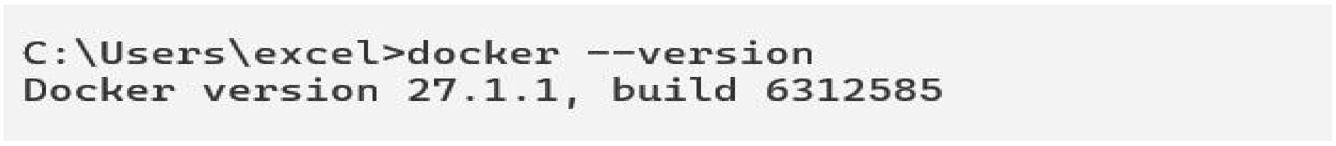
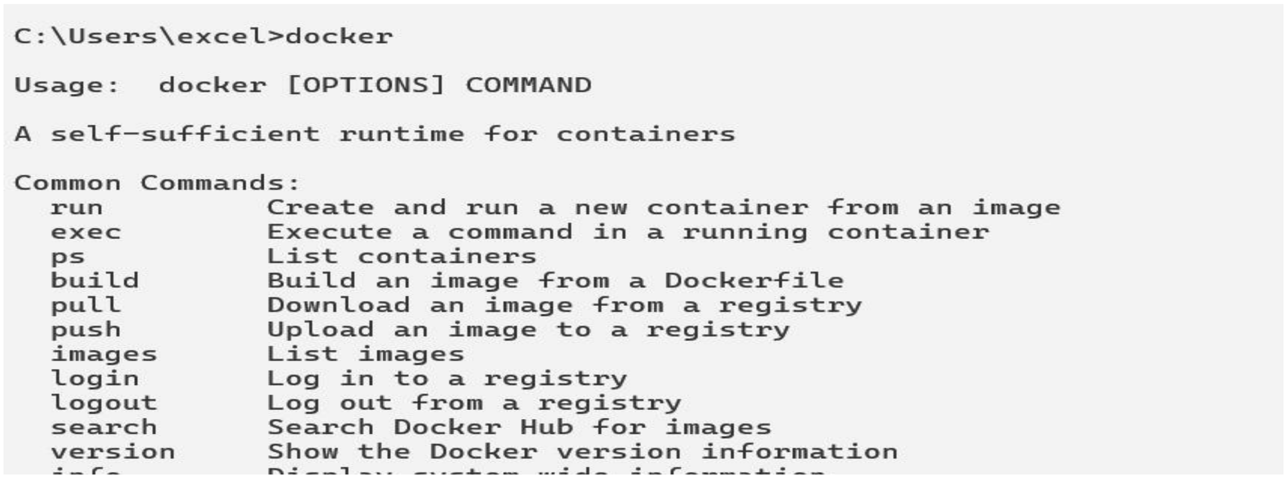
# Experiment No:6

**Implementation:**
**A. Creating docker image using terraform**

Prerequisites:
1. Download and install Docker Desktop from
Website: https://www.docker.com.



```
C:\Users\excel>docker --version
Docker version 27.1.1, build 6312585
```



```
C:\Users\excel>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
```

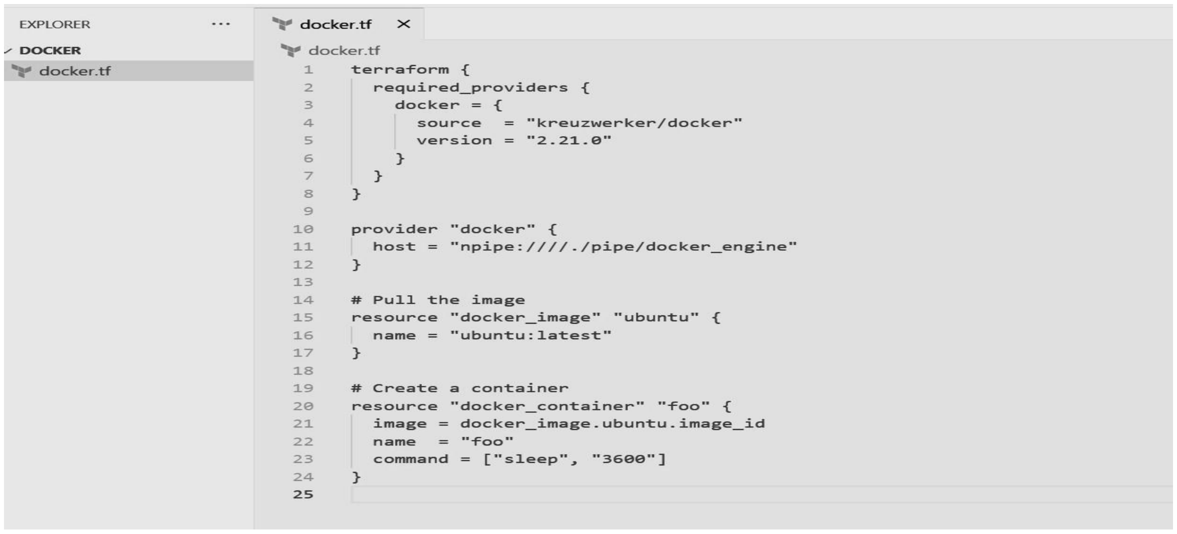
Step 1:To Verify Docker Functionality

1. Create a folder named `Terraform Scripts` to store various scripts for this experiment.

Step 2:To Set Up Terraform Configuration

1. Inside the `Terraform Scripts` folder, create a new folder named `Docker`.
2. Within the `Docker` folder, create a file named `docker.tf` using Atom editor and insert the following content to configure an Ubuntu Linux container:
terraform {

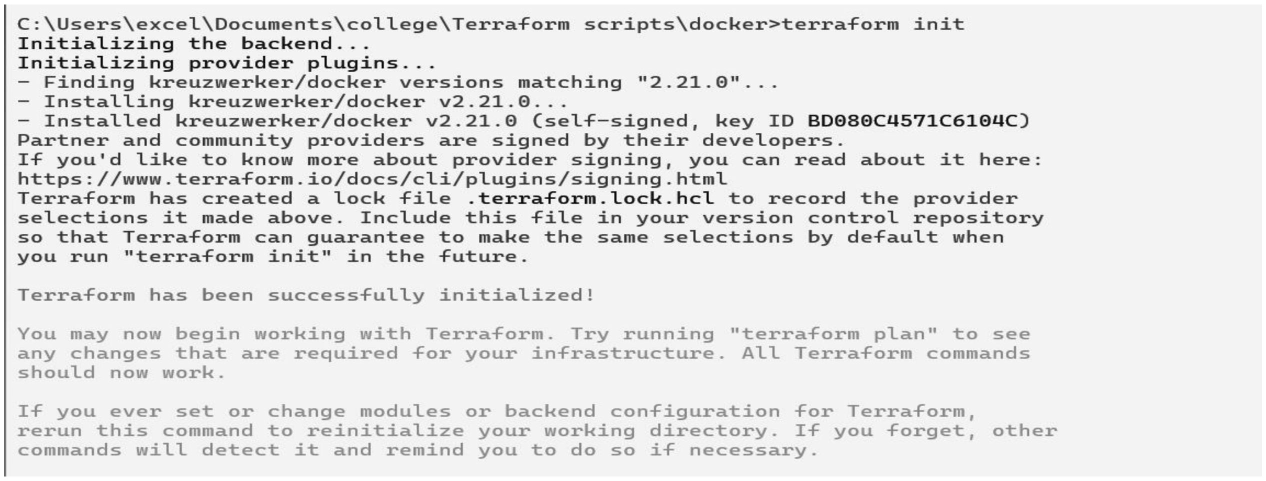
```
  required_providers
    { docker = {
      source = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

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

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

# Create a container
resource "docker_container" "foo"
  { image =
  docker_image.ubuntu.image_id name =
  "foo"
  command = ["sleep", "3600"]
}
```



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

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

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

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep", "3600"]
}
```

Step 3:To Initialize Terraform

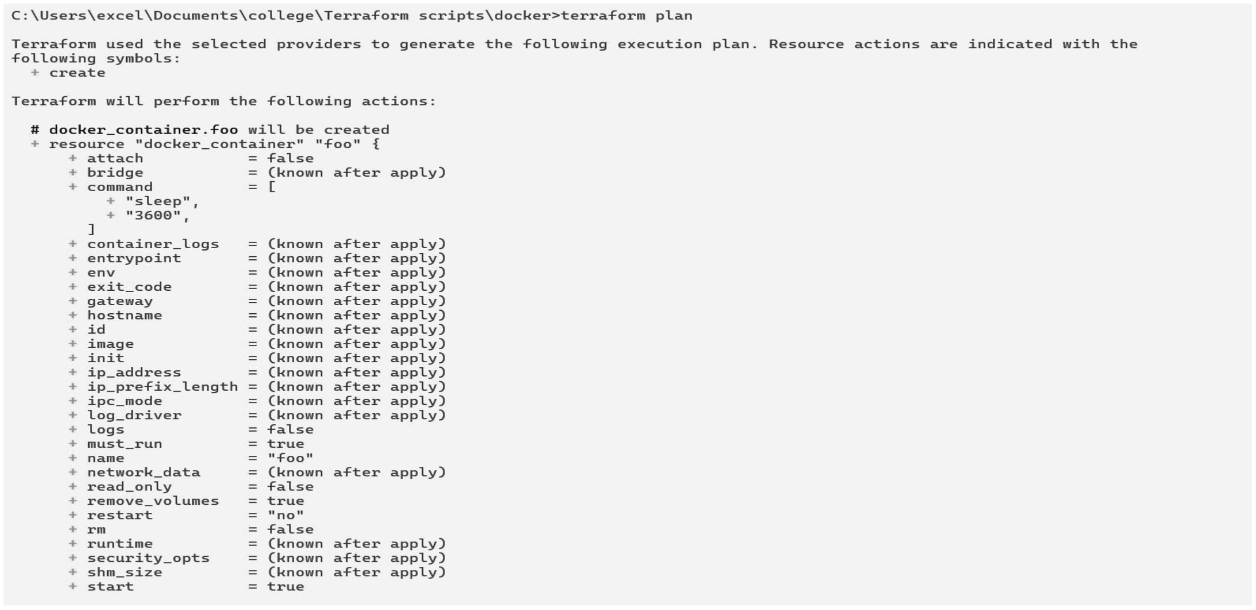
Run the command `terraform init` to initialize the Terraform configuration.



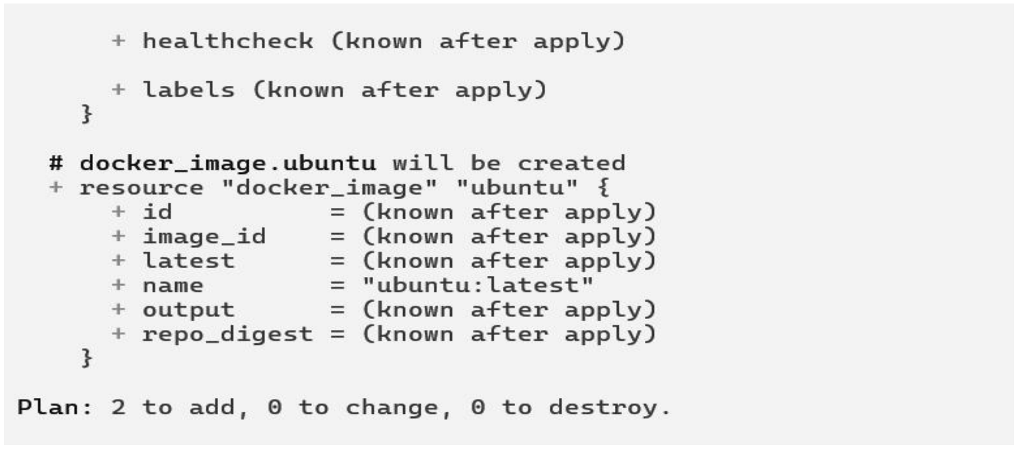
```
C:\Users\excel\Documents\college\Terraform scripts\docker>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (self-signed, key ID BD080C4571C6104C)
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.
```

Step 4:To Review Terraform Plan

Execute `terraform plan` to preview the resources that will be created.

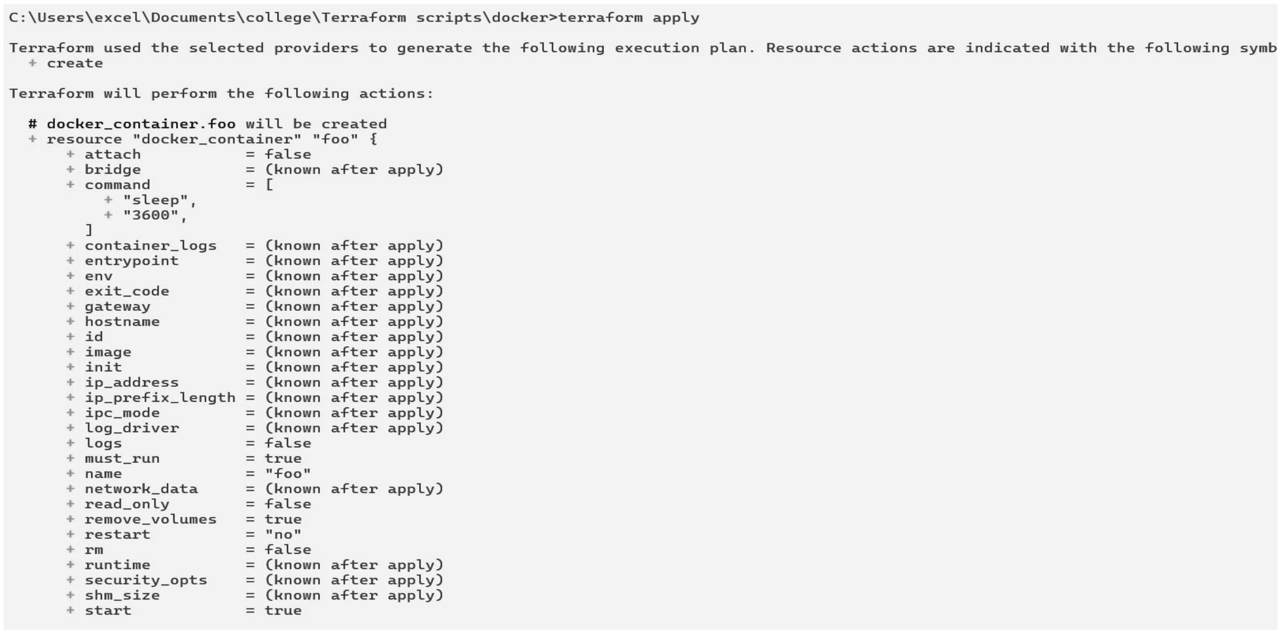


```
C:\Users\excel\Documents\college\Terraform scripts\docker>terraform plan

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           = [
          + "sleep",
          + "3600",
        ]
      + 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"
      + 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
```

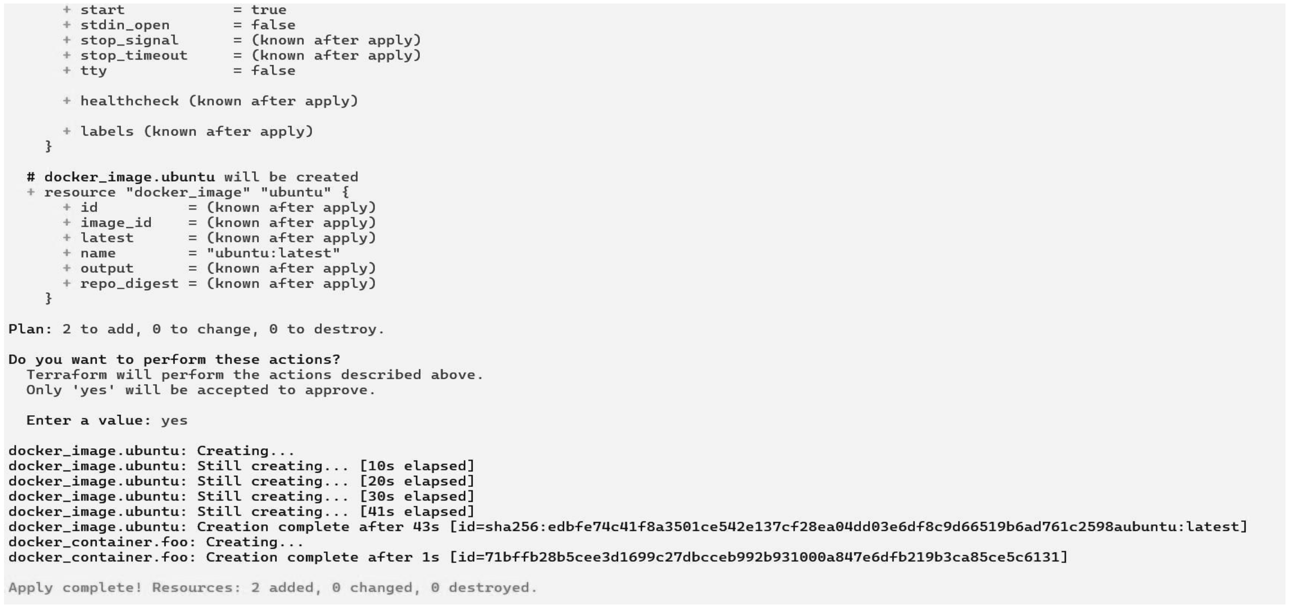


```
      + healthcheck (known after apply)

      + 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.
```

Step 5:To Apply Terraform Configuration
Run `terraform apply` to apply the configuration and create the Ubuntu Linux container.

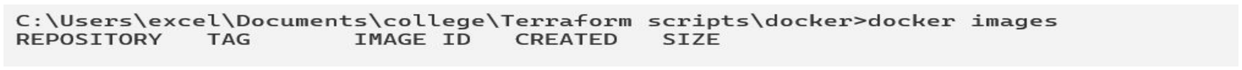


```
C:\Users\excel\Documents\college\Terraform scripts\docker>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symb
  + 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",
          + "3600",
        ]
      + 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"
      + 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
```



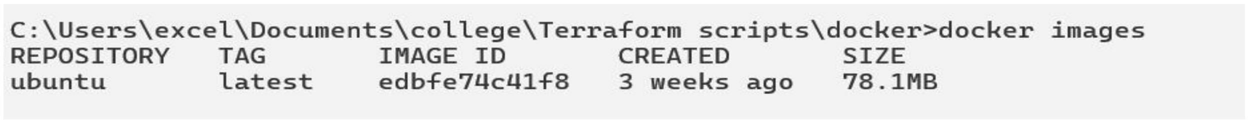
```
      + start             = true
      + stdin_open        = false
      + stop_signal       = (known after apply)
      + stop_timeout      = (known after apply)
      + tty               = false

      + 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.

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: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Still creating... [41s elapsed]
docker_image.ubuntu: Creation complete after 43s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=71bffb28b5cee3d1699c27dbcceb992b931000a847e6dfb219b3ca85ce5c6131]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Before executing `terraform apply`, list the Docker images.



```
C:\Users\excel\Documents\college\Terraform scripts\docker>docker images
REPOSITORY    TAG        IMAGE ID    CREATED    SIZE
```

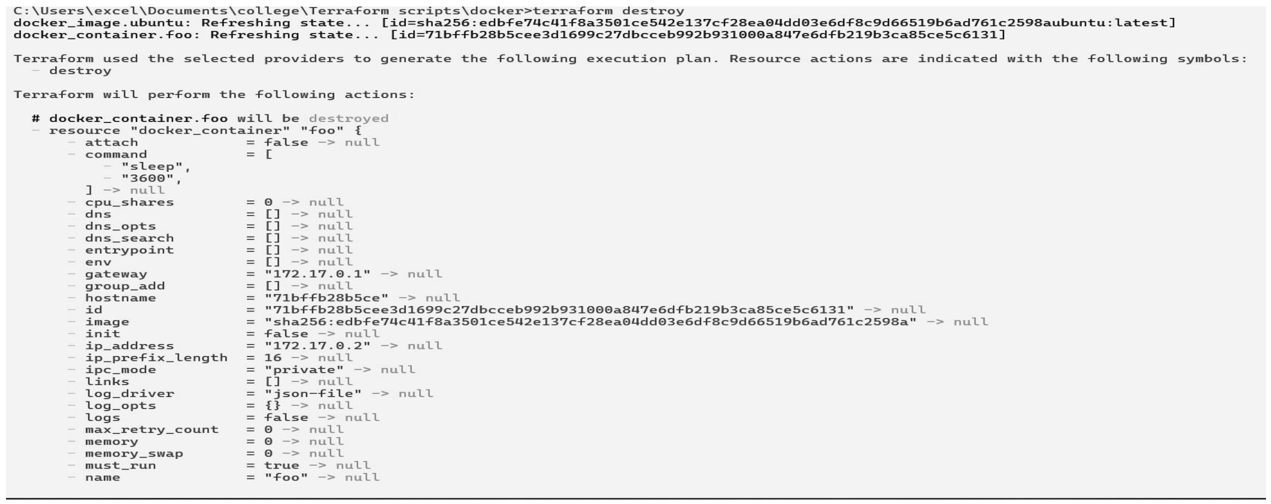
After executing `terraform apply`, list the Docker images again.



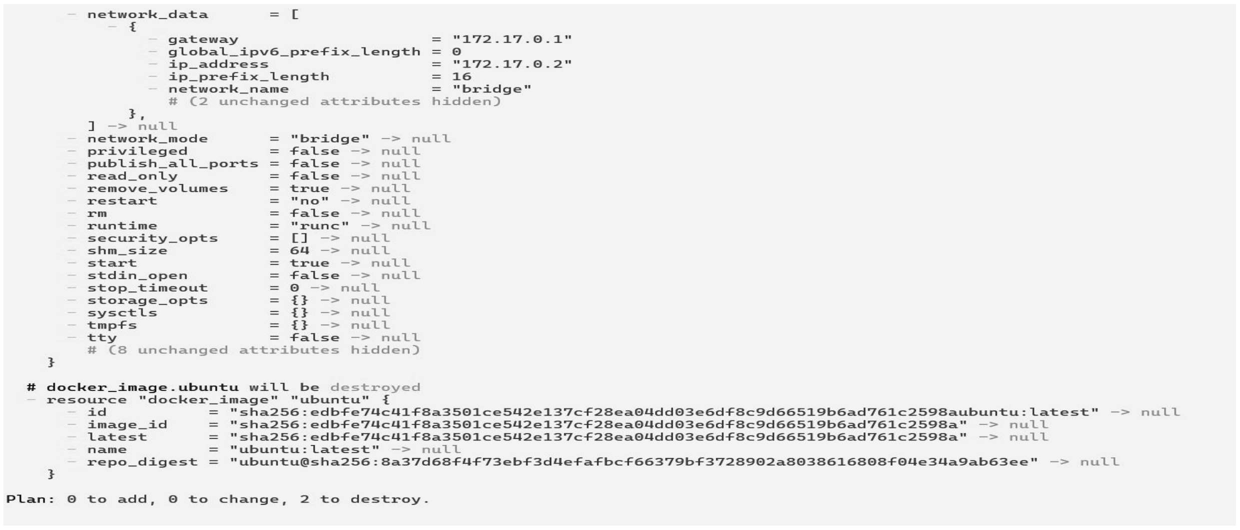
```
C:\Users\excel\Documents\college\Terraform scripts\docker>docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
ubuntu        latest    edbfe74c41f8   3 weeks ago   78.1MB
```

Step 6: Clean Up

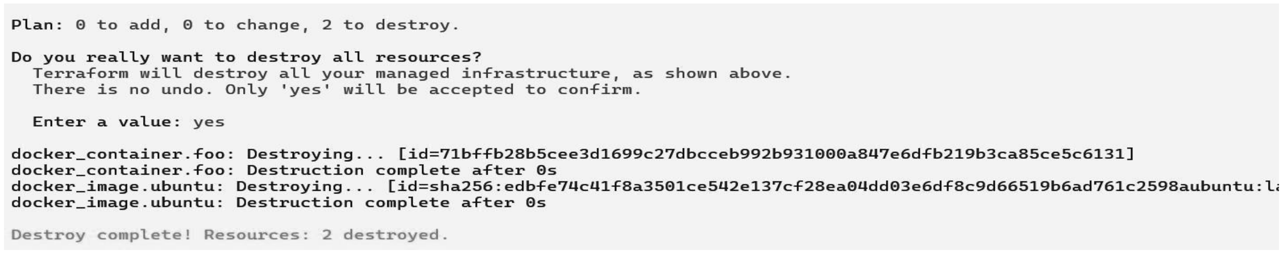
To delete the created Ubuntu container, run `terraform destroy`.



```
C:\Users\excel\Documents\college\Terraform scripts\docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=71bffb28b5cee3d1699c27dbcceb992b931000a847e6dfb219b3ca85ce5c6131]

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",
          - "3600",
        ] -> 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          = "71bffb28b5ce" -> null
      - id                = "71bffb28b5cee3d1699c27dbcceb992b931000a847e6dfb219b3ca85ce5c6131" -> 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
      - log_opts          = {} -> null
      - logs              = false -> null
      - max_retry_count   = 0 -> null
      - memory            = 0 -> null
      - memory_swap       = 0 -> null
      - must_run          = true -> null
      - name              = "foo" -> null
```

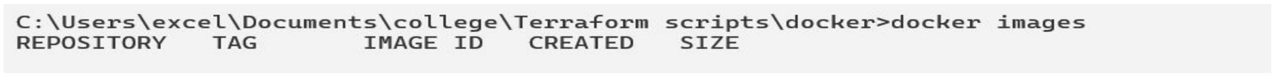


```
      - network_data      = [
          - {
              - gateway                   = "172.17.0.1"
              - global_ipv6_prefix_length = 0
              - ip_address                = "172.17.0.2"
              - ip_prefix_length          = 16
              - network_name              = "bridge"
                # (2 unchanged attributes hidden)
            },
        ] -> null
      - network_mode      = "bridge" -> null
      - privileged        = false -> null
      - publish_all_ports = false -> null
      - read_only         = false -> null
      - remove_volumes    = true -> null
      - restart           = "no" -> null
      - rm                = false -> null
      - runtime           = "runc" -> null
      - security_opts     = [] -> null
      - shm_size          = 64 -> null
      - start             = true -> null
      - stdin_open        = false -> null
      - stop_timeout      = 0 -> null
      - storage_opts      = {} -> null
      - sysctls           = {} -> null
      - tmpfs             = {} -> 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.
```



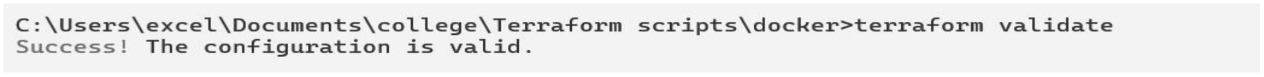
```
Plan: 0 to add, 0 to change, 2 to destroy.

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=71bffb28b5cee3d1699c27dbcceb992b931000a847e6dfb219b3ca85ce5c6131]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:la
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

After executing `terraform destroy`, list the Docker images one more time.



```
C:\Users\excel\Documents\college\Terraform scripts\docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

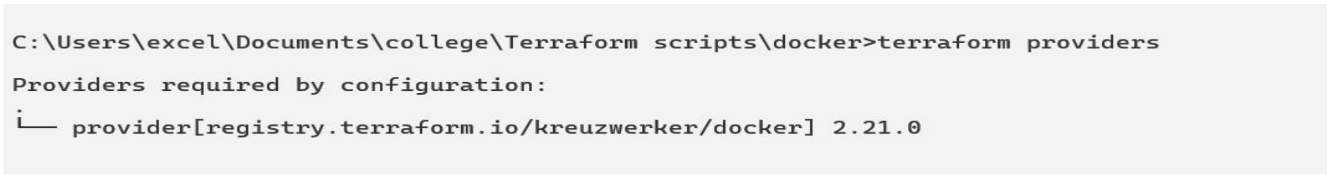
Step 7:To check correctness of configured files.
Execute `terraform validate` to check the correctness of your Terraform configuration files.



```
C:\Users\excel\Documents\college\Terraform scripts\docker>terraform validate
Success! The configuration is valid.
```

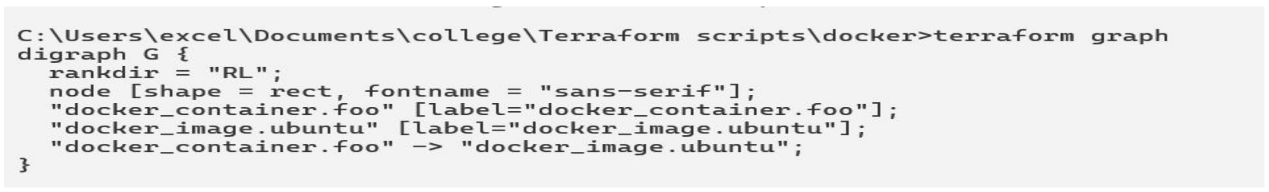
Step 8:To verify the details.

Run `terraform providers` to list the providers used in your configuration and verify their details.



```
C:\Users\excel\Documents\college\Terraform scripts\docker>terraform providers

Providers required by configuration:
└── provider[registry.terraform.io/kreuzwerker/docker] 2.21.0
```

Step 9:To generate visual representation.
Generate a visual representation of the dependency graph of your Terraform resources.



```
C:\Users\excel\Documents\college\Terraform scripts\docker>terraform graph
digraph G {
  rankdir = "RL";
  node [shape = rect, fontname = "sans-serif"];
  "docker_container.foo" [label="docker_container.foo"];
  "docker_image.ubuntu" [label="docker_image.ubuntu"];
  "docker_container.foo" -> "docker_image.ubuntu";
}
```