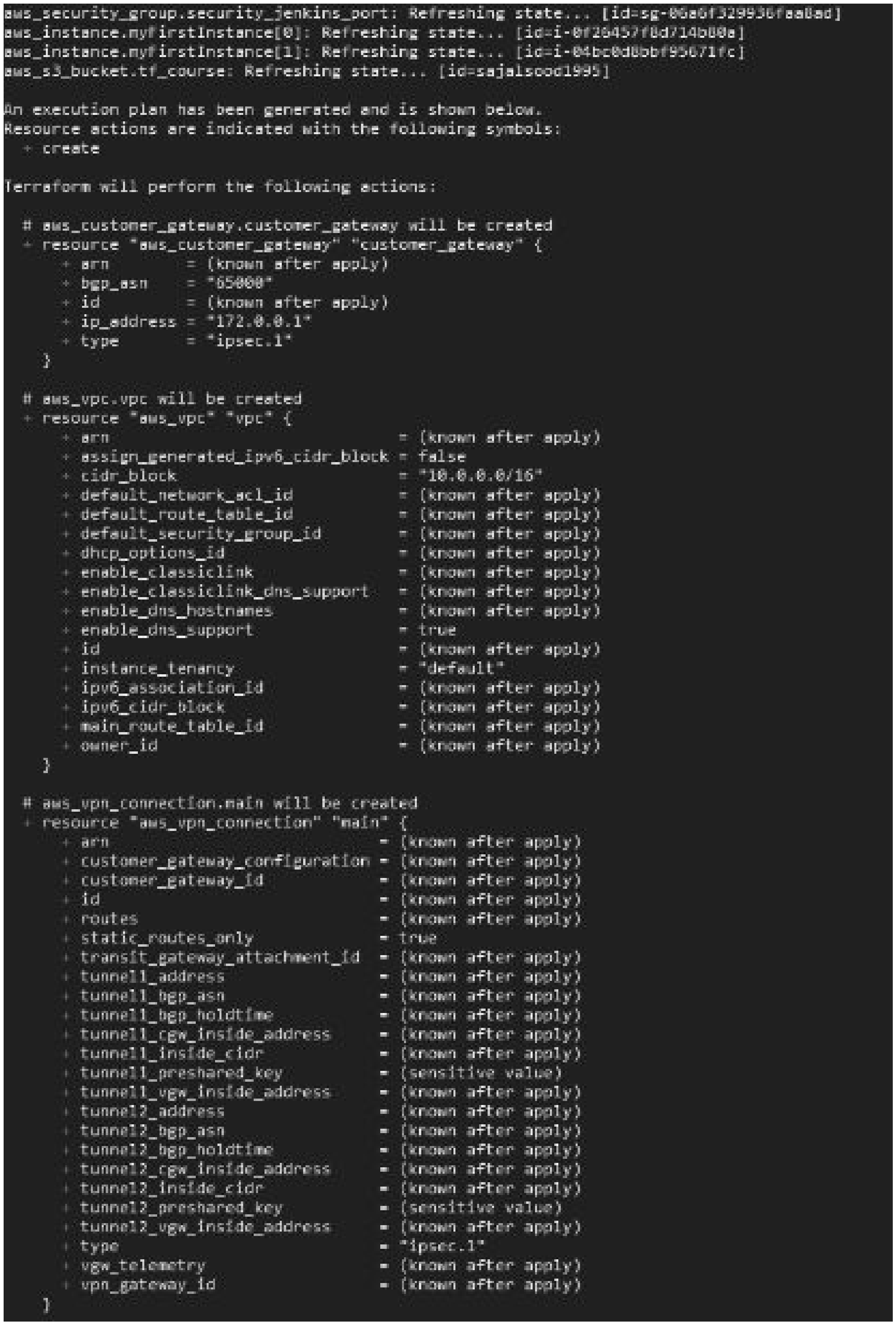


```
aws_security_group.security_jenkins_port: Refreshing state... [id=sg-06a6f329936faa8ad]
aws_instance.myFirstInstance[0]: Refreshing state... [id=i-0f26457f8d714b80a]
aws_instance.myFirstInstance[1]: Refreshing state... [id=i-04bc0d8bbf95671fc]

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_s3_bucket.tf_course will be created
  + resource "aws_s3_bucket" "tf_course" {
      + acceleration_status         = (known after apply)
      + acl                         = "private"
      + arn                         = (known after apply)
```

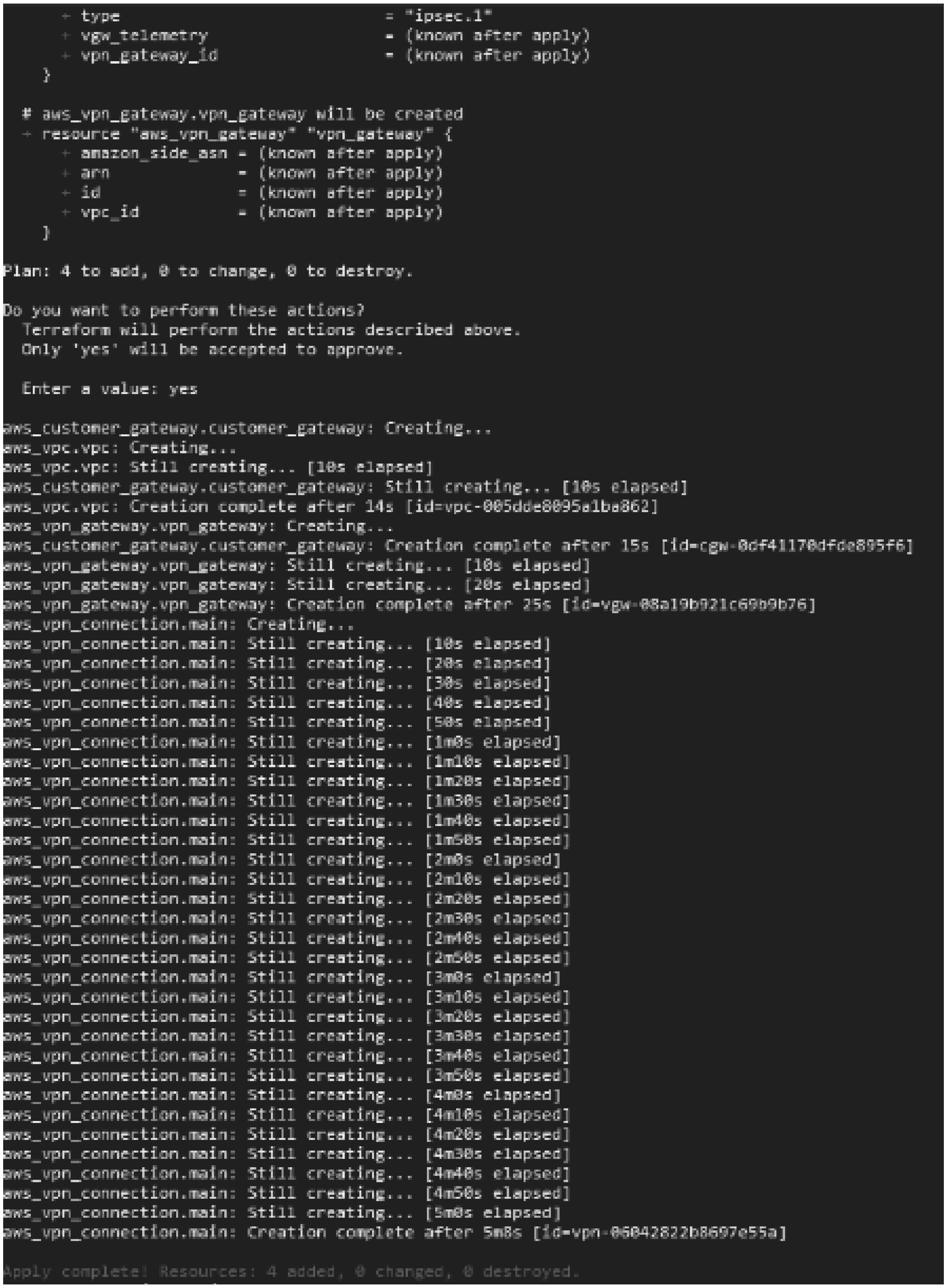


```
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

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

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket.tf_course: Creating...
aws_s3_bucket.tf_course: Still creating... [10s elapsed]
aws_s3_bucket.tf_course: Still creating... [20s elapsed]
aws_s3_bucket.tf_course: Still creating... [30s elapsed]
aws_s3_bucket.tf_course: Creation complete after 33s [id=sajalsood1995]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Check through windows powershell as well:



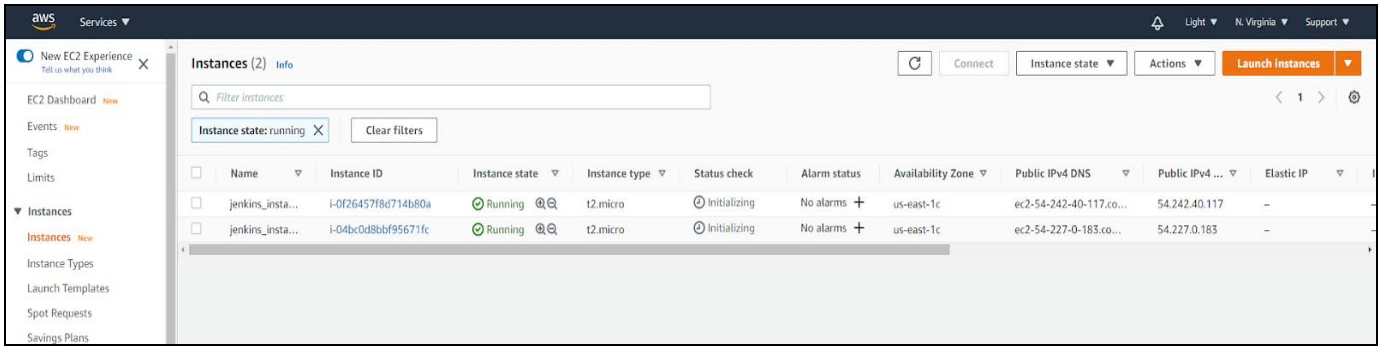
```
aws_security_group.security_jenkins_port: Refreshing state... [id=sg-06a6f329936faa8ad]
aws_instance.myfirstInstance[0]: Refreshing state... [id=i-0f26457f8d714b80a]
aws_instance.myfirstInstance[1]: Refreshing state... [id=i-04bc0d8bbf95671fc]
aws_s3_bucket.tf_course: Refreshing state... [id=sajalsood1995]

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_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.main will be created
  + resource "aws_vpn_connection" "main" {
      + arn                            = (known after apply)
      + customer_gateway_configuration = (known after apply)
      + customer_gateway_id            = (known after apply)
      + id                             = (known after apply)
      + routes                         = (known after apply)
      + static_routes_only             = true
      + transit_gateway_attachment_id  = (known after apply)
      + tunnel1_address                = (known after apply)
      + tunnel1_bgp_asn                = (known after apply)
      + tunnel1_bgp_holdtime           = (known after apply)
      + tunnel1_cgw_inside_address     = (known after apply)
      + tunnel1_inside_cidr            = (known after apply)
      + tunnel1_preshared_key          = (sensitive value)
      + tunnel1_vgw_inside_address     = (known after apply)
      + tunnel2_address                = (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)
    }
```



```
      + 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: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_customer_gateway.customer_gateway: Creating...
aws_vpc.vpc: Creating...
aws_vpc.vpc: Still creating... [10s elapsed]
aws_customer_gateway.customer_gateway: Still creating... [10s elapsed]
aws_vpc.vpc: Creation complete after 14s [id=vpc-005dde8095a1ba862]
aws_vpn_gateway.vpn_gateway: Creating...
aws_customer_gateway.customer_gateway: Creation complete after 15s [id=cgw-0df41170dfde895f6]
aws_vpn_gateway.vpn_gateway: Still creating... [10s elapsed]
aws_vpn_gateway.vpn_gateway: Still creating... [20s elapsed]
aws_vpn_gateway.vpn_gateway: Creation complete after 25s [id=vgw-08a19b921c69b9b76]
aws_vpn_connection.main: Creating...
aws_vpn_connection.main: Still creating... [10s elapsed]
aws_vpn_connection.main: Still creating... [20s elapsed]
aws_vpn_connection.main: Still creating... [30s elapsed]
aws_vpn_connection.main: Still creating... [40s elapsed]
aws_vpn_connection.main: Still creating... [50s elapsed]
aws_vpn_connection.main: Still creating... [1m0s elapsed]
aws_vpn_connection.main: Still creating... [1m10s elapsed]
aws_vpn_connection.main: Still creating... [1m20s elapsed]
aws_vpn_connection.main: Still creating... [1m30s elapsed]
aws_vpn_connection.main: Still creating... [1m40s elapsed]
aws_vpn_connection.main: Still creating... [1m50s elapsed]
aws_vpn_connection.main: Still creating... [2m0s elapsed]
aws_vpn_connection.main: Still creating... [2m10s elapsed]
aws_vpn_connection.main: Still creating... [2m20s elapsed]
aws_vpn_connection.main: Still creating... [2m30s elapsed]
aws_vpn_connection.main: Still creating... [2m40s elapsed]
aws_vpn_connection.main: Still creating... [2m50s elapsed]
aws_vpn_connection.main: Still creating... [3m0s elapsed]
aws_vpn_connection.main: Still creating... [3m10s elapsed]
aws_vpn_connection.main: Still creating... [3m20s elapsed]
aws_vpn_connection.main: Still creating... [3m30s elapsed]
aws_vpn_connection.main: Still creating... [3m40s elapsed]
aws_vpn_connection.main: Still creating... [3m50s elapsed]
aws_vpn_connection.main: Still creating... [4m0s elapsed]
aws_vpn_connection.main: Still creating... [4m10s elapsed]
aws_vpn_connection.main: Still creating... [4m20s elapsed]
aws_vpn_connection.main: Still creating... [4m30s elapsed]
aws_vpn_connection.main: Still creating... [4m40s elapsed]
aws_vpn_connection.main: Still creating... [4m50s elapsed]
aws_vpn_connection.main: Still creating... [5m0s elapsed]
aws_vpn_connection.main: Creation complete after 5m8s [id=vpn-06042822b8697e55a]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
```

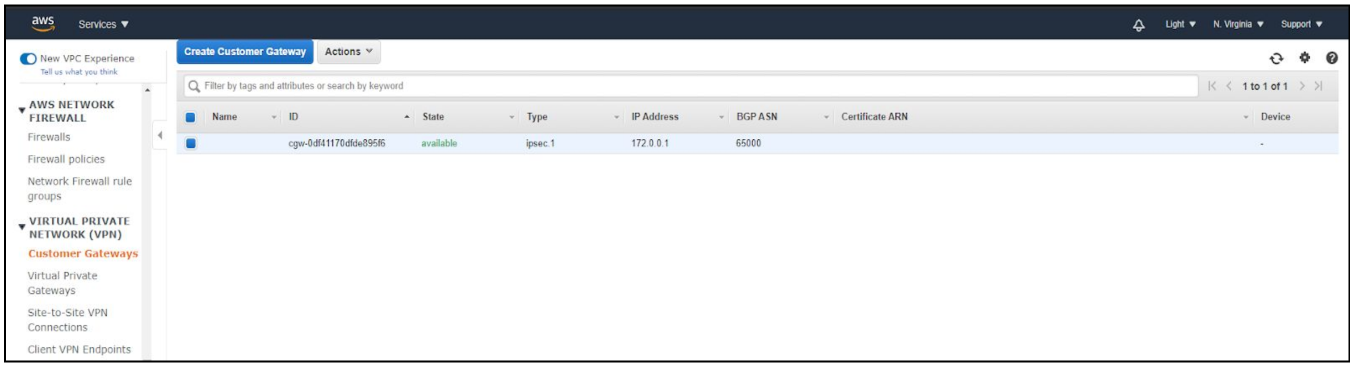
Now, visit your aws management console and see:

- 2 EC2 instances have been created
- VPN is created
- S3 bucket is created:

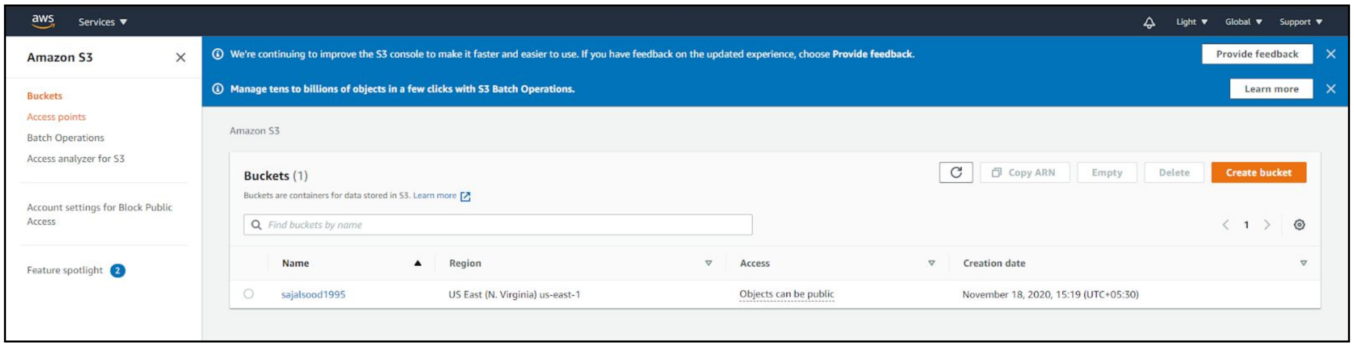


Instances (2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
jenkins_insta...	i-0f26457f8d714b80a	Running	t2.micro	Initializing	No alarms	us-east-1c	ec2-54-242-40-117.co...	54.242.40.117	–
jenkins_insta...	i-04bc0d8bbf95671fc	Running	t2.micro	Initializing	No alarms	us-east-1c	ec2-54-227-0-183.co...	54.227.0.183	–



Name	ID	State	Type	IP Address	BGP ASN	Certificate ARN	Device
	cgw-0df41170dfde895f6	available	ipsec.1	172.0.0.1	65000		-



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**.

Manage tens to billions of objects in a few clicks with S3 Batch Operations.

Buckets (1)

Buckets are containers for data stored in S3. Learn more

Name	Region	Access	Creation date
sajalsood1995	US East (N. Virginia) us-east-1	Objects can be public	November 18, 2020, 15:19 (UTC+05:30)