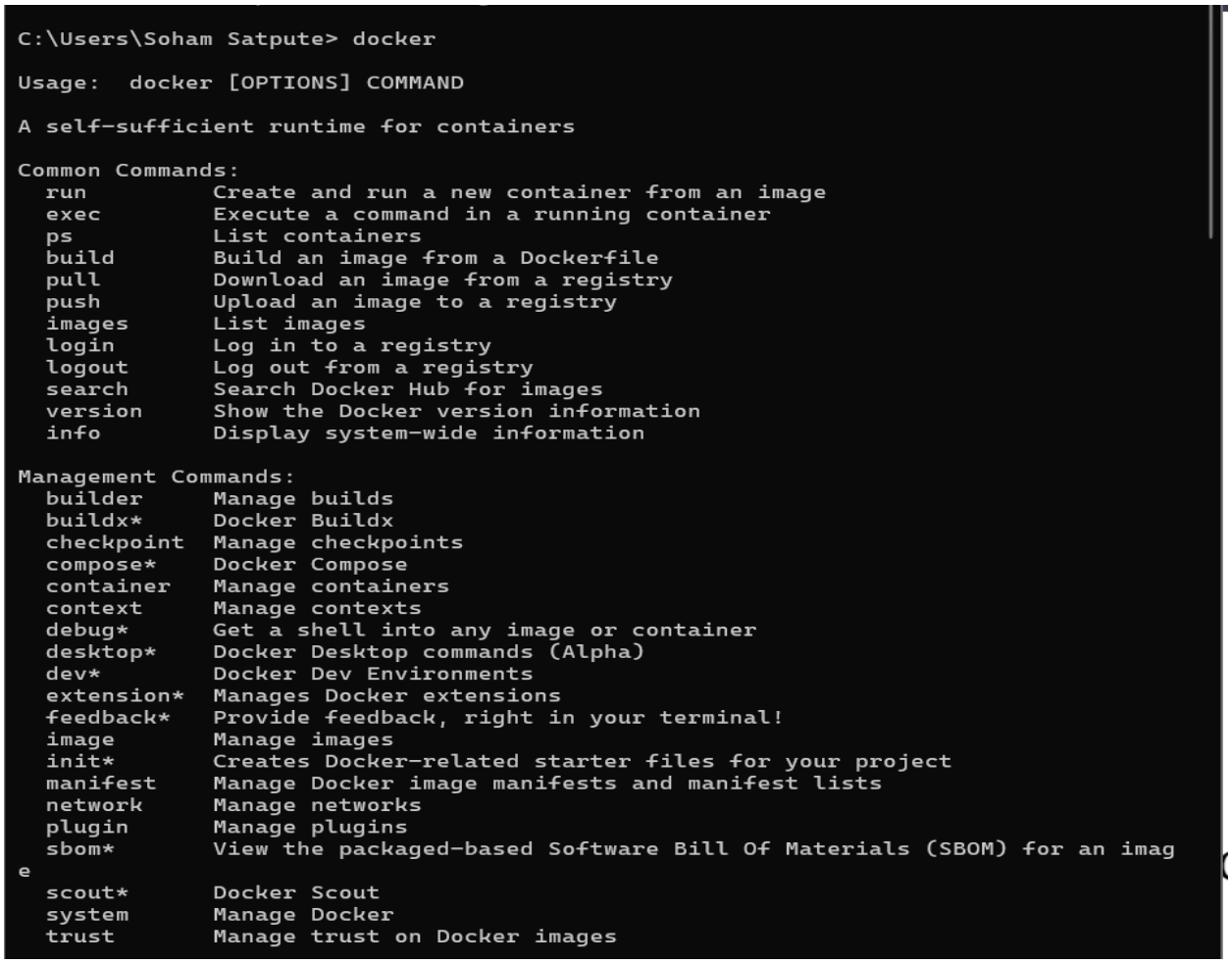
Experiment no 6

Name: Soham Satpute
Roll_No :52

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

Part A:Creating docker image using terraform Prerequisite:
1) Download and Install Docker Desktop from https://www.docker.com/

Step 1:
Check Docker functionality:



```
C:\Users\Soham Satpute> 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
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
  image       Manage images
  init*       Creates Docker-related starter files for your project
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an imag
e
  scout*      Docker Scout
  system      Manage Docker
  trust       Manage trust on Docker images
```

Check for the docker version with the following command:

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

C:\Users\Soham Satpute>
```

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 Atom editor and write the following contents into it to create a Ubuntu Linux container.

Script:

```
terraform {

required_providers {

docker = {

source = "kreuzwerker/docker"
version = "2.25.0"

   }
 }
}

provider "docker" {

host = "npipe:////.//pipe//docker_engine"
}

resource "docker_image" "ubuntu" {
name = "ubuntu:latest"
}

resource "docker_container" "foo" {
```

image = docker_image.ubuntu.image_id
name = "foo"
command = ["sleep", "3600"]
}

docker.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.25.0"
    }
  }
}

provider "docker" {
  host = "npipe:////.//pipe//docker_engine"
}

resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

resource "docker_container" "foo" {
  image   = docker_image.ubuntu.image_id
  name    = "foo"
  command = ["sleep", "3600"]
}
```

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

Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\soham22\Terraform Scripts\Docker> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of kreuzwerker/docker from the dependency lock file
- Using previously-installed kreuzwerker/docker v2.25.0

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:

```
PS C:\soham22\Terraform Scripts\Docker> terraform plan
docker_image.ubuntu: Refreshing state... [id=sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7ubun
tu:latest]
docker_container.foo: Refreshing state... [id=0200e3c246ed3069424b55d2bb9803da5442e04190c918629f4ee6a8879218fa]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # docker_container.foo must be replaced
-/+ resource "docker_container" "foo" {
      + bridge                                      = (known after apply)
      + container_logs                              = (known after apply)
      - cpu_shares                                  = 0 -> null
      - dns                                         = [] -> null
      - dns_opts                                    = [] -> null
      - dns_search                                  = [] -> null
      ~ entrypoint                                  = [] -> (known after apply)
      ~ env                                         = [] -> (known after apply)
      + exit_code                                   = (known after apply)
      ~ gateway                                     = "172.17.0.1" -> (known after apply)
      - group_add                                   = [] -> null
      ~ hostname                                    = "0200e3c246ed" -> (known after apply)
      ~ id                                          = "0200e3c246ed3069424b55d2bb9803da5442e04190c918629f4ee6a8879218fa"
 -> (known after apply)
      ~ init                                        = false -> (known after apply)
      ~ ip_address                                  = "172.17.0.2" -> (known after apply)
```

Step 5 :

Type terraform apply to apply changes:

```
PS C:\soham22\Terraform Scripts\Docker> 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                                     = [
          + "sleep",
          + "3600",
        ]
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + 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)
```

```
  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 images , Before Executing Apply

```
PS C:\soham22\Terraform Scripts\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
```

Docker images , After Executing Apply

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS C:\soham22\Terraform Scripts\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
ubuntu        latest    b1e9cef3f297   3 weeks ago      78.1MB
nginx         latest    39286ab8a5e1   5 weeks ago      188MB
hello-world   latest    d2c94e258dcb   16 months ago    13.3kB
PS C:\soham22\Terraform Scripts\Docker>
```

Step 6:

Execute Terraform destroy to delete the configuration ,which will automatically delete the Ubuntu Container:

```
PS C:\soham22\Terraform Scripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7ubuntu:latest]
docker_container.foo: Refreshing state... [id=877e2e0bb5cfd00634ddc5cde0f73121bc963959cb5c8f1ddf827471f515e4d5]

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
      - container_read_refresh_timeout_milliseconds = 15000 -> 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                                    = "877e2e0bb5cf" -> null
      - id                                          = "877e2e0bb5cfd00634ddc5cde0f73121bc963959cb5c8f1ddf827471f515e4d5" -> null
      - image                                       = "sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7" -> null
```

Docker images After Executing Destroy step:

```
Destroy complete! Resources: 2 destroyed.
PS C:\soham22\Terraform Scripts\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
nginx         latest    39286ab8a5e1   5 weeks ago     188MB
hello-world   latest    d2c94e258dcb   16 months ago   13.3kB
```