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

1) Check the docker version and functionality if its not downloaded you can download it from https://www.docker.com/
2) Use *docker --version* command to check for the version of docker.

```
PS C:\Users\hp> docker --version
Docker version 27.0.3, build 7d4bcd8
```

```
PS C:\Users\hp> 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
```

3) Create a folder named **'Terraform Scripts**' in which we save our different types of scripts which will be further used in this experiment.
4) Create a new docker.tf file using an IDE and write the following contents into it to create a 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"]
}
```

docker.tf

C: > Users > hp > OneDrive > Desktop > TerraformScripts > Docker > 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"]
}
```

5) Execute *terraform init* command to initialize the resources (Make sure you are in the Docker directory before executing the command)

```
PS C:\Users\hp\OneDrive\Desktop\TerraformScripts\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.
```

6) Execute *terraform plan* to see the available resources if the build fails, there may be a chance that the docker engine is not running. Go to Docker Desktop and start the engine.

```
PS C:\Users\hp\OneDrive\Desktop\TerraformScripts\Docker> terraform plan

Planning failed. Terraform encountered an error while generating this plan.


  Error: Error pinging Docker server: error during connect: This error may indicate that the docker daemon is not running.: Get "http://%2F%2F.%2Fpipe%2Fdo
cker_engine/_ping": open //./pipe/docker_engine: The system cannot find the file specified.

    with provider["registry.terraform.io/kreuzwerker/docker"],
    on docker.tf line 10, in provider "docker":
    10: provider "docker" {
```

7) Again, run the terraform plan.

```
PS C:\Users\hp\OneDrive\Desktop\TerraformScripts\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)
```

8) Execute *terraform apply* to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration.

```
PS C:\Users\hp\OneDrive\Desktop\TerraformScripts\Docker> terraform apply

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

```
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... [40s elapsed]
docker_image.ubuntu: Still creating... [50s elapsed]
docker_image.ubuntu: Still creating... [1m0s elapsed]
docker_image.ubuntu: Still creating... [1m10s elapsed]
docker_image.ubuntu: Creation complete after 1m11s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad
761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Still creating... [10s elapsed]
docker_container.foo: Still creating... [20s elapsed]
docker_container.foo: Still creating... [30s elapsed]
docker_container.foo: Creation complete after 34s [id=0461fcf8f00556f61c080a147e0d0b33687fc09868abbad5c3e8205f6f6b033
1]
```

9) Docker images before executing this command

```
PS C:\Users\hp\OneDrive\Desktop\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Docker images after executing this command

```
PS C:\Users\hp\OneDrive\Desktop\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB
```

10) Execute *terraform destroy* to delete the configuration, which will automatically delete the Ubuntu Container.

```
PS C:\Users\hp\OneDrive\Desktop\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=0461fcf8f00556f61c080a147e0d0b33687fc09868abbad5c3e8205f6f6b0331]

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          = "0461fcf8f005" -> null
      - id                = "0461fcf8f00556f61c080a147e0d0b33687fc09868abbad5c3e8205f6f6b0331" -> 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
```

```
  # 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=0461fcf8f00556f61c080a147e0d0b33687fc09868abbad5c3e8205f6f6b0331]
docker_container.foo: Still destroying... [id=0461fcf8f00556f61c080a147e0d0b33687fc09868abbad5c3e8205f6f6b0331, 10s elapsed]
docker_container.foo: Still destroying... [id=0461fcf8f00556f61c080a147e0d0b33687fc09868abbad5c3e8205f6f6b0331, 20s elapsed]
docker_container.foo: Still destroying... [id=0461fcf8f00556f61c080a147e0d0b33687fc09868abbad5c3e8205f6f6b0331, 30s elapsed]
docker_container.foo: Still destroying... [id=0461fcf8f00556f61c080a147e0d0b33687fc09868abbad5c3e8205f6f6b0331, 40s elapsed]
docker_container.foo: Destruction complete after 44s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 7s

Destroy complete! Resources: 2 destroyed.
```

11) Docker images after the destroy command execution

```
PS C:\Users\hp\OneDrive\Desktop\TerraformScripts\Docker> docker images
REPOSITORY    TAG        IMAGE ID    CREATED    SIZE
```