Experiment 6:

1. Run some basic commands to check the version and confirm installation of docker

```
C:\Users\INFT505-17>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
```

```
C:\Users\INFT505-17>docker --version
Docker version 27.1.1, build 6312585
```

2. Initialise terraform

```
C:\Users\INFT505-17>terraform init
Terraform initialized in an empty directory!

The directory has no Terraform configuration files. You may begin working
with Terraform immediately by creating Terraform configuration files.

C:\Users\INFT505-17>cd Desktop

C:\Users\INFT505-17\Desktop>
C:\Users\INFT505-17\Desktop>cd Terraform Scripts

C:\Users\INFT505-17\Desktop\Terraform Scripts>cd Docker

C:\Users\INFT505-17\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.
```

3. Terraform plan

```
C:\Users\INFT505-17\Desktop\Terraform Scripts\Docker>terraform plan

Terraform used the selected providers to generate the following exec
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"
      + network_data     = (known after apply)
```

4. Terraform 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 3s [id=1adc9dfc498825398e014939d7966749d843181417953d0b9f462a55ae7c1492]

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

```
C:\Users\INFT505-17\Desktop\Terraform Scripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB
```

5. Terraform destroy

```
C:\Users\INFT505-17\Desktop\Terraform Scripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=ad11c2cdc9ccae208b906f7e6be260805512a66568a65f6b3bffb3682cf30cb4]

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           = [
          - "/bin/bash",
          - "-c",
          - "while true; do sleep 3600; done",
        ] -> null
      - cpu_shares        = 0 -> null
      - dns               = [] -> null
      - dns_opts          = [] -> null
      - dns_search        = [] -> null
      - entrypoint        = [] -> 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=ad11c2cdc9ccae208b906f7e6be260805512a66568a65f6b3bffb3682cf30cb4]
docker_container.foo: Destruction complete after 2s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

```
C:\Users\INFT505-17\Desktop\Terraform Scripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

C:\Users\INFT505-17\Desktop\Terraform Scripts\Docker>
```

6. Terraform Validate

```
C:\Users\INFT505-17\Desktop\Terraform Scripts\Docker>terraform validate
Success! The configuration is valid.
```

7. Terraform refresh

```
:\Users\INFT505-17\Desktop\Terraform Scripts\Docker>terraform refresh
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubu
:u:latest]
docker_container.foo: Refreshing state... [id=4adda4f9a5c585809eebe921da757560d333f9a0101a8cd25784bd238355aba5]
```

8. Terraform state list

```
C:\Users\INFT505-17\Desktop\Terraform Scripts\Docker>terraform state list
docker_container.foo
docker_image.ubuntu
```