ADVANCE DEVOPS EXP6

Name:Prajjwal Pandey
Class:D15A
Roll No:33

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, Check for the docker version with the following command.

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\prajj>docker --version
Docker version 27.0.3, build 7d4bcd8

C:\Users\prajj>
```

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 followingcontents into it to create a Ubuntu Linux container. Script:

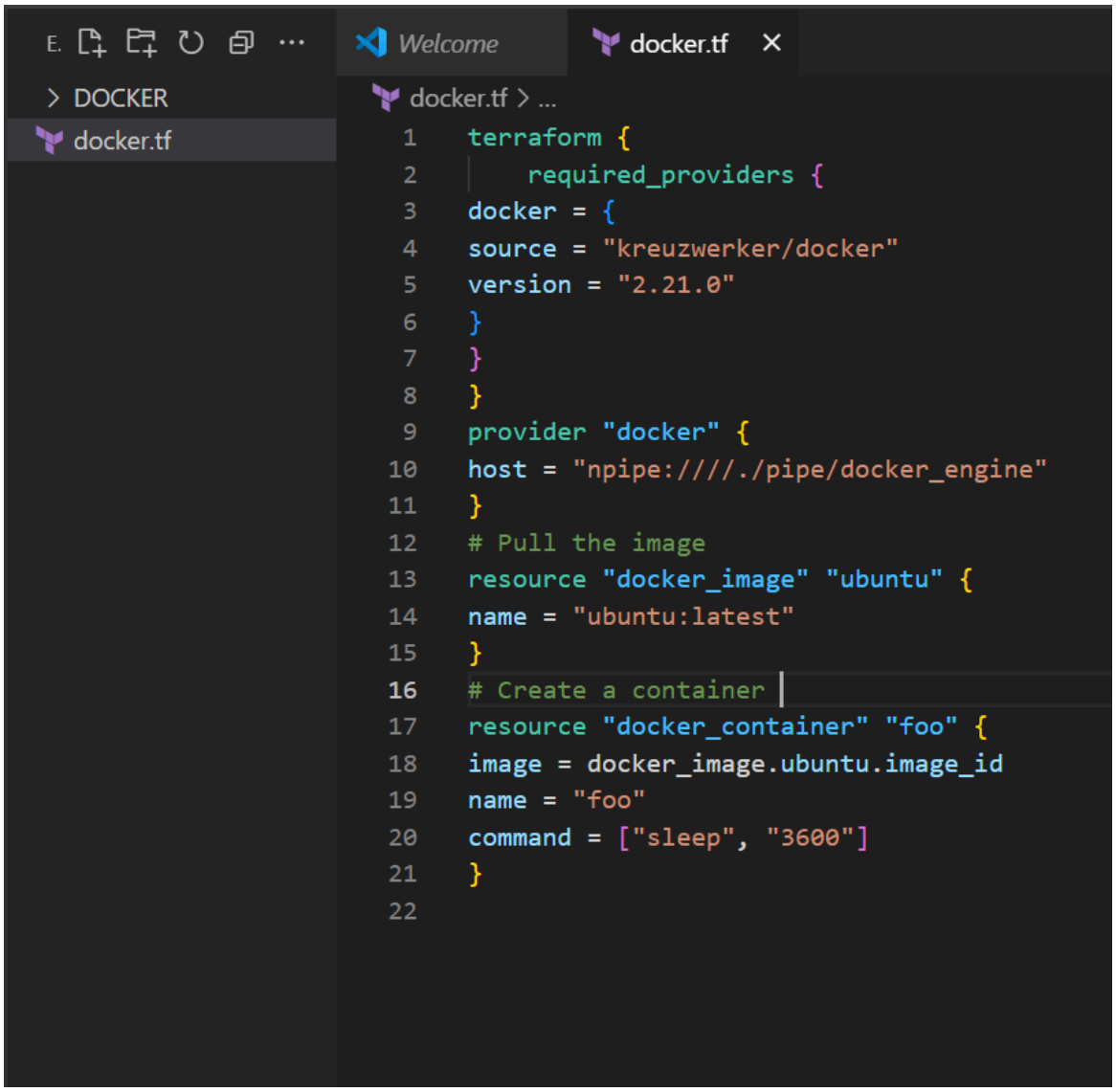
```
terraform { required_providers {
  docker = {
    source = "kreuzwerker/docker" version =
    "2.21.0"
  }
} }

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

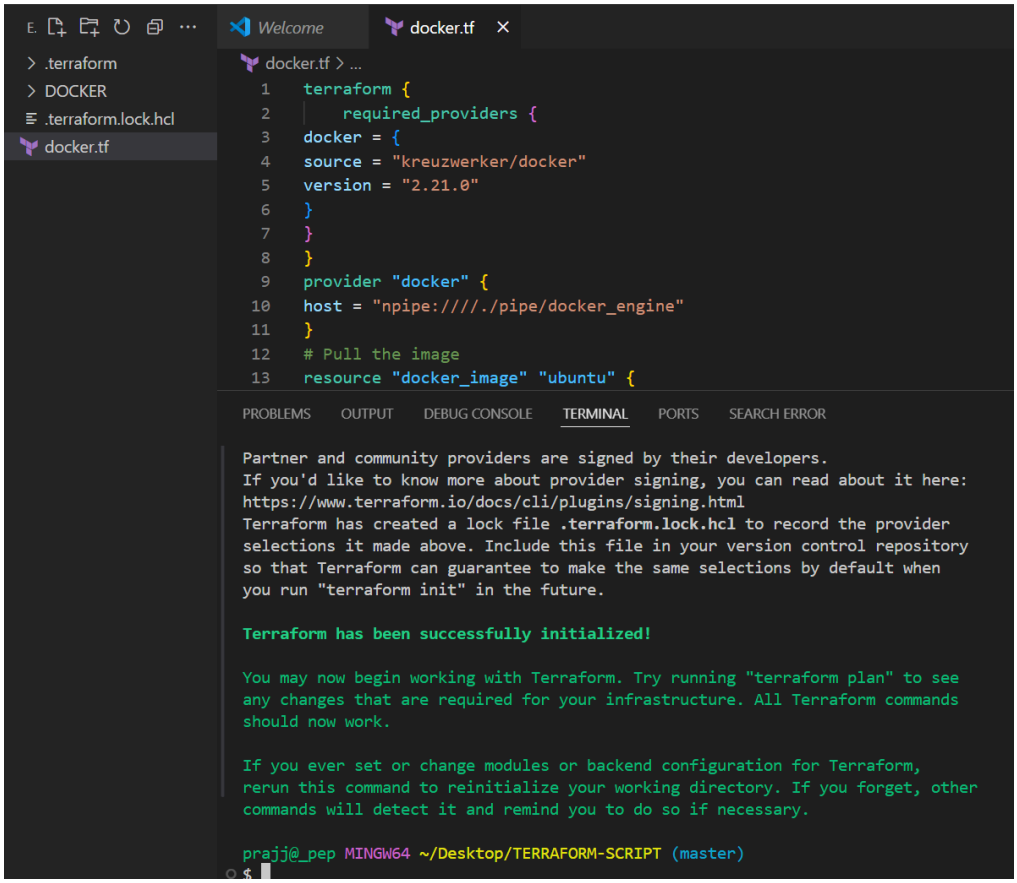
```
}

# 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"]
}
```



Step 3: Execute Terraform Init command to initialize the resources



Step 4: Execute Terraform plan to see the available resources

```
PS C:\Users\Admin\TerraformScripts\Docker> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the fol
  + 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
      + 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 :

```
docker_image.ubuntu: Creating...
docker_image.ubuntu: Creation complete after 9s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 2s [id=01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Docker images, Before Executing Apply step:

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

Docker images, After Executing Apply step:

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

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
  # 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=01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
```

Docker images After Executing Destroy step

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