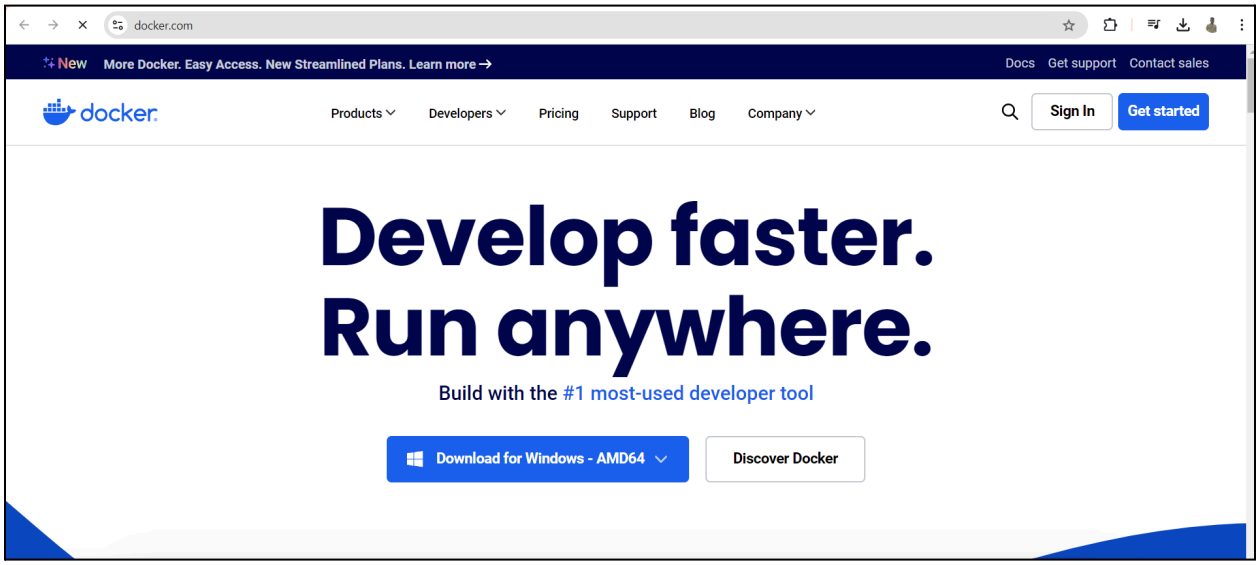
Name: Divesh Lulla Class: D15C Roll No : 31

Experiment 6

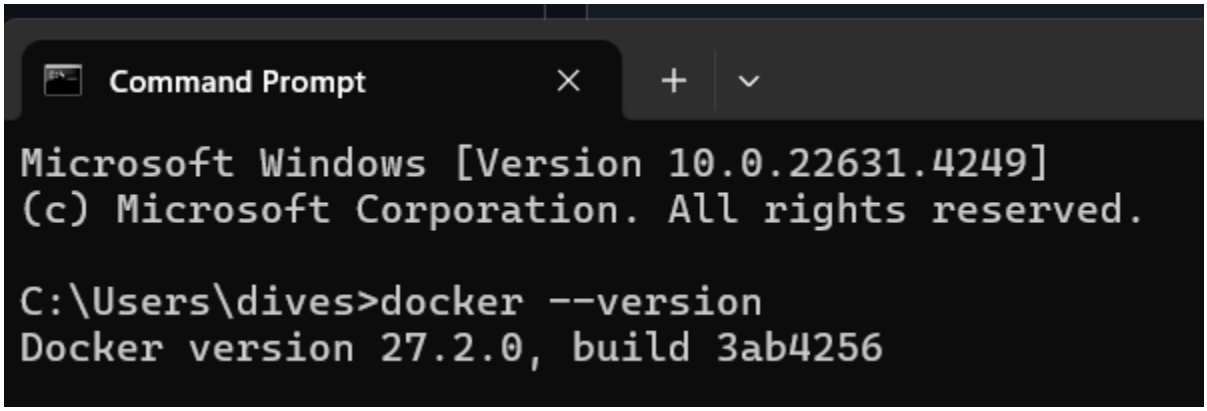
Aim: To Build, change and Destroy AWS/ GCP/ Microsoft Azure/ Digital Ocean using Terraform.

Steps:

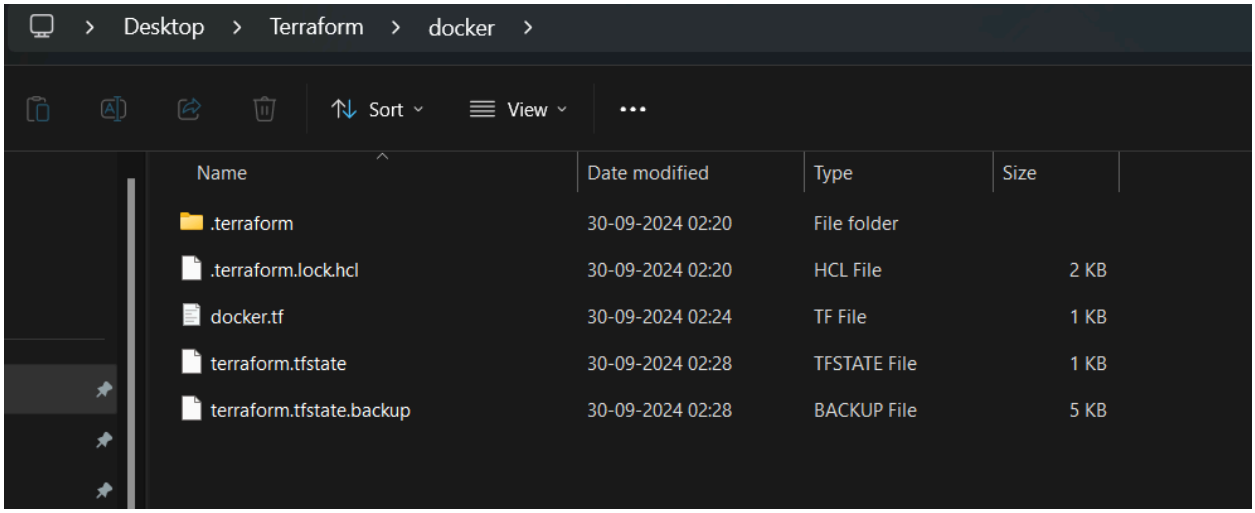
1) Install and setup the Docker Desktop using the official website.



2) Open the cmd and run –version command to check if the docker is installed correctly.



3) Create a new folder name Terraform scripts and inside that folder create one more folder name Docker where all the docker scripts will be saved.



4) Open the vs code and write down the below code in it and save the file with the .tf extension.

```
terraform {
  required_providers {
    docker = {
      source = "kreuzwerker/docker"
      version="2.21.0"
    }
  }
}
provider "docker" {
 host = "npipe:////.//pipe//docker_engine"
}
# Pulls 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","infinity"]
}
```

C: > Users > dives > Desktop > Terraform > docker > docker.tf

```
terraform {
  required_providers {
    docker = {
      source = "kreuzwerker/docker"
      version="2.21.0"
    }
  }
}
provider "docker" {
 host = "npipe:////.//pipe//docker_engine"
}
# Pulls the image
resource "docker_image" "ubuntu" {
    name = "ubuntu:latest"
}
# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name = "foo"
}
```

5) Now perform the terraform init command in the powershell.This command will initialize the working directory and install the necessary plugins.

```
PS C:\Users\dives\Desktop\Terraform\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\dives\Desktop\Terraform\docker>
```

6) Next execute the "terraform plan” command. The terraform plan command is used to create an execution plan for terraform. This plan shows you what changes Terraform will make to your infrastructure based on your current configuration files and the state of your existing infrastructure.

```
PS C:\Users\dives\Desktop\Terraform\docker> terraform plan

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

7) After this execute the "terraform apply" command. The terraform apply command executes the actions proposed in the terraform plan. It is used to deploy your infrastructure.

The changes will be made in the code because it was generating the error.

Command = [“sleep”, “infinity”] was added.

C: > Users > dives > Desktop > Terraform > docker > docker.tf

```
terraform {
  required_providers {
    docker = {
      source = "kreuzwerker/docker"
      version="2.21.0"
    }
  }
}
provider "docker" {
 host = "npipe:////.//pipe//docker_engine"
}
# Pulls 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","infinity"]
}
```

```
PS C:\Users\dives\Desktop\Terraform\docker> terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:dfc10878be8d8fc9c61cbff33166cb1d1fe4439153924370

Terraform used the selected providers to generate the following execution plan. Resource actions are
  + 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",
          + "infinity",
        ]
      + 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            = "sha256:dfc10878be8d8fc9c61cbff33166cb1d1fe44391539243703c72766894fa834a"
      + 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
```

```
      + 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)
    }

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 1s [id=4f97138fcbe96c8b3

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\dives\Desktop\Terraform\docker>
```

8) The image created can be checked by using the docker images command. It will show the repository, tag, image id, creation time, and size as shown below.

```
PS C:\Users\dives\Desktop\Terraform\docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
ubuntu        latest    dfc10878be8d   4 weeks ago   117MB
```

9) For destroying the container we can use the terraform destroy command.

```
PS C:\Users\dives\Desktop\Terraform\docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:dfc10878be8d8fc9c61cbff33166cb1d1fe4439153924370 3c72766894fa834aubuntu:latest]
docker_container.foo: Refreshing state... [id=4f97138fcbe96c8b3a118fc398abb71f57c3c729db3a40ee9c3dc5a11a6d9e54]

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",
          - "infinity",
        ] -> 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          = "4f97138fcbe9" -> null
      - id                = "4f97138fcbe96c8b3a118fc398abb71f57c3c729db3a40ee9c3dc5a11a6d9e54" -> null
      - image             = "sha256:dfc10878be8d8fc9c61cbff33166cb1d1fe4439153924370 3c72766894fa834a" -> 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
```

10) Now we will give command of docker images to check the image is destroyed or not.

```
PS C:\Users\dives\Desktop\Terraform\docker> docker images
REPOSITORY    TAG       IMAGE ID    CREATED    SIZE
PS C:\Users\dives\Desktop\Terraform\docker>
```