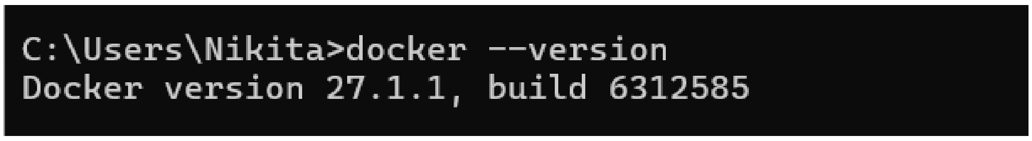
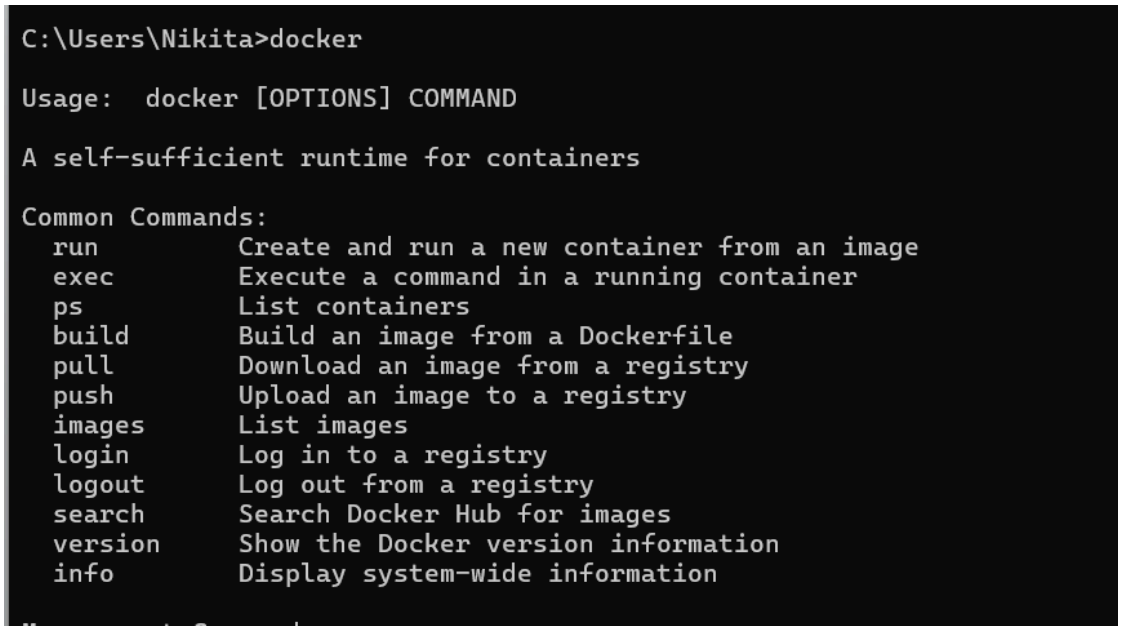
Experiment No: 6

Aim : Exp 6 To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.(S3 bucket or Docker)

Creating the docker image using terraform
1: Check the docker version and functionality if its not downloaded you can download it from https://www.docker.com/



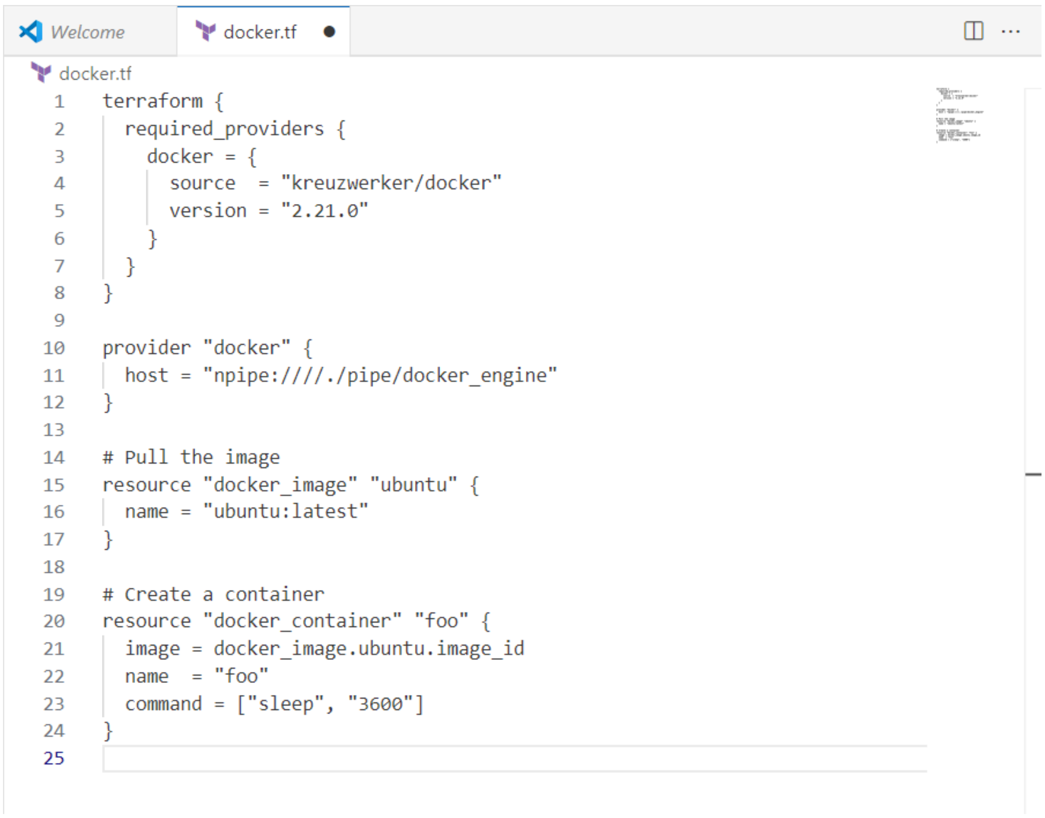
```
C:\Users\Nikita>docker --version
Docker version 27.1.1, build 6312585
```



```
C:\Users\Nikita>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
  info        Display system-wide information
```

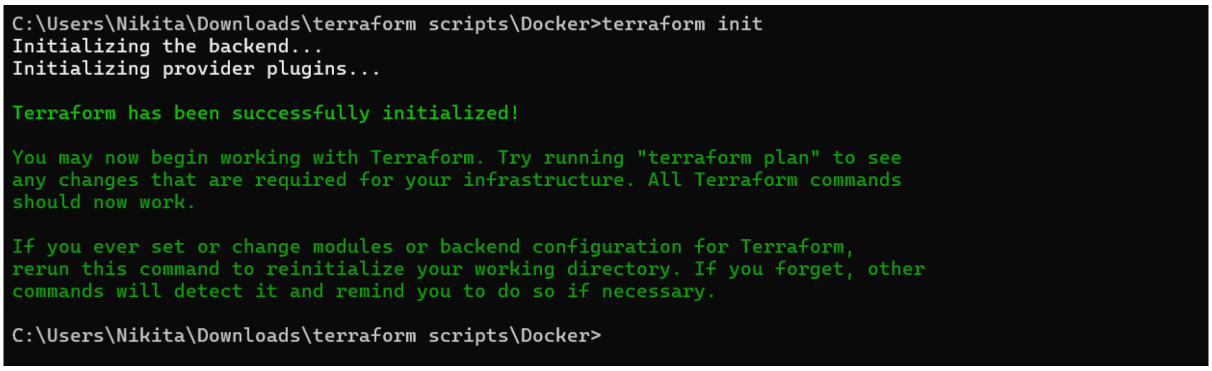
Now, create a folder named 'Terraform Scripts' in which we save our different types of scripts which will be further used in this experiment.
Step 2: Firstly create a new folder named **'Docker'** in the **'TerraformScripts'** folder. Then create a new docker.tf

```
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"]
}
```



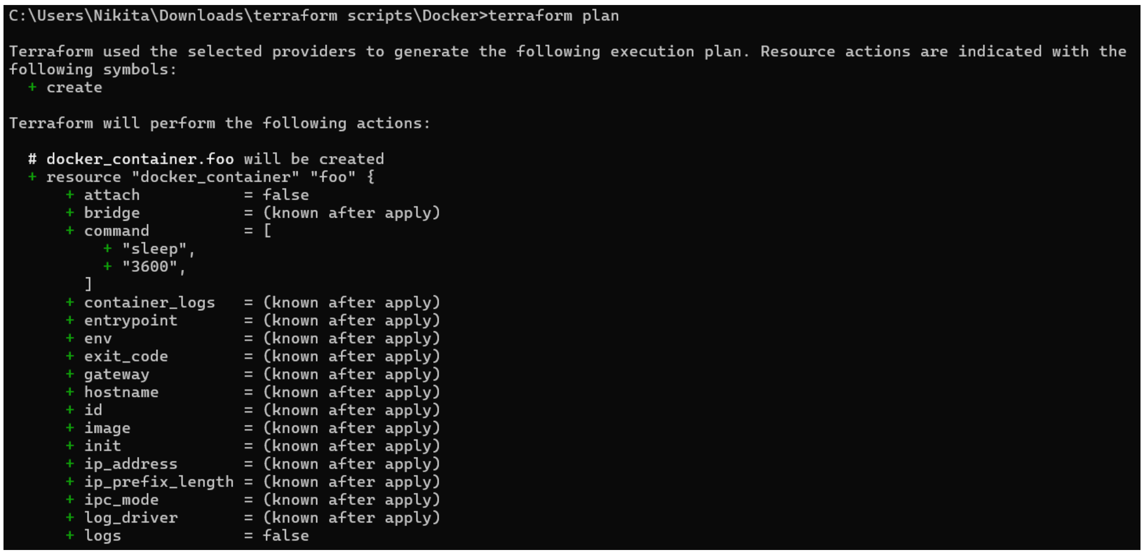
```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

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: Execute **Terraform Init** command to initialize the resources



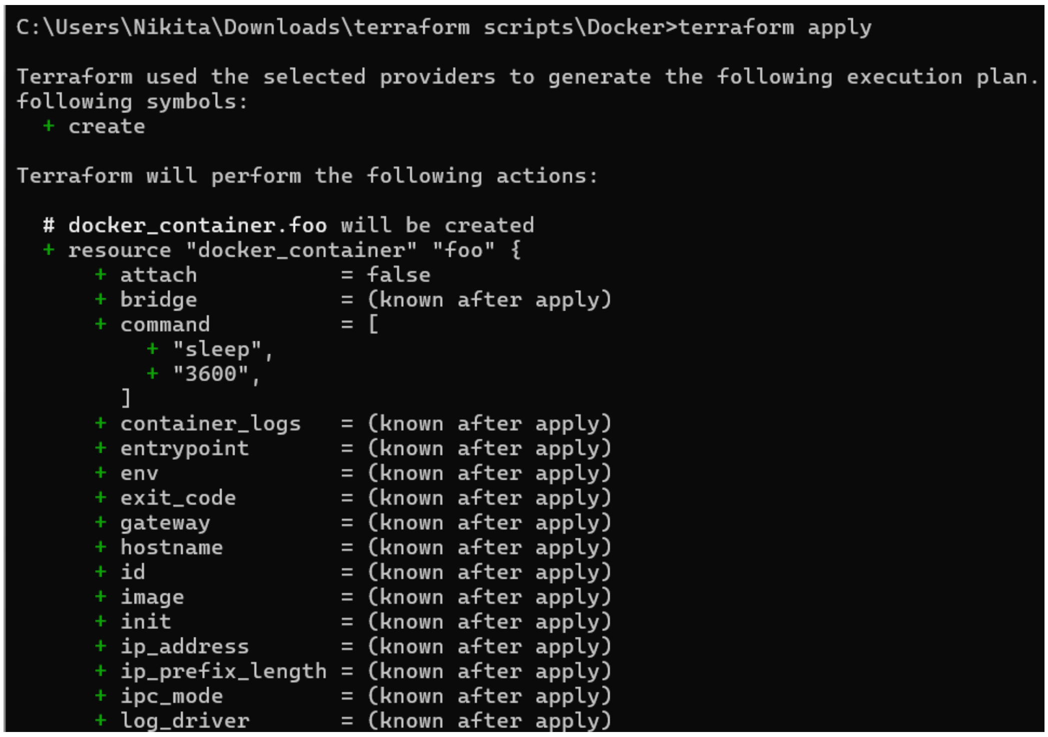
```
C:\Users\Nikita\Downloads\terraform scripts\Docker>terraform init
Initializing the backend...
Initializing provider plugins...

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.

C:\Users\Nikita\Downloads\terraform scripts\Docker>
```

4. Execute **Terraform plan** to see the available resources

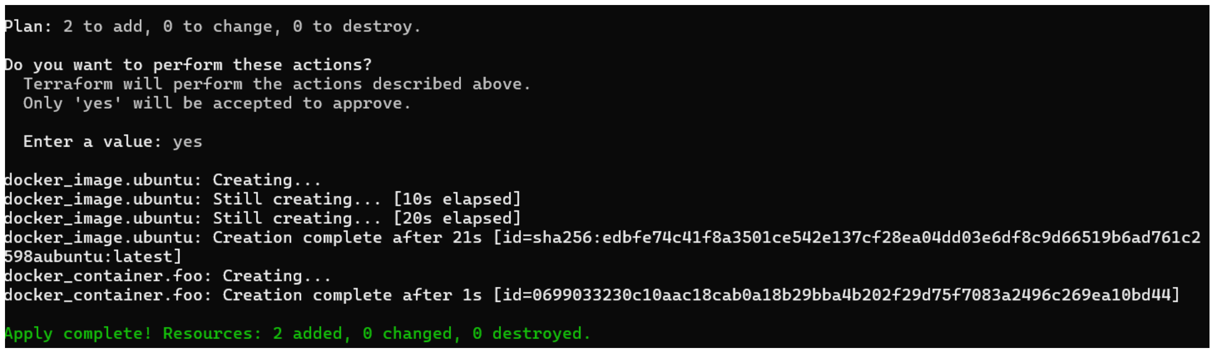


```
C:\Users\Nikita\Downloads\terraform scripts\Docker>terraform plan

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
```

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**”

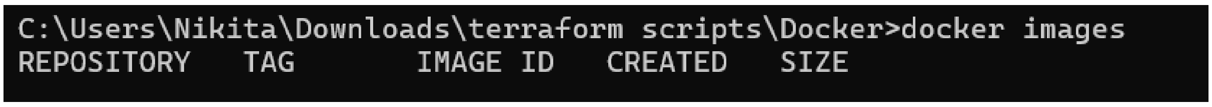


```
C:\Users\Nikita\Downloads\terraform scripts\Docker>terraform apply

Terraform used the selected providers to generate the following execution plan.
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)
```



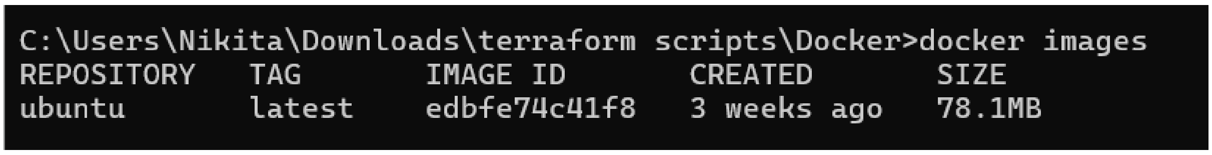
```
Plan: 2 to add, 0 to change, 0 to destroy.

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: Creation complete after 21s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2
598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=0699033230c10aac18cab0a18b29bba4b202f29d75f7083a2496c269ea10bd44]

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

Step 6. Docker images before executing this command



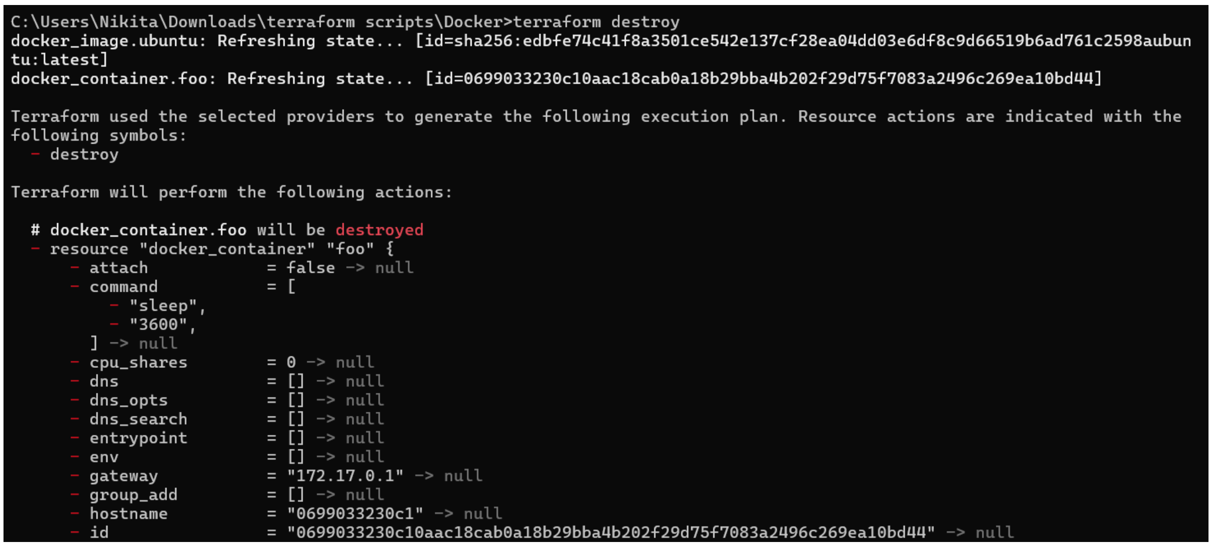
```
C:\Users\Nikita\Downloads\terraform scripts\Docker>docker images
REPOSITORY    TAG       IMAGE ID    CREATED    SIZE
```

Docker images after the execution of command

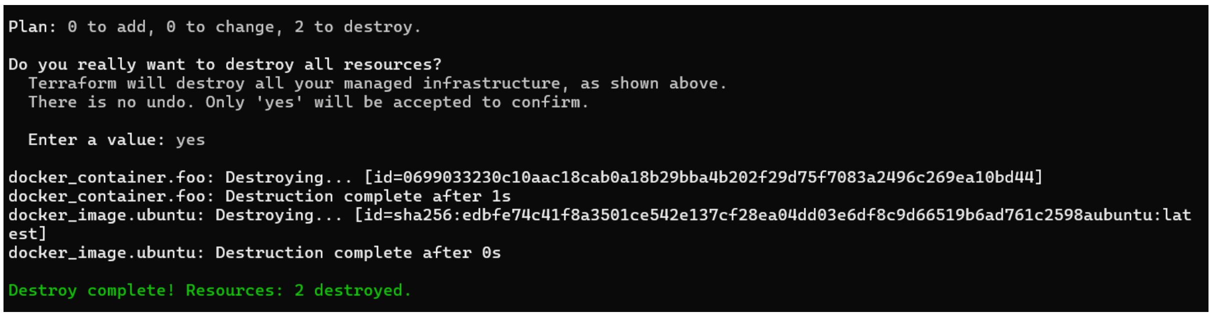


```
C:\Users\Nikita\Downloads\terraform scripts\Docker>docker images
REPOSITORY    TAG        IMAGE ID       CREATED         SIZE
ubuntu        latest     edbfe74c41f8   3 weeks ago     78.1MB
```

Step 7: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

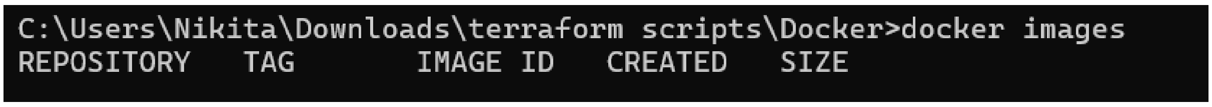


```
C:\Users\Nikita\Downloads\terraform scripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]
docker_container.foo: Refreshing state... [id=0699033230c10aac18cab0a18b29bba4b202f29d75f7083a2496c269ea10bd44]

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          = "0699033230c1" -> null
      - id                = "0699033230c10aac18cab0a18b29bba4b202f29d75f7083a2496c269ea10bd44" -> null
```

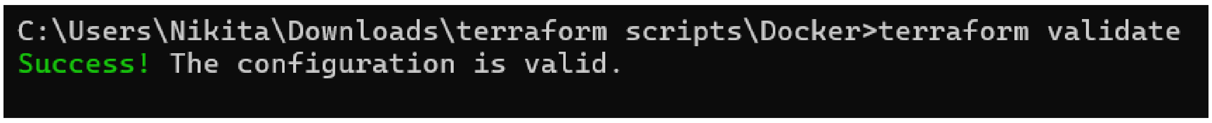


```
Plan: 0 to add, 0 to change, 2 to destroy.

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=0699033230c10aac18cab0a18b29bba4b202f29d75f7083a2496c269ea10bd44]
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.
```

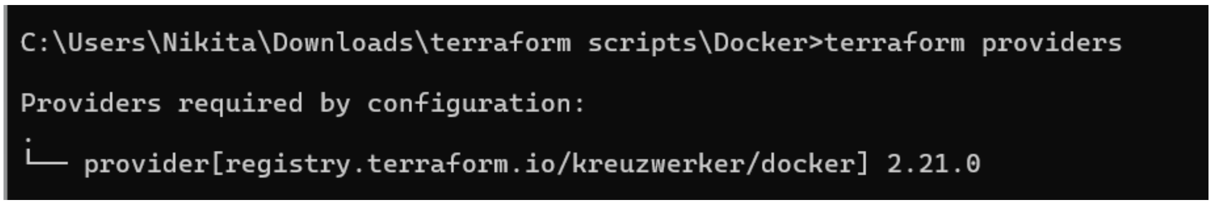
Docker images After Executing Destroy step



```
C:\Users\Nikita\Downloads\terraform scripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```



```
C:\Users\Nikita\Downloads\terraform scripts\Docker>terraform validate
Success! The configuration is valid.
```



```
C:\Users\Nikita\Downloads\terraform scripts\Docker>terraform providers

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

Conclusion:
We learned to use terraform and run commands using it