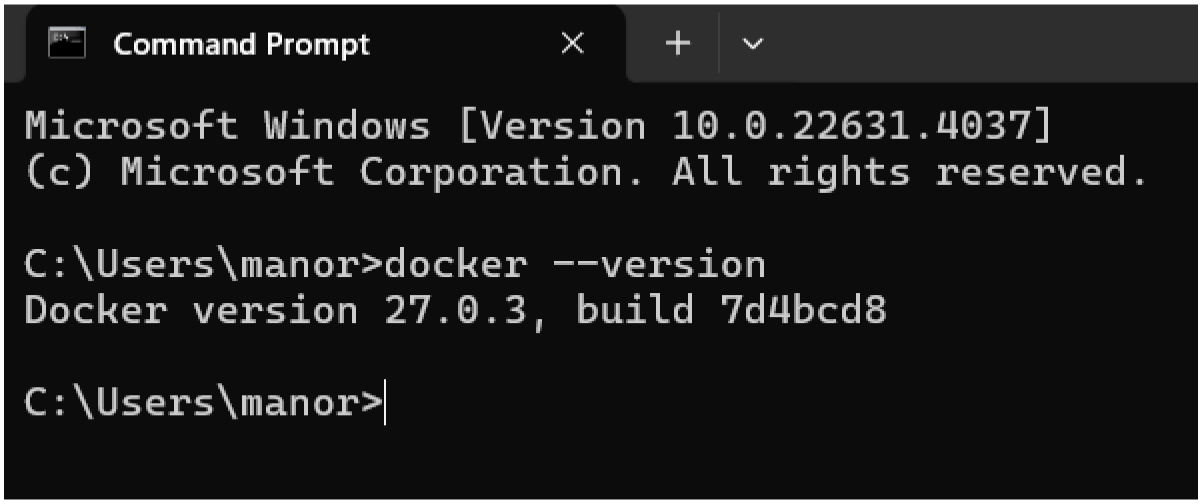
EXPERIMENT-06

MANORATH ITAL
D15A/19

AIM:To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.

Implementation:

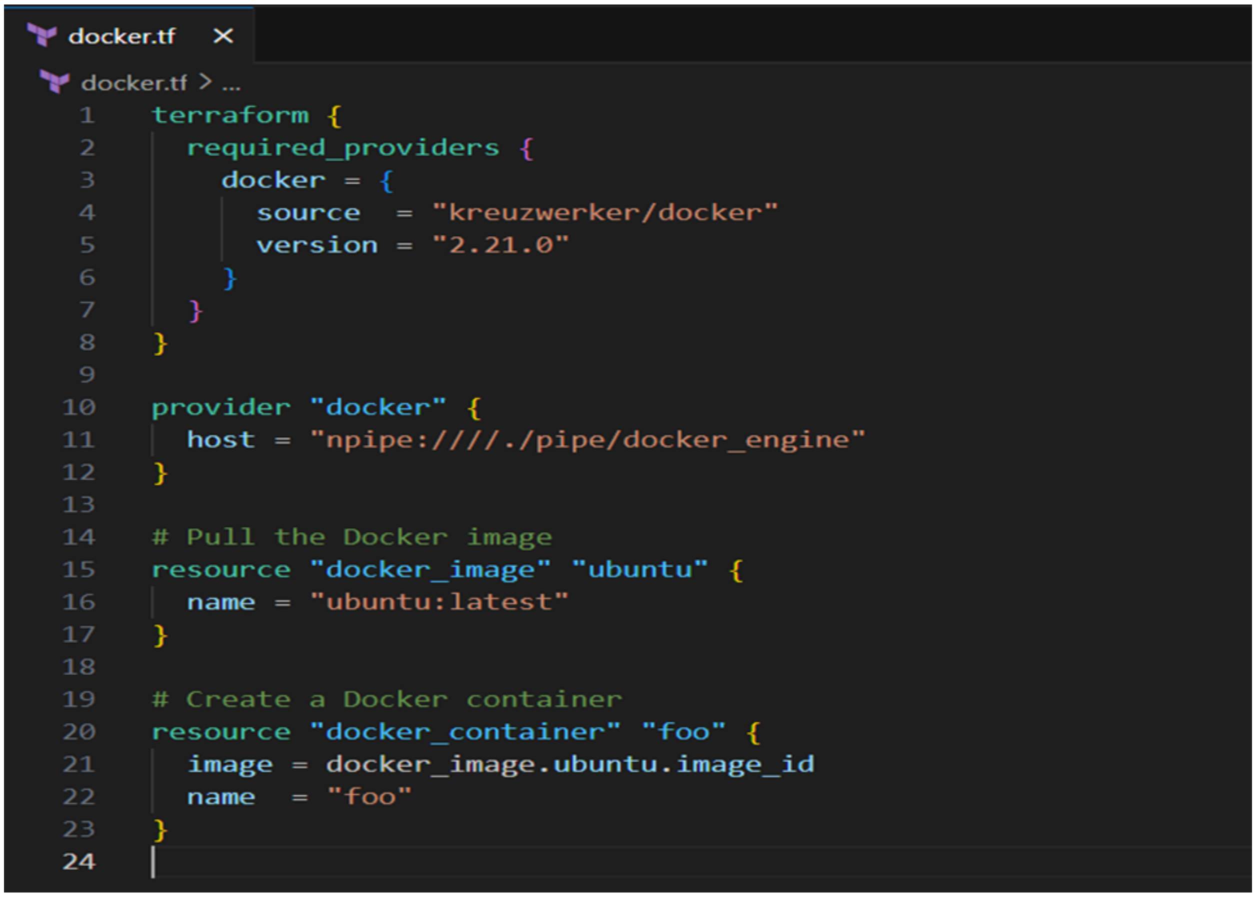
A. Creating docker image using terraform
Prerequisite:
1) Download and Install Docker Desktop from https://www.docker.com/
Step 1: Check the docker functionality



```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\manor>docker --version
Docker version 27.0.3, build 7d4bcd8

C:\Users\manor>
```

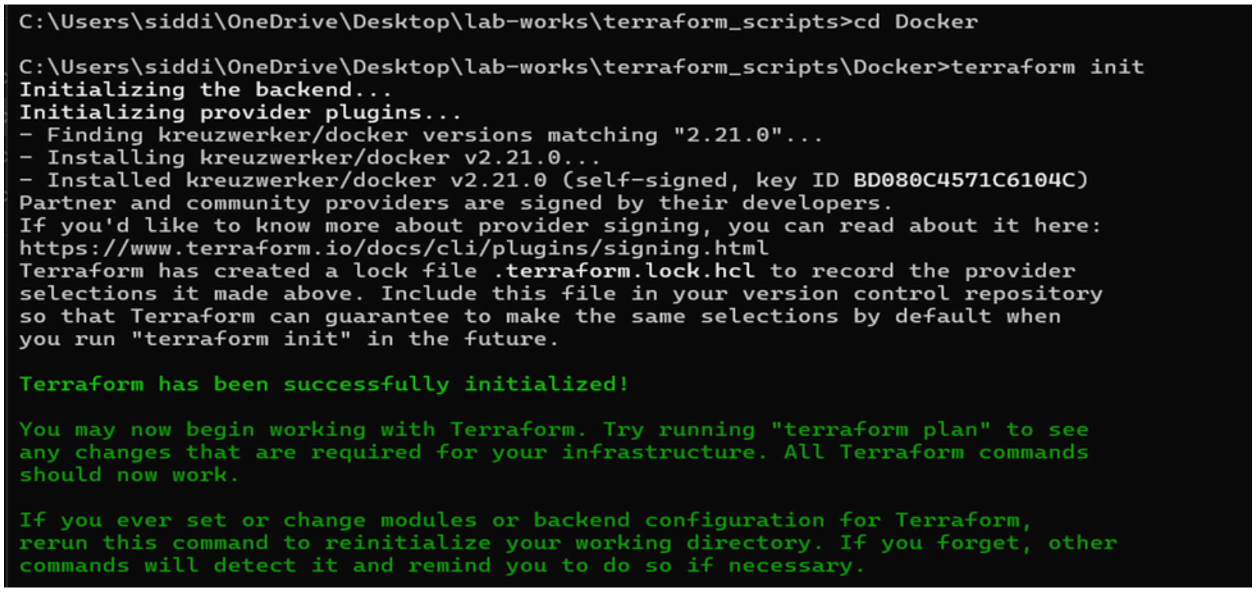
Now, create a folder named ‘Terraform Scripts’ in which we save our different types of
scripts which will be further used in this experiment.
Step 2: Firstly create a new folder named ‘Docker’ in the ‘TerraformScripts’ folder. Then
create a new docker.tf file using Atom editor and write the followingcontents 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"
}
```



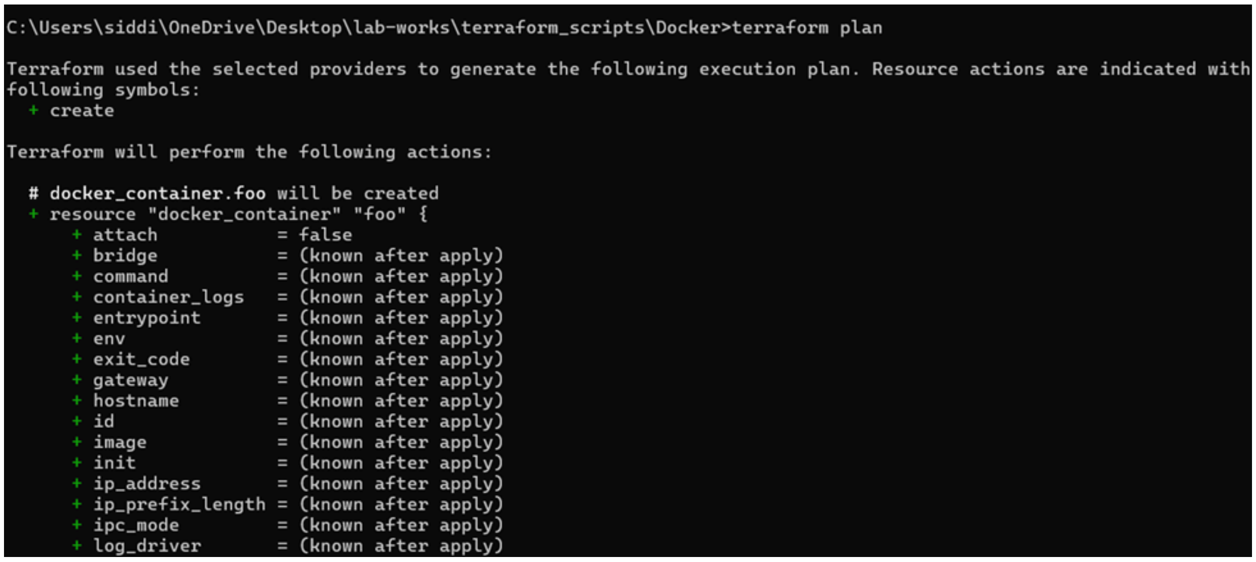
```
docker.tf
docker.tf > ...
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

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

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

# Create a Docker container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
}

```

step2:Terraform init

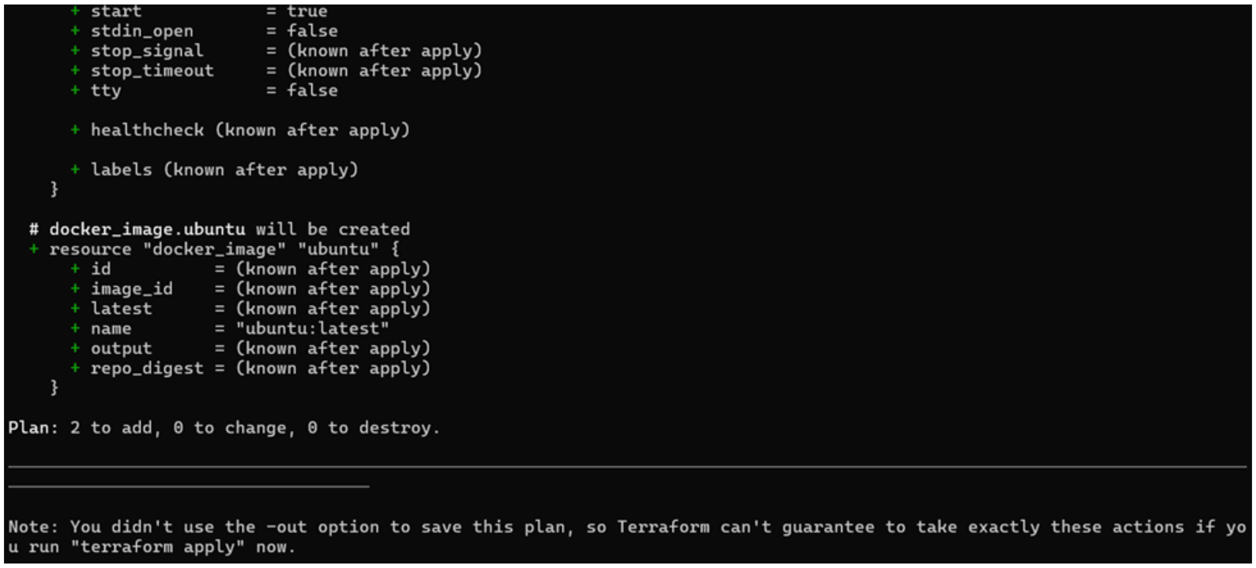


```
C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts>cd Docker

C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
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.
```

3.Terraform plan

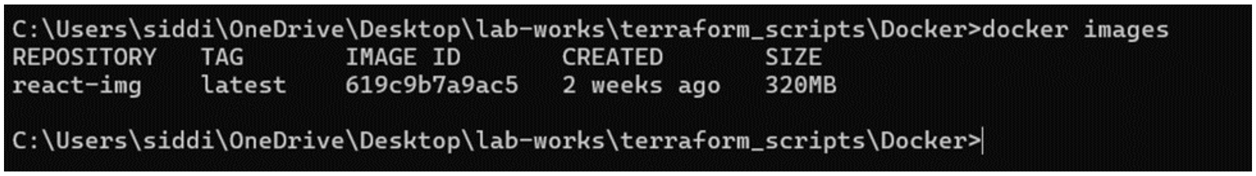


```
C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with
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)
```



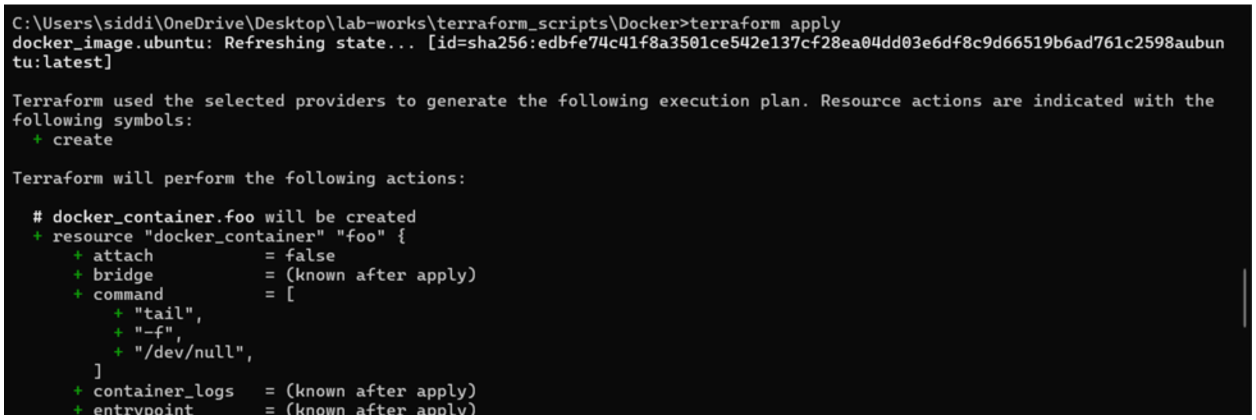
```
      + start             = true
      + stdin_open        = false
      + stop_signal       = (known after apply)
      + stop_timeout      = (known after apply)
      + tty               = false

      + 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 yo
u run "terraform apply" now.
```

4.Check docker images before applying

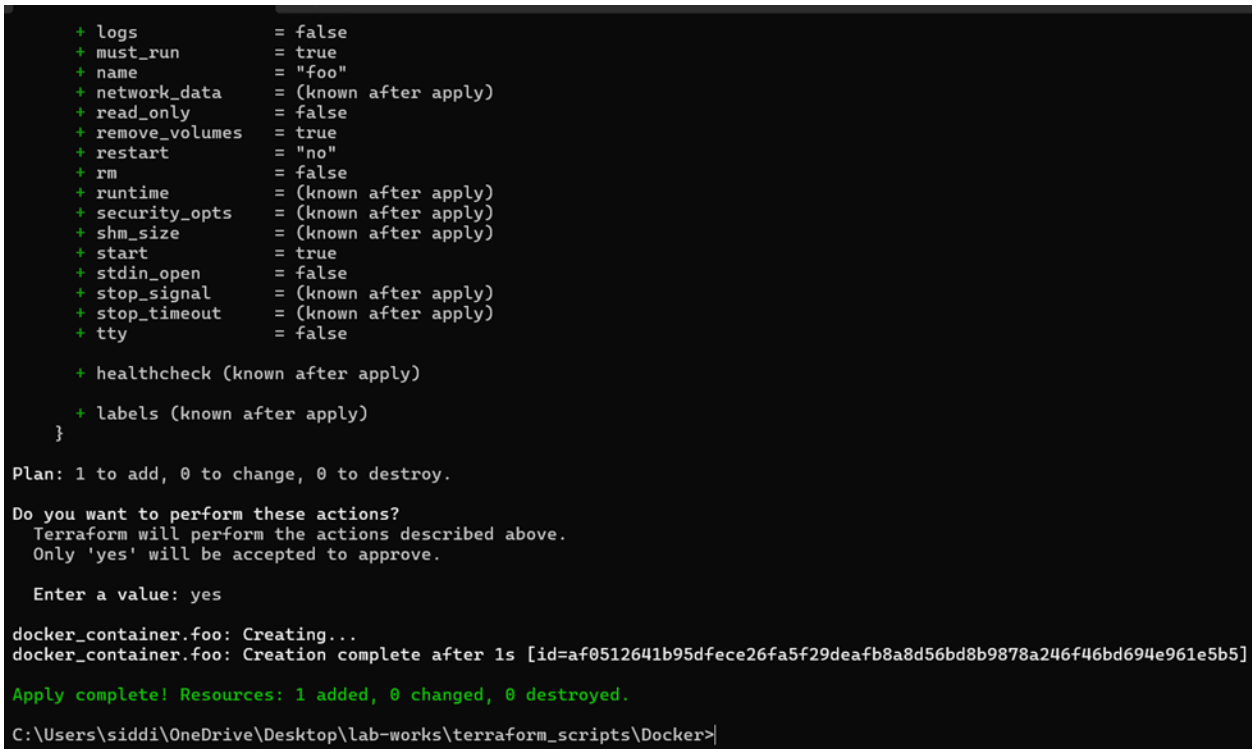


```
C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
react-img    latest    619c9b7a9ac5   2 weeks ago   320MB

C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>
```

5.Terraform apply

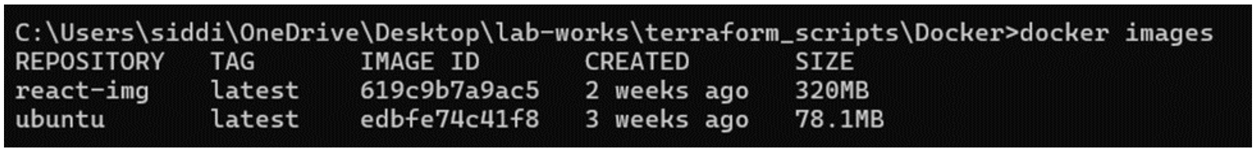


```
C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]

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          = [
          + "tail",
          + "-f",
          + "/dev/null",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
```



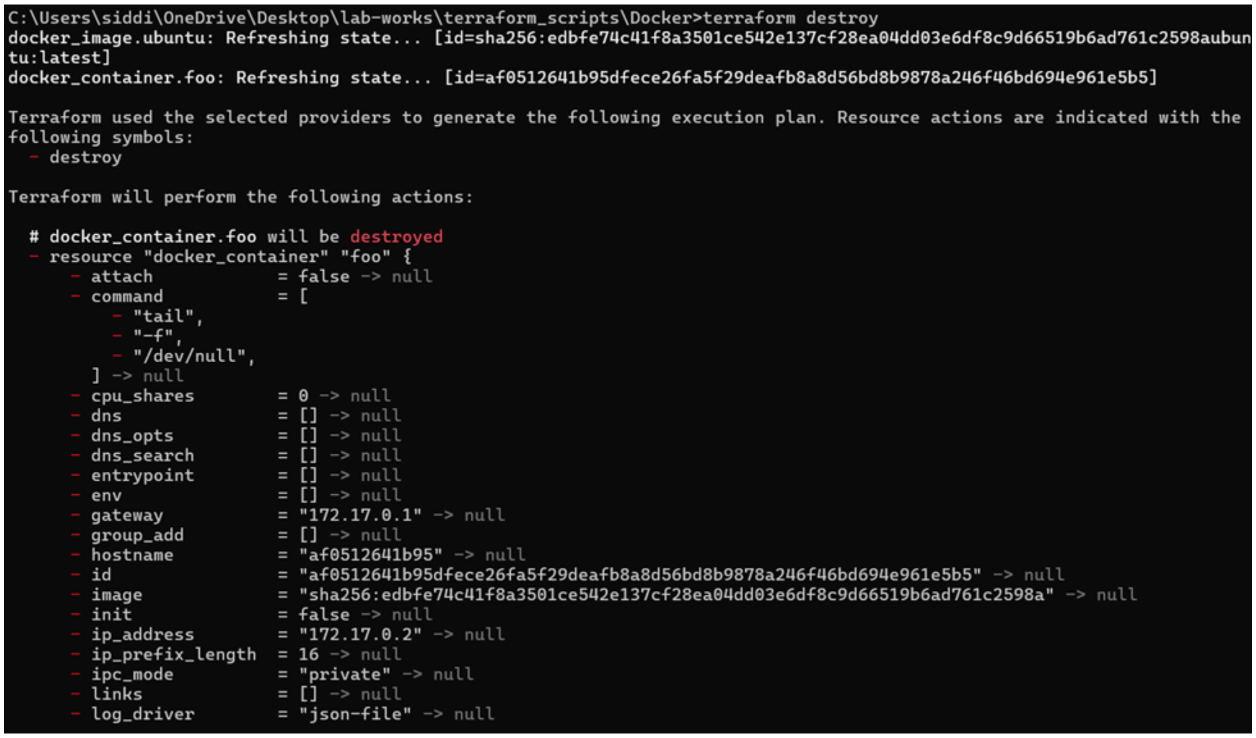
```
      + logs              = false
      + must_run          = true
      + name              = "foo"
      + network_data      = (known after apply)
      + read_only         = false
      + remove_volumes    = true
      + restart           = "no"
      + rm                = false
      + runtime           = (known after apply)
      + security_opts     = (known after apply)
      + 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)
    }

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

docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=af0512641b95dfece26fa5f29deafb8a8d56bd8b9878a246f46bd694e961e5b5]

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

C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>
```

6.Docker images after apply

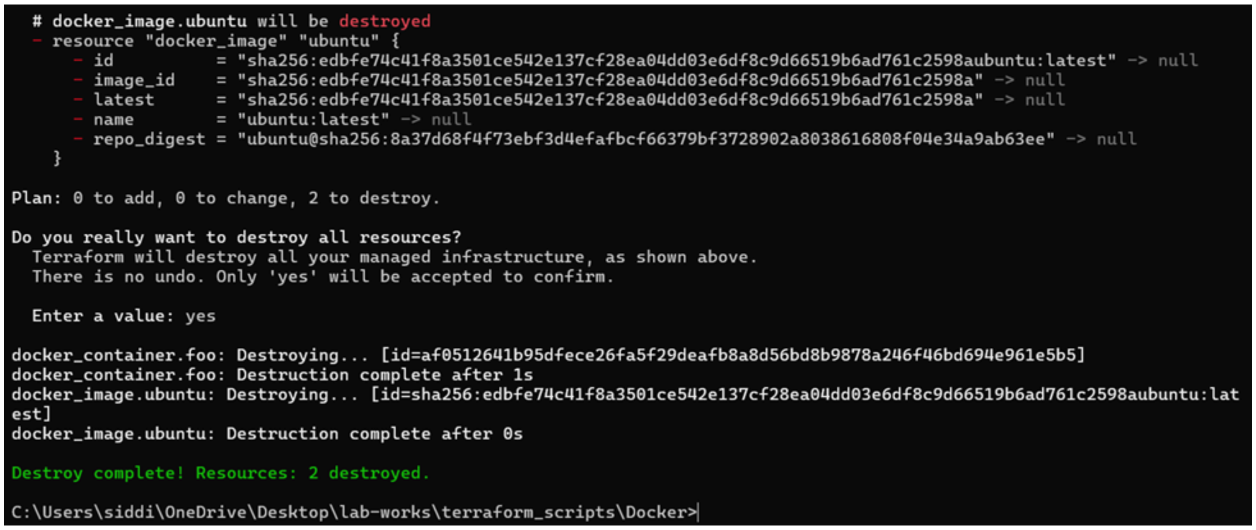


```
C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
react-img    latest    619c9b7a9ac5   2 weeks ago   320MB
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB
```

7.Terraform destroy

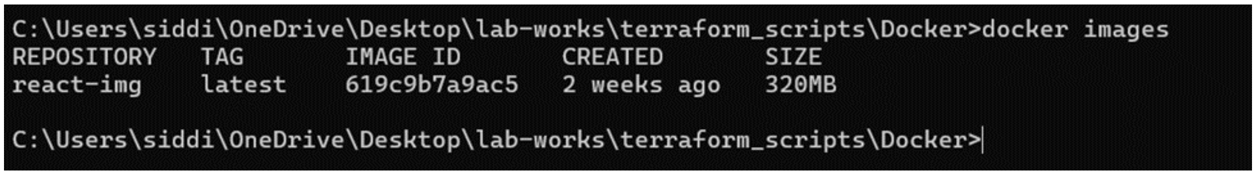


```
C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]
docker_container.foo: Refreshing state... [id=af0512641b95dfece26fa5f29deafb8a8d56bd8b9878a246f46bd694e961e5b5]

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            = [
          - "tail",
          - "-f",
          - "/dev/null",
        ] -> 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           = "af0512641b95" -> null
      - id                 = "af0512641b95dfece26fa5f29deafb8a8d56bd8b9878a246f46bd694e961e5b5" -> null
      - image              = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init               = false -> null
      - ip_address         = "172.17.0.2" -> null
      - ip_prefix_length   = 16 -> null
      - ipc_mode           = "private" -> null
      - links              = [] -> null
      - log_driver         = "json-file" -> null
```



```
  # 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=af0512641b95dfece26fa5f29deafb8a8d56bd8b9878a246f46bd694e961e5b5]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:lat
est]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.

C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>
```

8.Docker images after apply



```
C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
react-img    latest    619c9b7a9ac5   2 weeks ago    320MB

C:\Users\siddi\OneDrive\Desktop\lab-works\terraform_scripts\Docker>
```