Name: Aditya Kirtane **Div: D15C** **Roll No: 26**

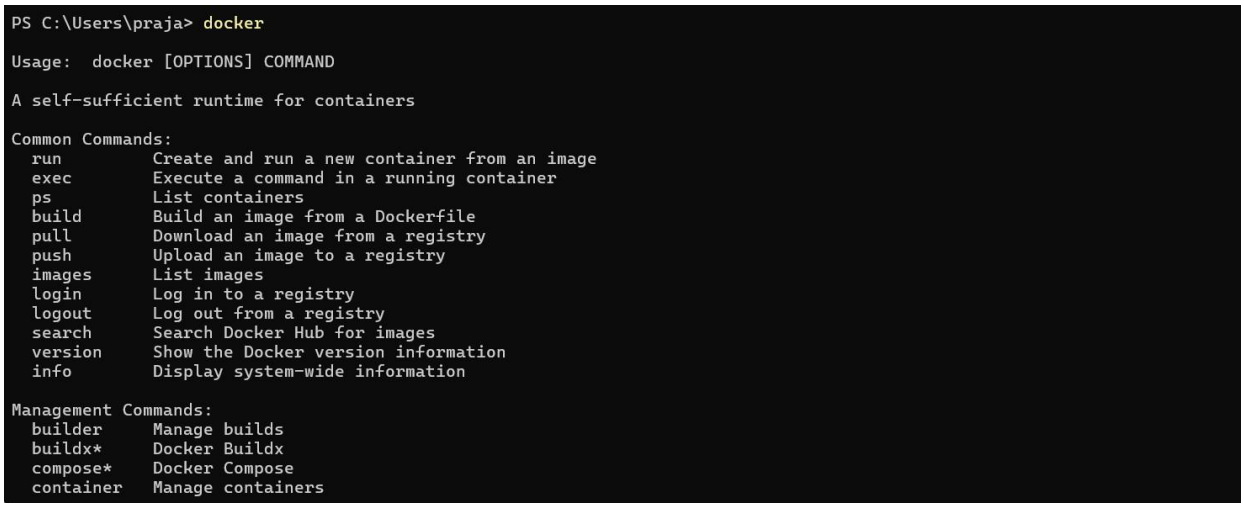
Experiment No: 6

Aim: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform. (S3 bucket or Docker) fdp.

A. Creating docker image using terraform

Step 1: In this experiment ,we need to install docker on our local Machine. Go to https://www.docker.com/ and download the file according to the OS you have.

Open the file and start the installation. After Installation, open your terminal and run 'docker' command. If this is your output, then docker is installed successfully .

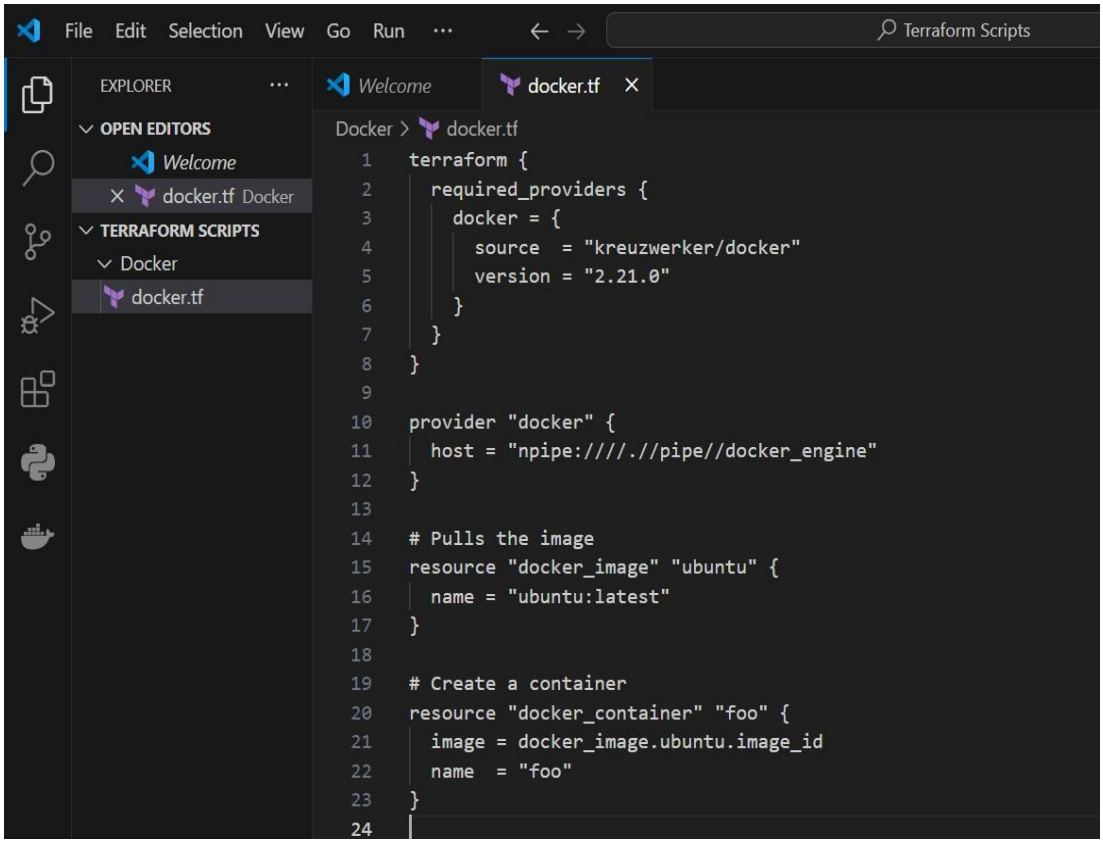


Step 2: Create a Folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file using Atom editor if you are using linux or else use VS code in windows and write the following contents into it to create a Ubuntu Linux container.

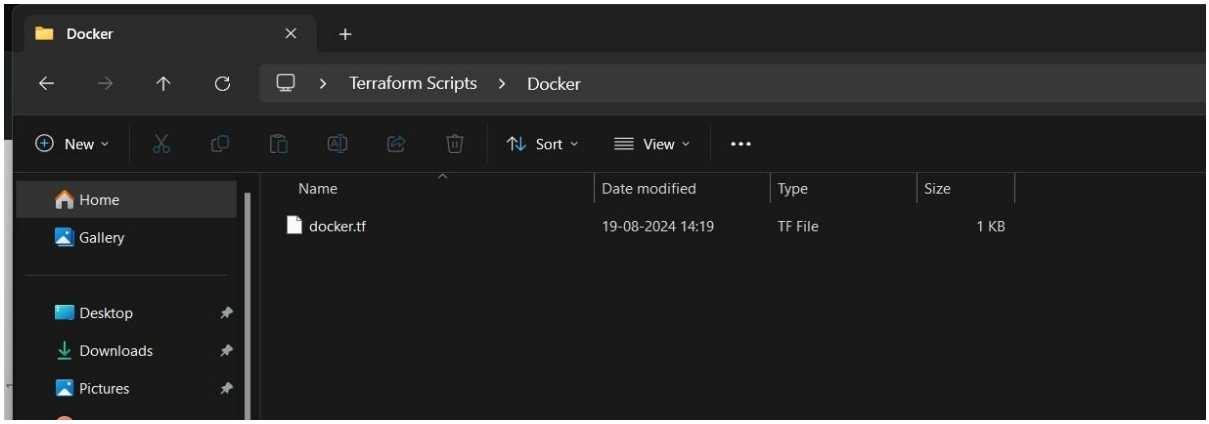
Script:

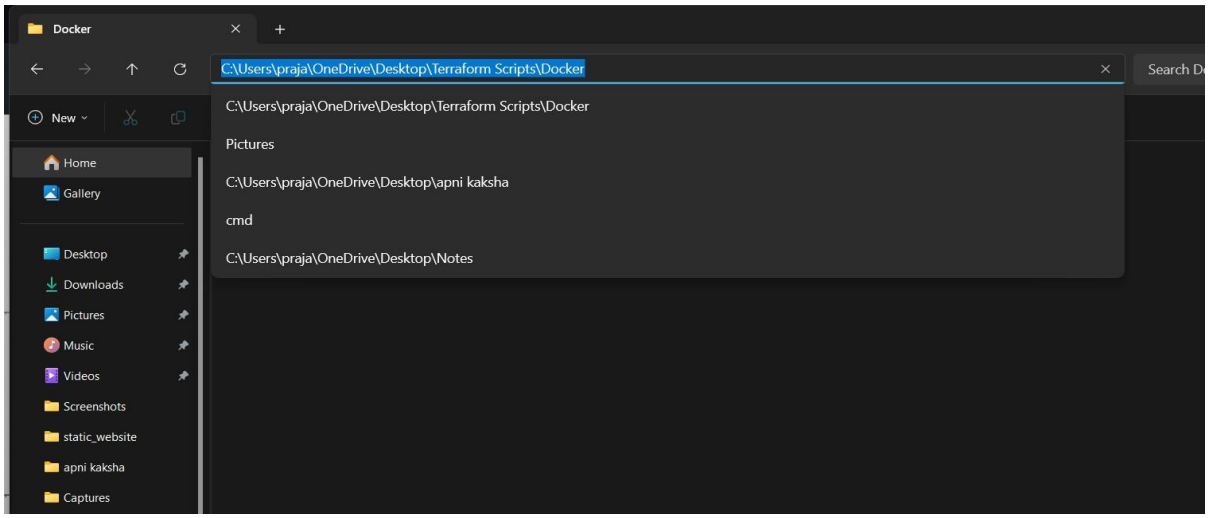
```
terraform
  { required_providers
  {docker = {
     source = "kreuzwerker/docker"
     version = "2.21.0"
   }
 }
```

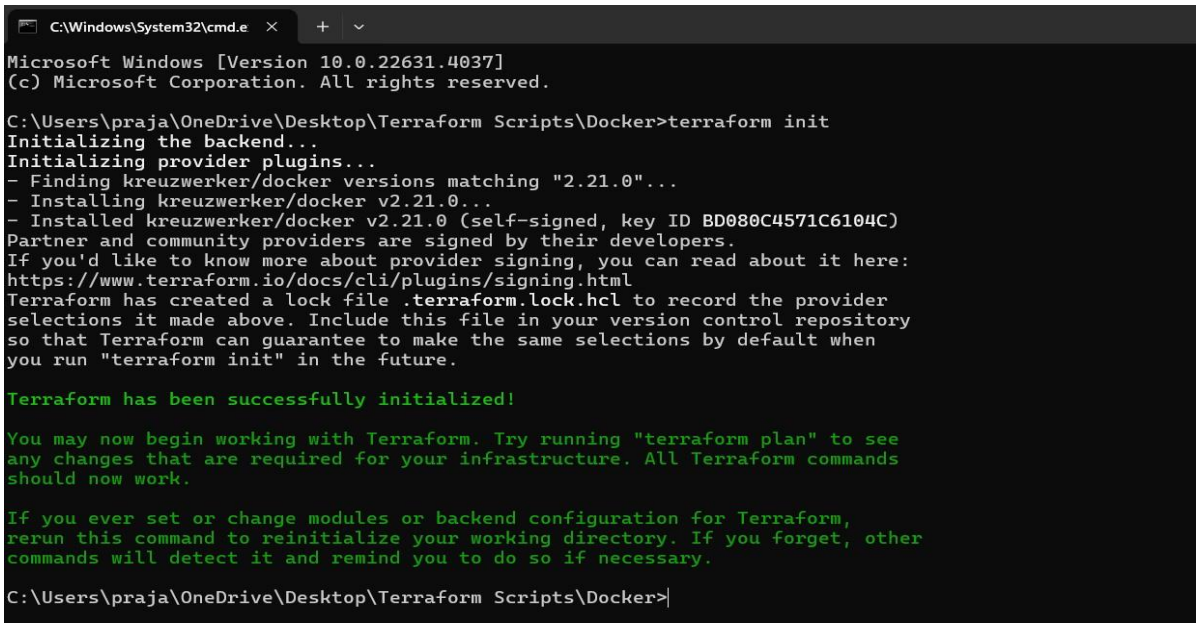
```
}

provider "docker" {
  host = "npipe:////.//pipe//docker_engine" }

# Pulls the image
resource "docker_image" "ubuntu"
   {name = "ubuntu:latest"
 }

# Create a container
resource "docker_container" "foo" {
   image =
  docker_image.ubuntu.image_idname = "foo"
 }
```



Step 3: Execute Terraform Init command to initialize the resources .Now for this go to file manager ->Open Terraform script folder then open Docker folder ->Click on the path of these folder and type cmd this will open Command Prompt window to initialize it in our directory.







Step 4: Execute Terraform plan to see the available resources.

This command helps to get an execution plan and lets us overview changes that are going to happen in your infrastructure.

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>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:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
```

```
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

Plan: 2 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.

C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>
```

Name: Shivam.R.Prajapati **Div: D15C**

Roll No: 41

Step 5: Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration. Using command : “terraform apply”

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>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:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
```

This will ask You to enter a value so Type “Yes”.

```
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

Plan: 2 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

docker_image.ubuntu: Creating...
docker_image.ubuntu: Creation complete after 7s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c25
98aubuntu:latest]
docker_container.foo: Creating...

Error: container exited immediately

  with docker_container.foo,
  on docker.tf line 20, in resource "docker_container" "foo":
  20: resource "docker_container" "foo" {


C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>
```

Name: Shivam.R.Prajapati **Div: D15C**

Roll No: 41

The script that we are using is going to throw an error. Error: container exited immediately .This is because the script used is way too small or took a lot less time to execute. To fix this, we add a line to the code. 'Command = ["sleep", "infinity"]'. This line of code lets docker know to keep the program in sleep mode for an infinite amount of time so that the output can be observed rather than stopping after running immediately.

Do the following changes in the last line of the code as follows to solve the error

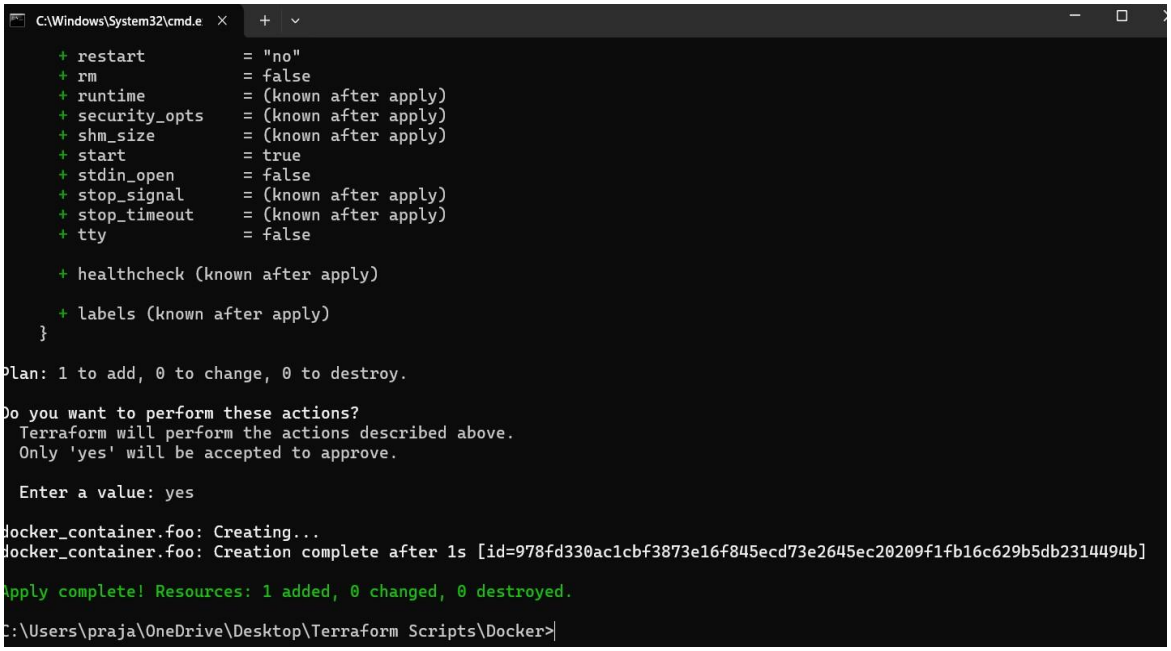
```
# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep","infinity"]
}
```

Now run the command again

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c25
tu:latest]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated wi
following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "infinity",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
```

Name: Shivam.R.Prajapati **Div: D15C**

Roll No: 41



Docker images, Before Executing Apply step:

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>docker images
REPOSITORY                 TAG       IMAGE ID       CREATED        SIZE
docker/welcome-to-docker   latest    c1f619b6477e   9 months ago   18.6MB
```

Docker images, After Executing Apply step:

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>docker images
REPOSITORY                 TAG       IMAGE ID       CREATED        SIZE
ubuntu                     latest    edbfe74c41f8   2 weeks ago    78.1MB
docker/welcome-to-docker   latest    c1f619b6477e   9 months ago   18.6MB
```

Step 6: Now the image is created, if we have to destroy it. For this, we use the '*terraform destroy*' command. Again, this command will ask for a prompt to enter yes, as a confirmation to destroy the image we created.

Type Yes.

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]
docker_container.foo: Refreshing state... [id=978fd330ac1cbf3873e16f845ecd73e2645ec20209f1fb16c629b5db2314494b]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  - destroy

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach            = false -> null
      - command           = [
          - "sleep",
          - "infinity",
        ] -> null
      - cpu_shares        = 0 -> null
      - dns               = [] -> null
      - dns_opts          = [] -> null
      - dns_search        = [] -> null
      - entrypoint        = [] -> null
      - env               = [] -> null
      - gateway           = "172.17.0.1" -> null
      - group_add         = [] -> null
      - hostname          = "978fd330ac1c" -> null
      - id                = "978fd330ac1cbf3873e16f845ecd73e2645ec20209f1fb16c629b5db2314494b" -> null
      - tty               = false -> null
        # (8 unchanged attributes hidden)
    }

  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id          = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
      - image_id    = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - latest      = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - name        = "ubuntu:latest" -> null
      - repo_digest = "ubuntu@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee" -> null
    }

Plan: 0 to add, 0 to change, 2 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

docker_container.foo: Destroying... [id=978fd330ac1cbf3873e16f845ecd73e2645ec20209f1fb16c629b5db2314494b]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:lat
est]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.

C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>
```

Docker images After Executing Destroy step

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>docker images
REPOSITORY                  TAG       IMAGE ID       CREATED        SIZE
docker/welcome-to-docker    latest    c1f619b6477e   9 months ago   18.6MB
```

Thus we have Successfully created the Docker image using terraform in this experiment and have also destroyed it