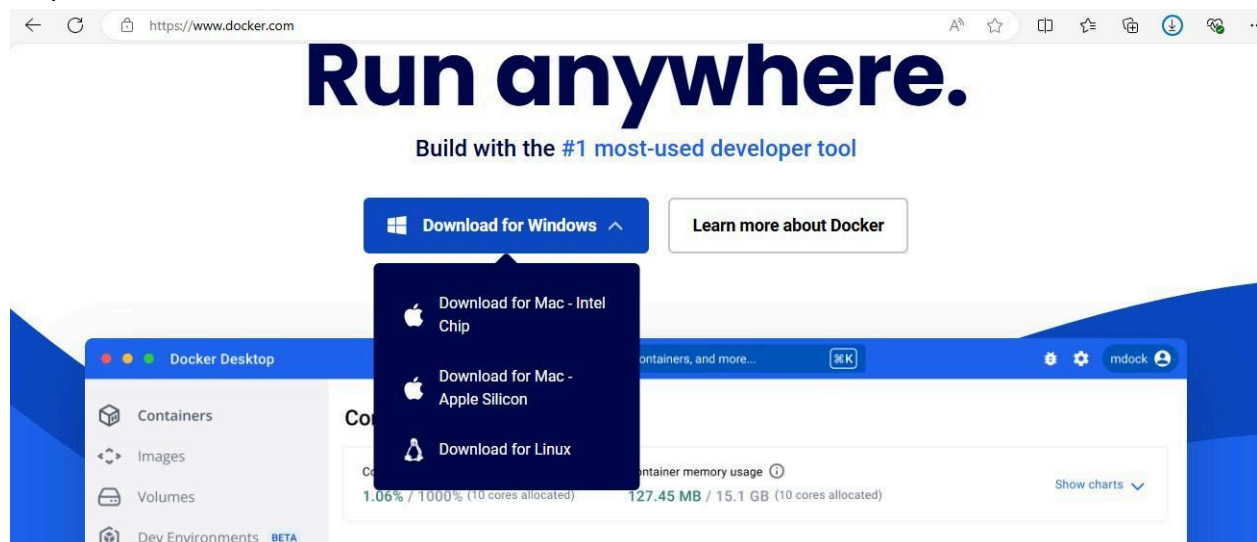


Experiment 6

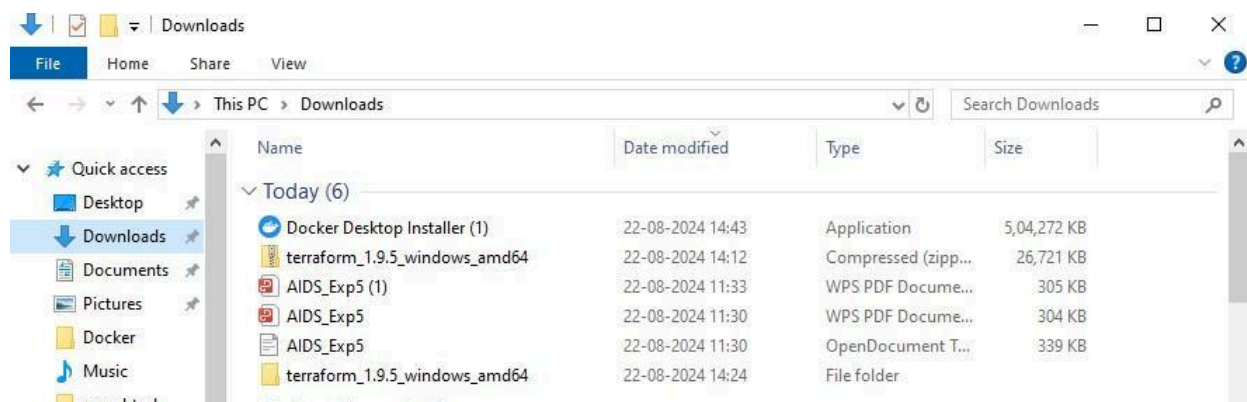
Aim:

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

Step 1: Download Docker form www.docker.com



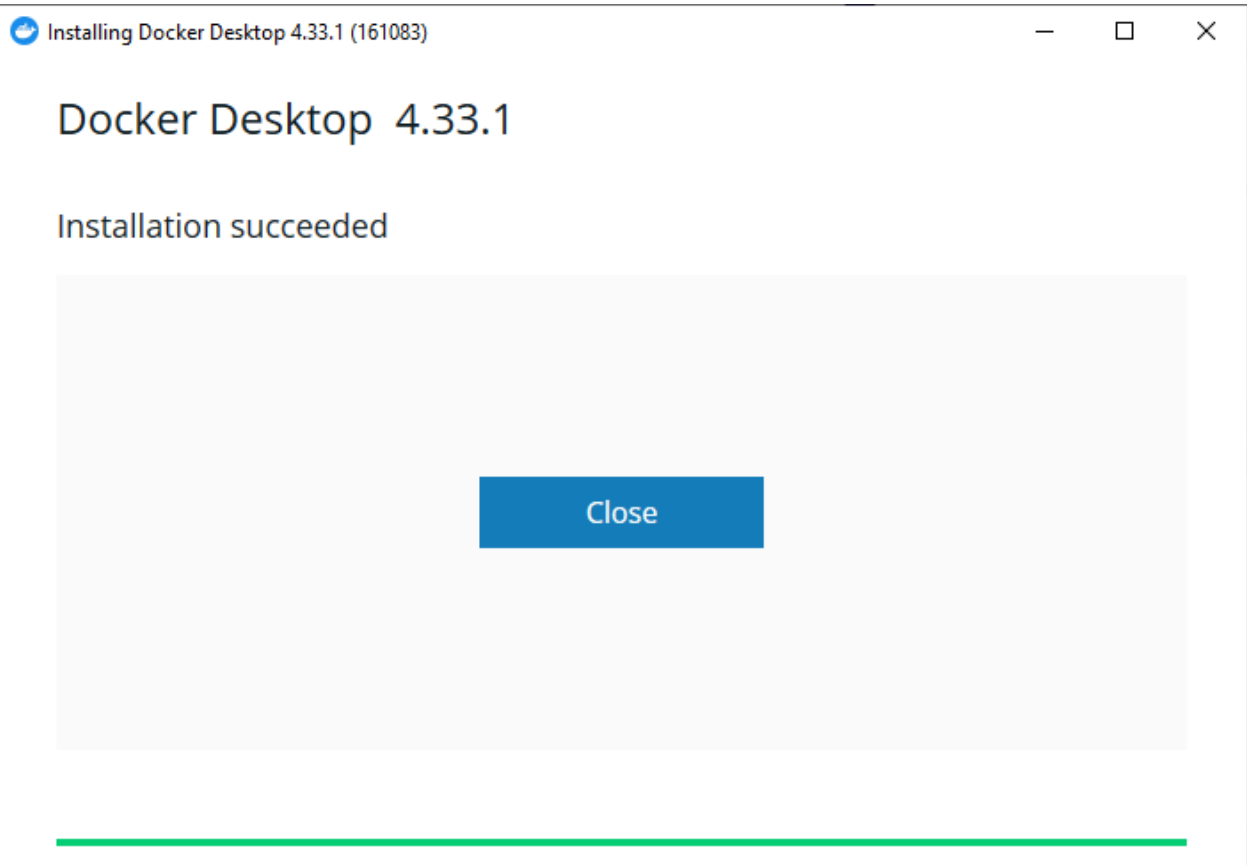
Step 2: The Docker is successfully downloaded. Now, run the docker installer and complete the installation.



Docker Desktop 4.33.1

Unpacking files...

```
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
Unpacking file: frontend/resources/regedit/vbs/uttl.vbs
Unpacking file: frontend/resources/regedit/vbs/regUtl.vbs
```



Step 3: Open Command Prompt and run as administrator. Enter the command `docker --version`, to check whether the docker is successfully installed.

A screenshot of a Windows Command Prompt window titled "Command Prompt". The window is black with white text. The user has entered the command `C:\Users\INFT>docker --version`, and the output is `Docker version 27.1.1, build 6312585`. Below this, the user has entered `C:\Users\INFT>docker`, and the output shows the usage information for the Docker CLI, including a list of common commands and their descriptions, and a list of management commands.

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

C:\Users\INFT>docker

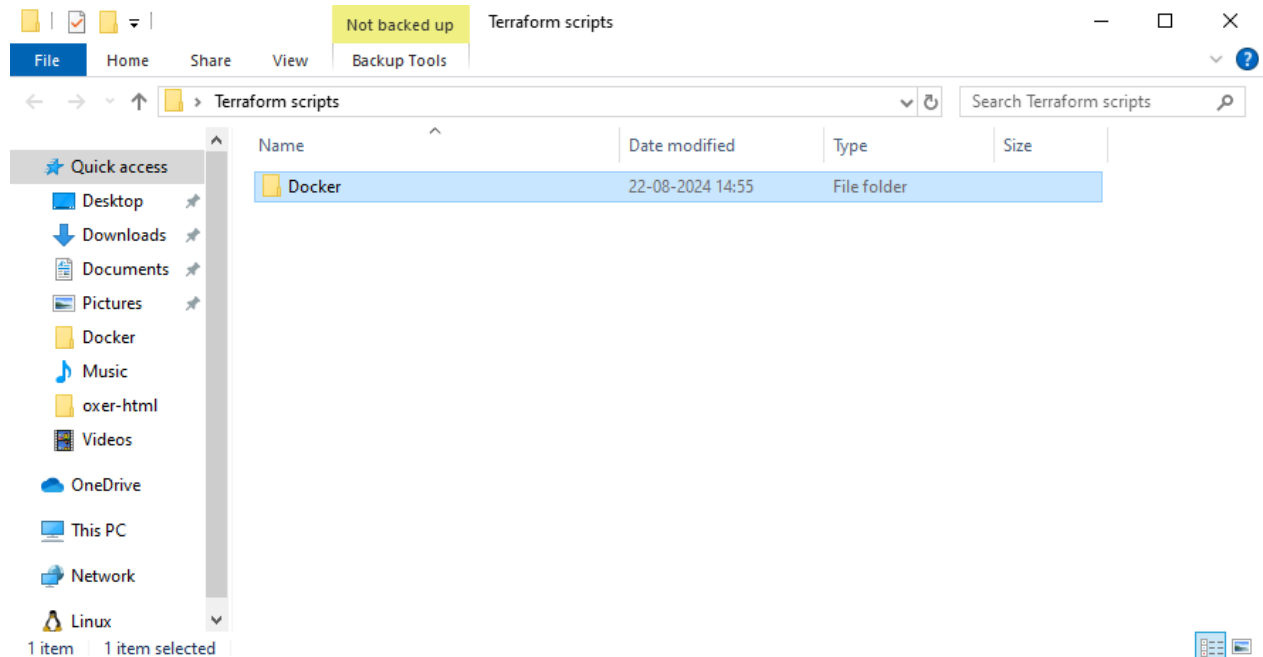
Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

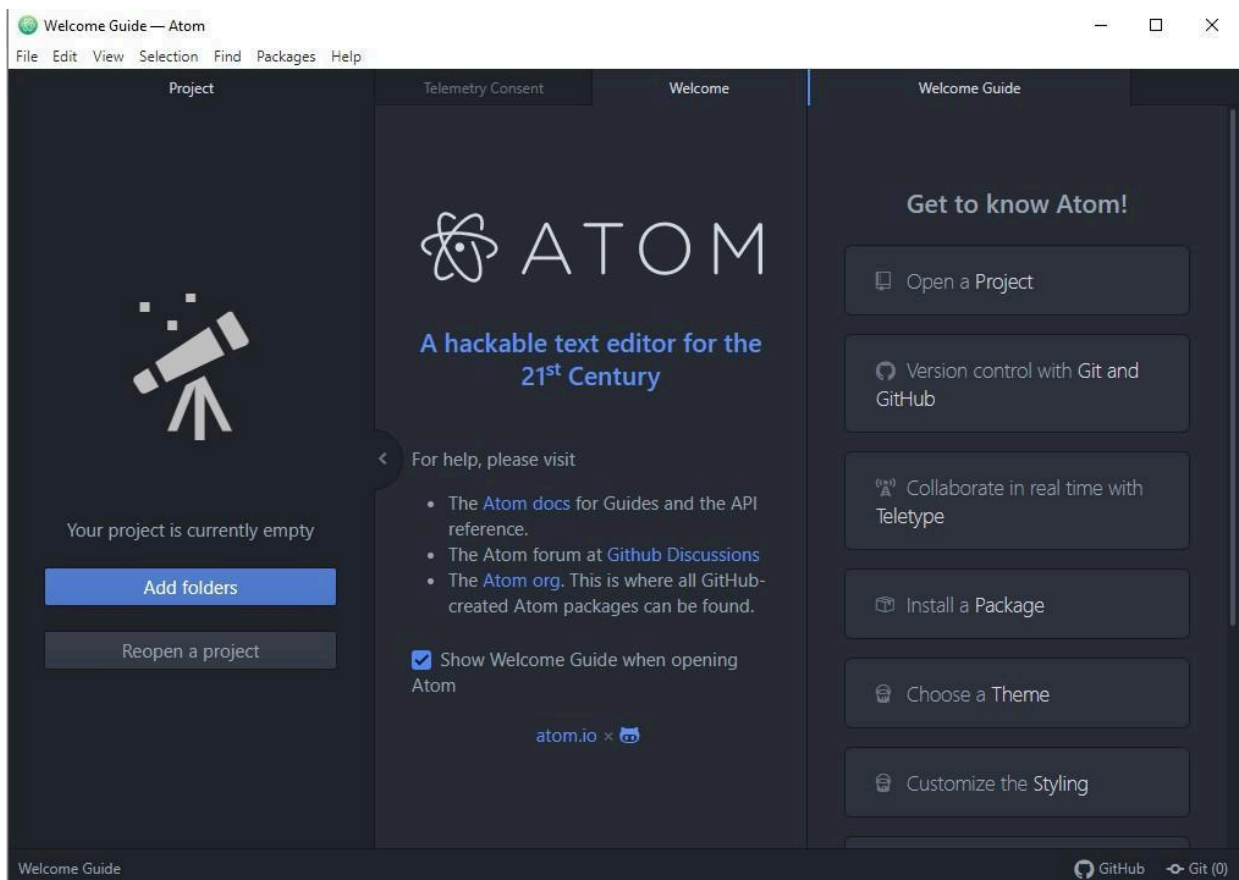
Common Commands:
run      Create and run a new container from an image
exec     Execute a command in a running container
ps       List containers
build    Build an image from a Dockerfile
pull     Download an image from a registry
push     Upload an image to a registry
images   List images
login    Log 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

Management Commands:
builder  Manage builds
buildx*  Docker Buildx
compose* Docker Compose
container Manage containers
context  Manage contexts
```

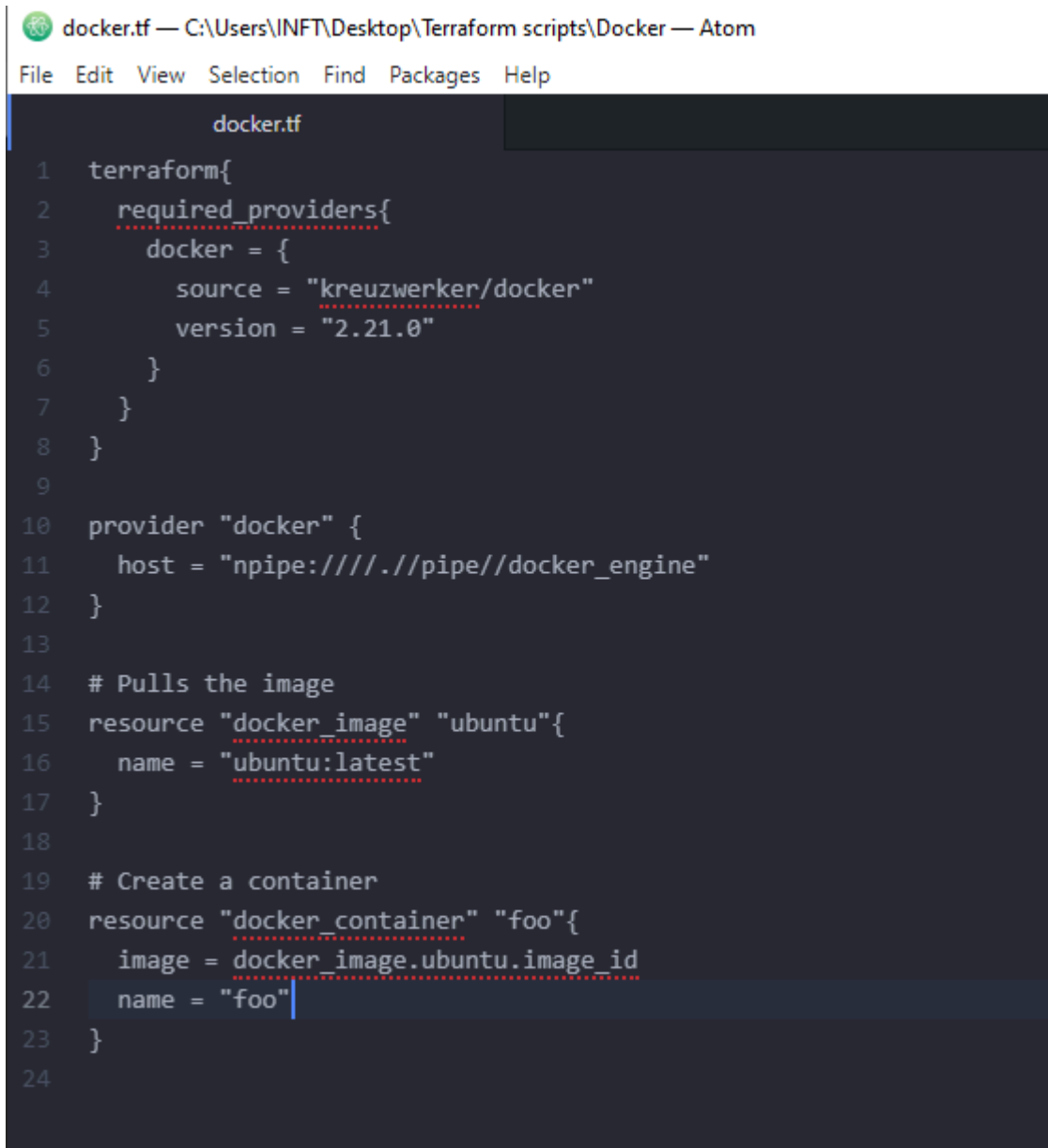
Step 4: Create a folder `Terraform_scripts` and inside it create a folder named `Docker`.



Step 5: Download Atom Editor.



Step 6: Run the following script in the Atom Editor



```
docker.tf
1 terraform{
2   required_providers{
3     docker = {
4       source = "kreuzwerker/docker"
5       version = "2.21.0"
6     }
7   }
8 }
9
10 provider "docker" {
11   host = "npipe:////./pipe//docker_engine"
12 }
13
14 # Pulls the image
15 resource "docker_image" "ubuntu"{
16   name = "ubuntu:latest"
17 }
18
19 # Create a container
20 resource "docker_container" "foo"{
21   image = docker_image.ubuntu.image_id
22   name = "foo"
23 }
24
```

Step 7: Open Windows Explorer and run the following command terraform init, terraform plan, terraform apply, terraform destroy and docker images.

```
PS C:\Users\INFT\Desktop\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.
PS C:\Users\INFT\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)
  + endpoint        = (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)
```

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.

PS C:\Users\INFT\Desktop\Terraform_scripts\Docke> 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)
  + endpoint         = (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
  + 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: Creation complete after 11s [id=sha256:edbf74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu: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" {

```
Windows PowerShell

PS C:\Users\INFT\Desktop\Terraform_scripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]

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_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, 1 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_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:lat
est]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\INFT\Desktop\Terraform_scripts\Docker>
```

```
Windows PowerShell

PS C:\Users\INFT\Desktop\Terraform_scripts\Docker> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
PS C:\Users\INFT\Desktop\Terraform_scripts\Docker>
```