Name = Devendra Sanjay Sutar
Batch = jan 15.

Terraform Task 1

1 **Make a note on:**

a. **What is Terraform?**
b. **Why Terraform?**
c. **Benefits of Terraform**

a) What is Terraform ?

i. Terraform is an open-source Infrastructure as Code (IaC) software tool.
ii. Terraform configuration language known as HashiCorp Configuration Language (HCL) or optionally JSON.
iii. Terraform is support Multicloud i.e AWS, AZURE, GCP.
iv. Its consist a different block it , it support different cloud providers, so we need to tell them in block which for service .
v. There are three blocks Terraform , provider and Resource.
vi. We are define a version in terraform block , we are define cloud service a which you want to use in the provider block i.e AWS ,AZURE, GCP , we are define a resources as per requirement in the resource block
vii. There are some commands in terraform we are using while creating a resources
    1. terraform init – it will initialize a terraform and also ill create a one file terraform.tfstate.
    2. terraform plan – it will show us a preview of our resources which we are mentaioned in the terraform code.
    3. terraform apply – it will start creating a resources.
    4. Terraform destroy – it will destroy all resources .
viii. Terraform we can used through server of cloud provider and also from our local machine
ix. So we no need to use of portal to launch resources .
x. File name must be with extention (.tf) example – main.tf

b) Why Terraform?

i. Terraform ensures consistent application of infrastructure changes. Applying the same configuration multiple times will always result in the same infrastructure state.
ii. Terraform supports a wide range of providers, including major cloud platforms (AWS, Azure, Google Cloud).
iii. The terraform plan allows users to preview changes before applying them, providing a clear understanding of what will happen.
iv. In one file we are settled all code of creating resources , in only one command all resources are creating at one time parallel.
v. No need to of portal of cloud providers.

c) Benefits of Terraform ?

i. If we are doing work through coding by terraform . so benefit is reusability.
ii. Applying the same configuration multiple times results in the same infrastructure state, reducing the risk of errors.
iii. Supports a wide range of cloud providers (AWS, Azure, Google Cloud).
iv. Users define the desired state of infrastructure, and Terraform figures out the necessary steps to achieve it.
v. Automates the creation, update, and deletion of infrastructure resources, reducing manual work and human error.
vi. Extensive documentation and an active community make it easier to find solutions and get help.
vii. By defining and managing infrastructure as code, Terraform helps optimize resource usage and reduce costs.

Name = Devendra Sanjay Sutar
Batch = jan 15. **Terraform Task 1**

2 **Launch two EC2 instances with names as “myapp-1” and “myapp-2” using Amazon-Linux OS in ‘ap-south-1’ region.**

```
  GNU nano 7.2                                                    main.tf
terraform {
  required_version = "~>1.1"
  required_providers {
    aws = {
        version = "~>3.1"
    }
  }
}
provider "aws" {
    access_key = var.access_key
    secret_key = var.secret_key
    region = var.region_name
}
resource "aws_instance" "myec2" {
    ami = var.ami_id
    instance_type = "t2.micro"
    key_name = aws_key_pair.tf-key-pair.key_name
    tags = {
      Name = "myapp-1"
    }
}
resource "aws_instance" "yourec2" {
    ami = var.ami_id
    instance_type = "t2.micro"
    key_name = aws_key_pair.tf-key-pair.key_name
    tags = {
      Name = "myapp-2"
    }
}
resource "aws_key_pair" "tf-key-pair" {
  key_name   = "tf-key-pair"
  public_key = tls_private_key.rsa.public_key_openssh
}

resource "tls_private_key" "rsa" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "local_file" "tf-key" {
  content  = tls_private_key.rsa.private_key_pem
  filename = "${path.module}/tf-key.pem"
}
```

```
^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute   ^C Location   M-U Undo      M-A Set Mark   M-] To Bracket  M-Q Previous   ^B Back
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line  M-E Redo      M-6 Copy       ^Q Where Was    M-W Next       ^F Forward
```

- **Main.tf file we declare to two EC2 instances with names as “myapp-1” and “myapp-2”**

```
ubuntu@ip-172-31-18-52:~/task$ ls
main.tf  terraform.tfvars  variable.tf
ubuntu@ip-172-31-18-52:~/task$
```

- **We created variable.tf and terrafrom.tfvars files also**

```
ubuntu@ip-172-31-18-52:~/task$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Finding hashicorp/aws versions matching "~> 3.1"...
- Finding latest version of hashicorp/tls...
- Installing hashicorp/local v2.5.1...
- Installed hashicorp/local v2.5.1 (signed by HashiCorp)
- Installing hashicorp/aws v3.76.1...
- Installed hashicorp/aws v3.76.1 (signed by HashiCorp)
- Installing hashicorp/tls v4.0.5...
- Installed hashicorp/tls v4.0.5 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
ubuntu@ip-172-31-18-52:~/task$ nano main.tf
```

- **We successfully initialize a terraform.**

Name = Devendra Sanjay Sutar

Batch = jan 15. **Terraform Task 1**

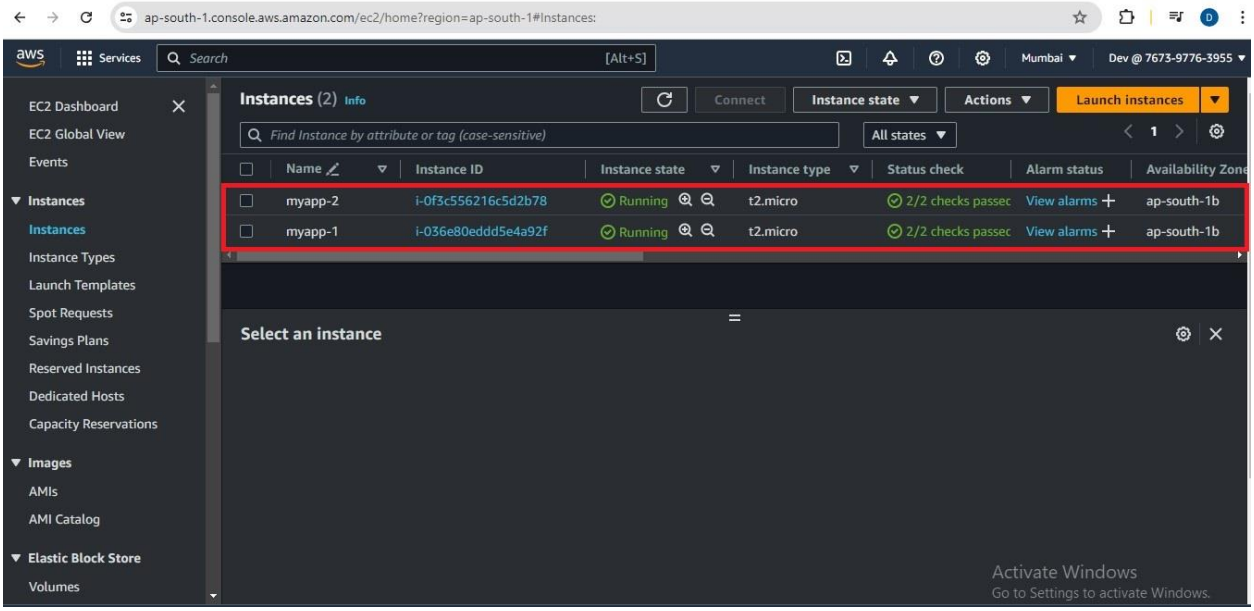
```
  }

  # aws_key_pair.tf-key-pair will be created
  + resource "aws_key_pair" "tf-key-pair" {
      + arn             = (known after apply)
      + fingerprint     = (known after apply)
      + id              = (known after apply)
      + key_name        = "tf-key-pair"
      + key_name_prefix = (known after apply)
      + key_pair_id     = (known after apply)
      + public_key      = (known after apply)
      + tags_all        = (known after apply)
    }

  # local_file.tf-key will be created
  + resource "local_file" "tf-key" {
      + content              = (sensitive value)
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "./tf-key.pem"
      + id                   = (known after apply)
    }

  # tls_private_key.rsa will be created
  + resource "tls_private_key" "rsa" {
      + algorithm                     = "RSA"
      + ecdsa_curve                   = "P224"
      + id                            = (known after apply)
      + private_key_openssh           = (sensitive value)
      + private_key_pem               = (sensitive value)
      + private_key_pem_pkcs8         = (sensitive value)
      + public_key_fingerprint_md5    = (known after apply)
      + public_key_fingerprint_sha256 = (known after apply)
      + public_key_openssh            = (known after apply)
      + public_key_pem                = (known after apply)
      + rsa_bits                      = 4096
    }

Plan: 5 to add, 0 to change, 0 to destroy.

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.
ubuntu@ip-172-31-18-52:~/task$
```

- **We successfully plan using terraform plan .**

```
ubuntu@ip-172-31-18-52: ~/task
        + "Name" = "myapp-2"
      }
      + tags_all                           = {
        + "Name" = "myapp-2"
      }
      + tenancy                            = (known after apply)
      + user_data                          = (known after apply)
      + user_data_base64                   = (known after apply)
      + vpc_security_group_ids             = (known after apply)
    }

  # aws_key_pair.tf-key-pair will be created
  + resource "aws_key_pair" "tf-key-pair" {
      + arn             = (known after apply)
      + fingerprint     = (known after apply)
      + id              = (known after apply)
      + key_name        = "tf-key-pair1"
      + key_name_prefix = (known after apply)
      + key_pair_id     = (known after apply)
      + public_key      = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDGphQRe5wd2D8EkCvv//HbzuKMGPgBe0Ewg4IN6ZxO1GOQMxcBt6k3OvahEBsOFhw/zSwS/bT+Lib/KpZgeEcMBgcFd0Cg0kWSENk+jf3CF0ISSJV1FpCrZWEvFa2t4
5PzSLbhXI8qQeQTZ1OpV6cm4jngZeYB1JuNJApzjVkZgqyer1vSbd8dyP3mwzokQ9fXkaOWuV9Of9cqQDb+kjdH05uBuhs2Ueudi2CFtsSxZG8M0/sHPTP1yXGvwQ7pL5F1uW8YC58wBhRT0NQ3FVOk82ycOiKnBgQDk3+WssuCV1qbum3EUKNwROCqBYCov
urnkZUHsekQJdD9mn5fUewJrpOc+jd23JOXuJJVcOgv4GvbGPZ+oFLyLzNJTOzW5RFTLqzHqN1/P4SQ12hxOKhZgZATH1OIWe/qdgxCSbjKib/IefW0vVKs9362bW688g1OUQQeCU6qOxjwwJ4hYWL6nYy6h0eIx5vRUFHjdYu7hKGhMwCN1A1f1ha4m2XNg
BSiE3v9V02p5MUNwsaGmAsAYQJ0fsPgQqTsx9gpfPXP2/wbHbGuw9c4ayysTswYHs8hnso3UeFvvg+iTbbxi72tSzPNwzSLjYDj6s9t6fPCCxZc3g91/7uLNPyqIR2gp/kX8p+QktOUDsCFA+SzVyuOqJQPd23uSIowIOkmx5s1Q==
      + tags_all        = (known after apply)
    }

Plan: 3 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_key_pair.tf-key-pair: Creating...
aws_key_pair.tf-key-pair: Creation complete after 2s [id=tf-key-pair1]
aws_instance.yourec2: Creating...
aws_instance.myec2: Creating...
aws_instance.yourec2: Still creating... [10s elapsed]
aws_instance.myec2: Still creating... [10s elapsed]
aws_instance.yourec2: Still creating... [20s elapsed]
aws_instance.myec2: Still creating... [20s elapsed]
aws_instance.yourec2: Still creating... [30s elapsed]
aws_instance.myec2: Still creating... [30s elapsed]
aws_instance.yourec2: Creation complete after 30s [id=i-0f3c556216c5d2b78]
aws_instance.myec2: Still creating... [40s elapsed]
aws_instance.myec2: Creation complete after 41s [id=i-036e80eddd5e4a92f]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-18-52:~/task$
```

- **We successfully created a EC2 instances with names as “myapp-1” and “myapp-2”**

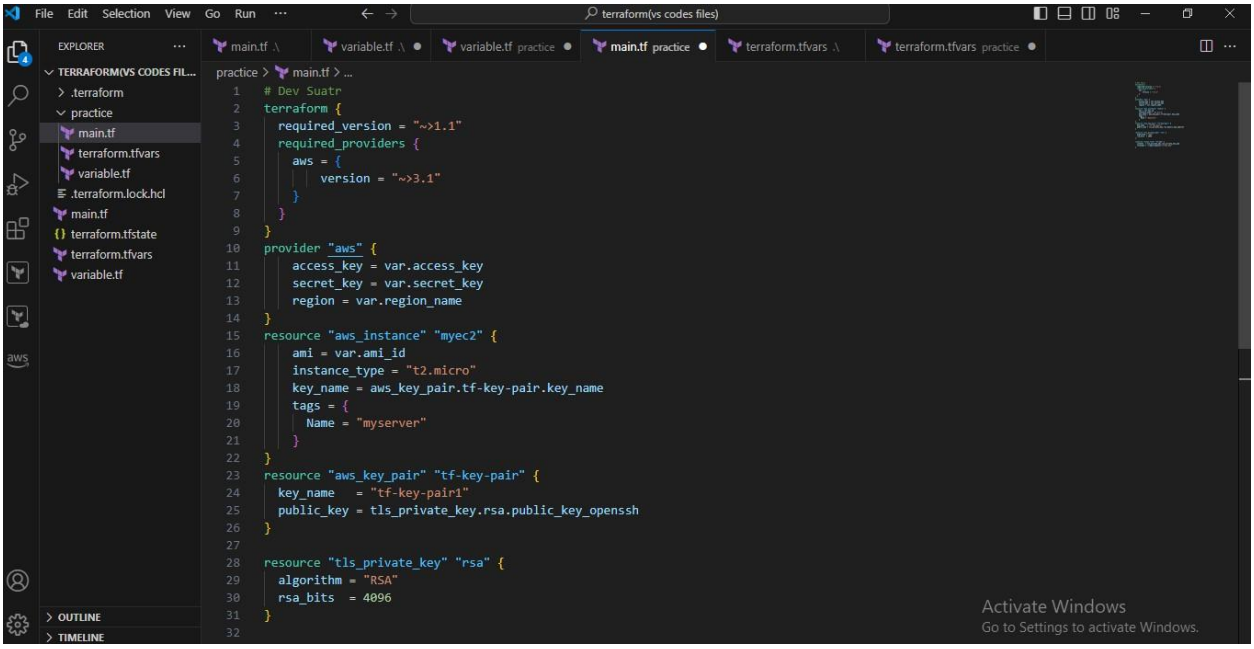
Name = Devendra Sanjay Sutar
Batch = jan 15. **Terraform Task 1**



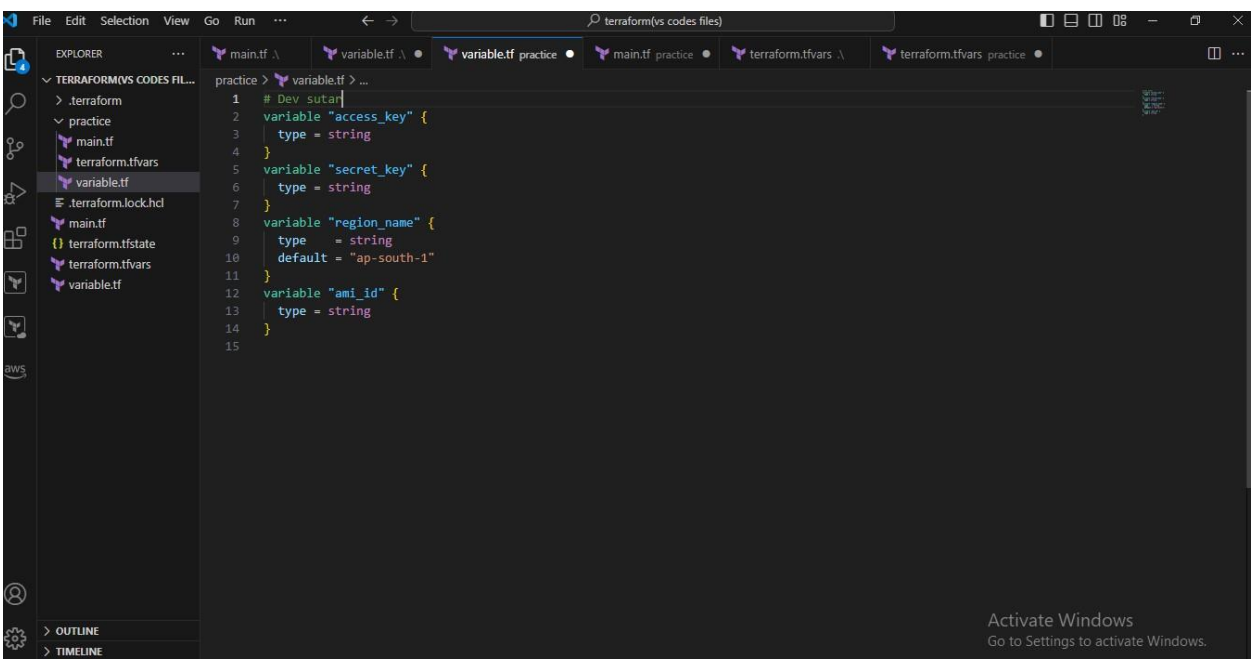
- **Completed.**

Name = Devendra Sanjay Sutar
Batch = jan 15. **Terraform Task 1**

3 **Install Terraform on local machine (Laptop), integrate aws and terraform with VS code. Using VS code launch an EC2 instances with name 'myserver' using Windows OS in 'ap-south-1' region**

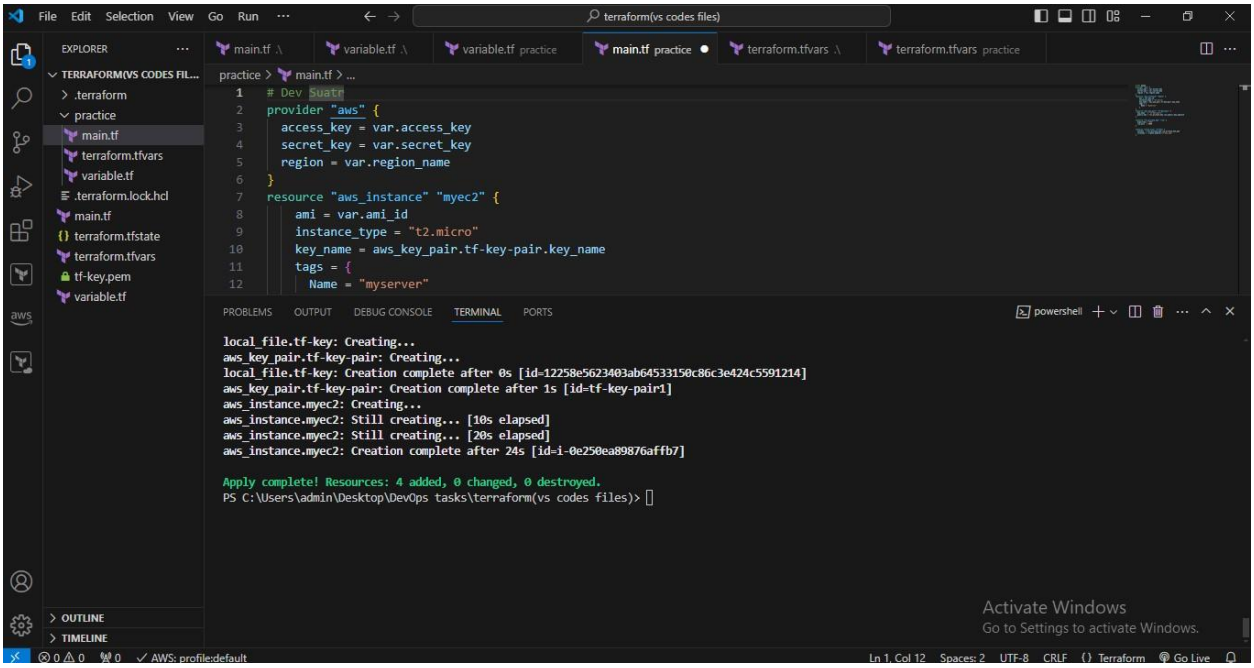


- **Successfully Install Terraform on local machine (Laptop), integrate aws and terraform with VS code. And write code in main.tf file**

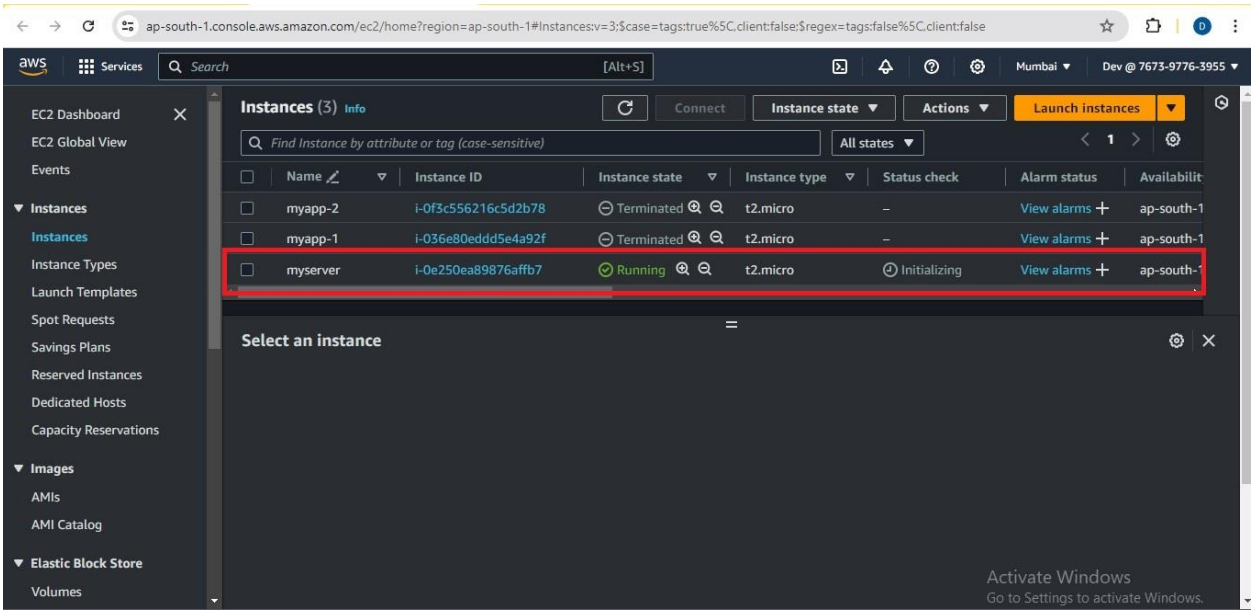


- **Successfully created variable.tf and terraform.tfvars files and declare values in it.**

Name = Devendra Sanjay Sutar
Batch = jan 15. **Terraform Task 1**



- **Successfully terraform apply command executed.**



- **Successfully completed.**