Aim:Creating docker image using terraform

1) Download and Install Docker Desktop from https://www.docker.com/

Step 1: Check the docker functionality

```
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % docker -v
Docker version 27.0.3, build 7d4bcd8
```

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 file using nano editor and write the followingcontents into it to create a Ubuntu Linux container.

Script:

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}
provider "docker" {
  host = "unix:///var/run/docker.sock"
}
# 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 = ["/bin/bash", "-c", "while true; do sleep 3600; done"]
}
```

```
  UW PICO 5.09

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

provider "docker" {
  host = "unix:///var/run/docker.sock"
}

# 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 = ["/bin/bash", "-c", "while true; do sleep 3600; done"]
}
```

Step 3: Execute Terraform Init command to initialize the resources:

```
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air 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.
```

Step 4: Execute Terraform plan to see the available resources

```
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)
      + 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)
      + 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.
```

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"

```
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 0s [id=f2b095b9576b22cc90eaae6860991144a11cc4a1255f9e1c4c99e5f4ca070857]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker %
```

Docker images, After Executing Apply step:

```
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Creation complete after 12s [id=sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=d2fa174ea2233d74813507a43ac8bd8136f623420da4ea251f0e43503b63446c]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    1a799365aa63   3 weeks ago   101MB
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS     NAMES
d2fa174ea223   1a799365aa63   "/bin/bash -c 'while…"   About a minute ago   Up About a minute             foo
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker %
```

Docker Validate and Docker providers:

```
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % terraform validate
Success! The configuration is valid.

(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % terraform providers

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

Providers required by state:

    provider[registry.terraform.io/kreuzwerker/docker]

(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker %
```

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.
Docker images After Executing Destroy step

```
  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id          = "sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d8b02ubuntu:latest" -> null
      - image_id    = "sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d8b02" -> null
      - latest      = "sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d8b02" -> 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=d2fa174ea2233d74813507a43ac8bd8136f623420da4ea251f0e43503b63446c]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d8b02ubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker %
```