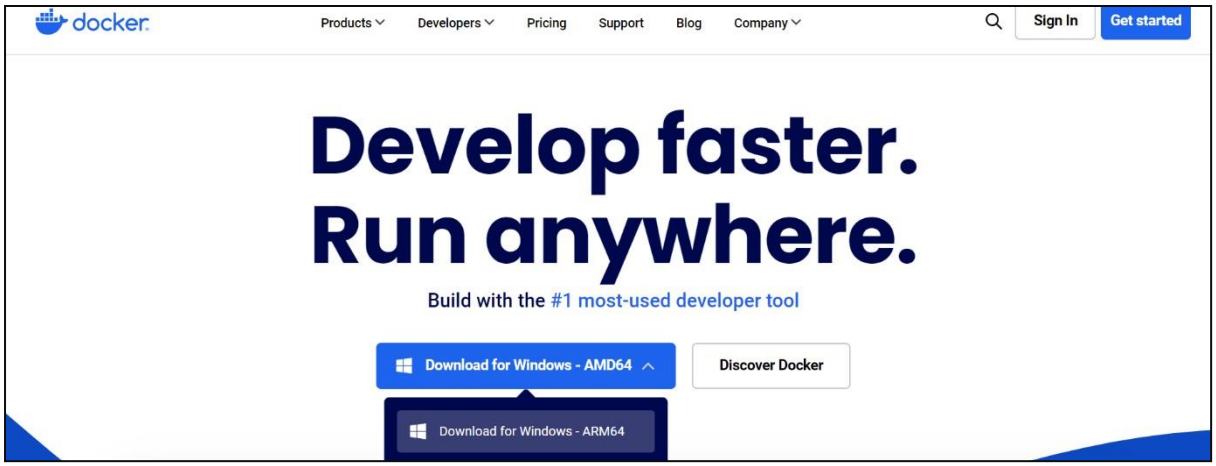
Advance DevOps

Experiment 6

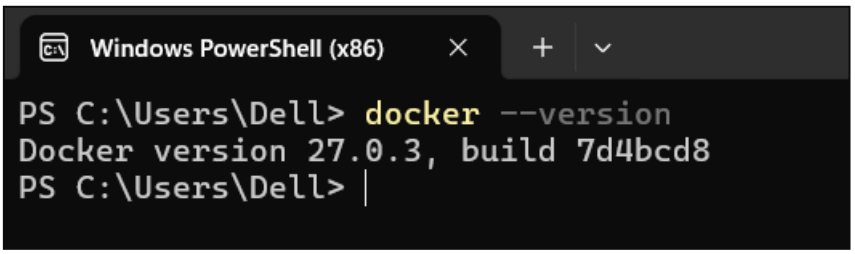
Creating docker image using terraform

Steps:

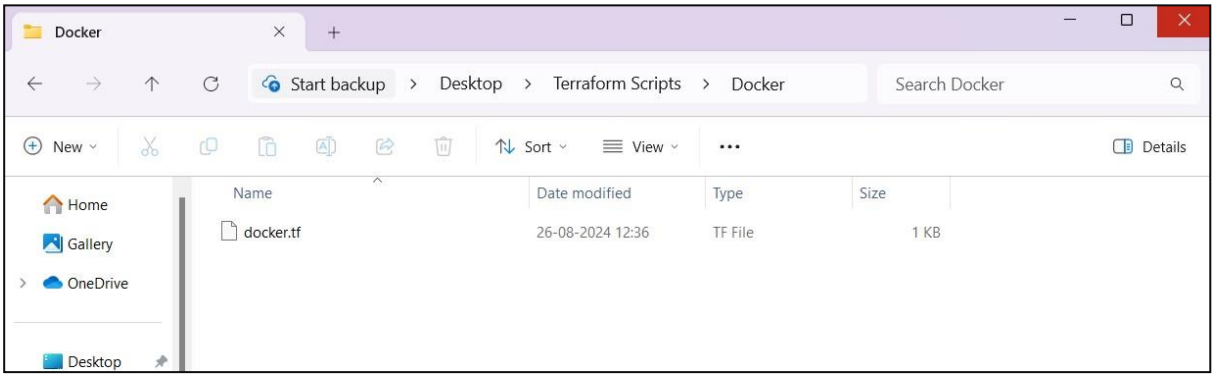
1. Firstly, we will have to download and install docker from its official website.



2. To check if docker has been correctly installed we can use the command “docker –version”. If it returns the version then installation has been successful.



3. Now we will create a folder “Terraform Scripts” and inside this folder we will create another folder “Docker” where our docker script will be saved. Using an editor create a file named docker and save it with .tf extension inside the above-mentioned directory structure.

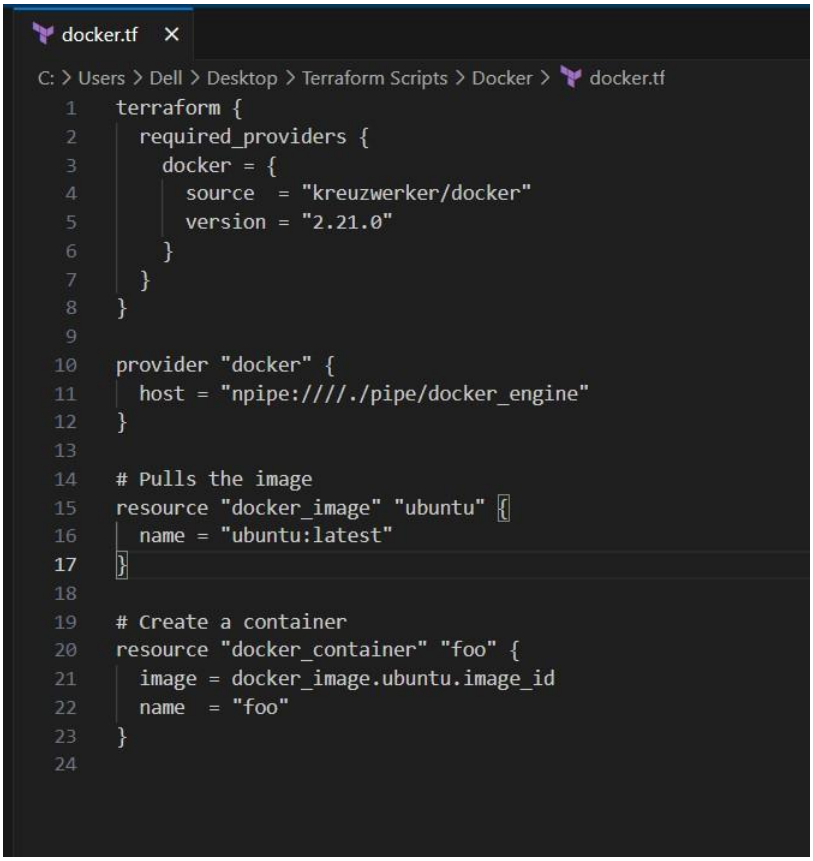


4. The docker.tf file will contain the following:

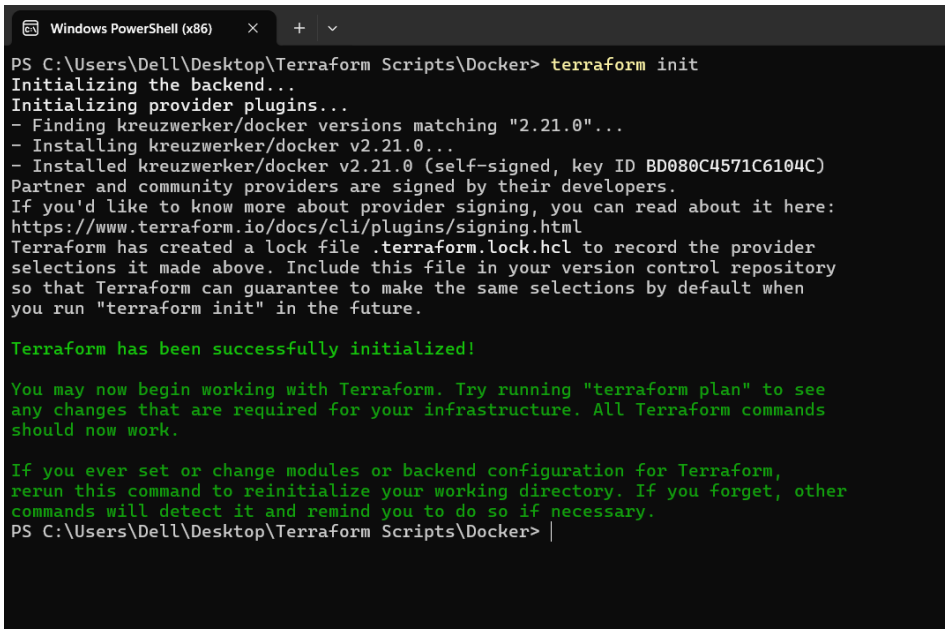
```
terraform {
  required_providers {
  docker = {
     source = "kreuzwerker/docker"
     version = "2.21.0"
    }
  }
}

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"
}
```

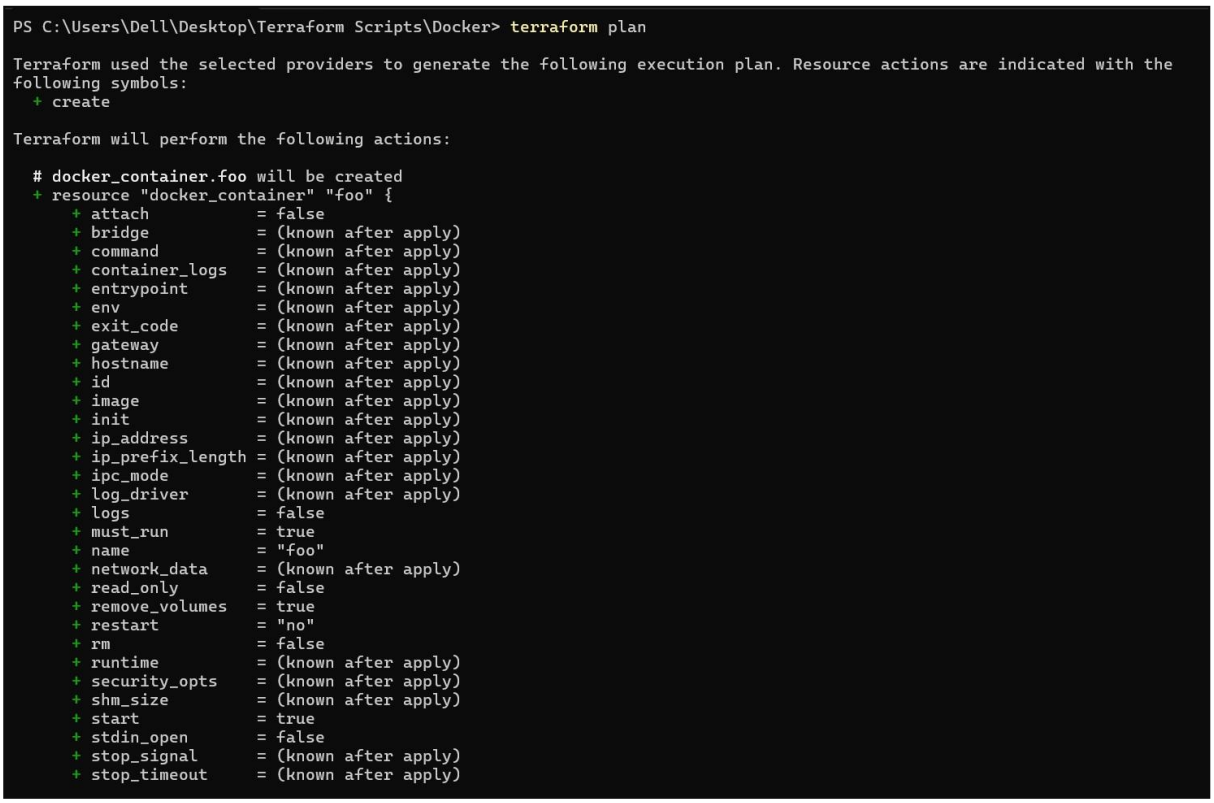
Ensure correct block formatting to avoid errors.



5. Now execute “terraform init” command in power shell. This command will initialize your working directory and download the necessary provider plugins. (Make sure you are in the “Docker” directory)



6. Next execute “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.



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

   In this step I got an error saying “container exited immediately” which indicates that the Docker container created by terraform configuration exited immediately after being started. In order to keep the container running I updated the script file with the following code:

   command = ["sleep", "infinity"] # Keeps the container running

```
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"
}

  resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep", "infinity"]  # Keeps the container running indefinitely
}
```

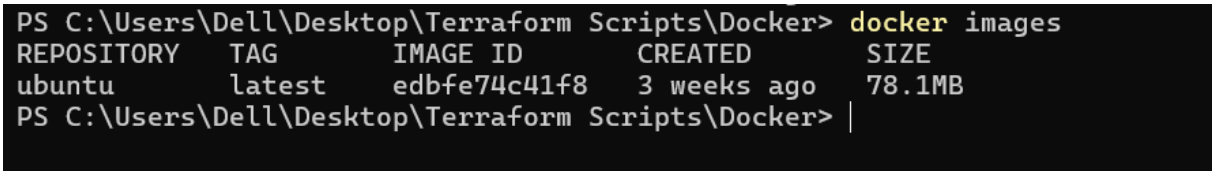
8. After updating the script, the error will be solved and we can again run apply command.

```
PS C:\Users\Dell\Desktop\Terraform Scripts\Docker> terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]

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",
          + "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:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
      + 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
```

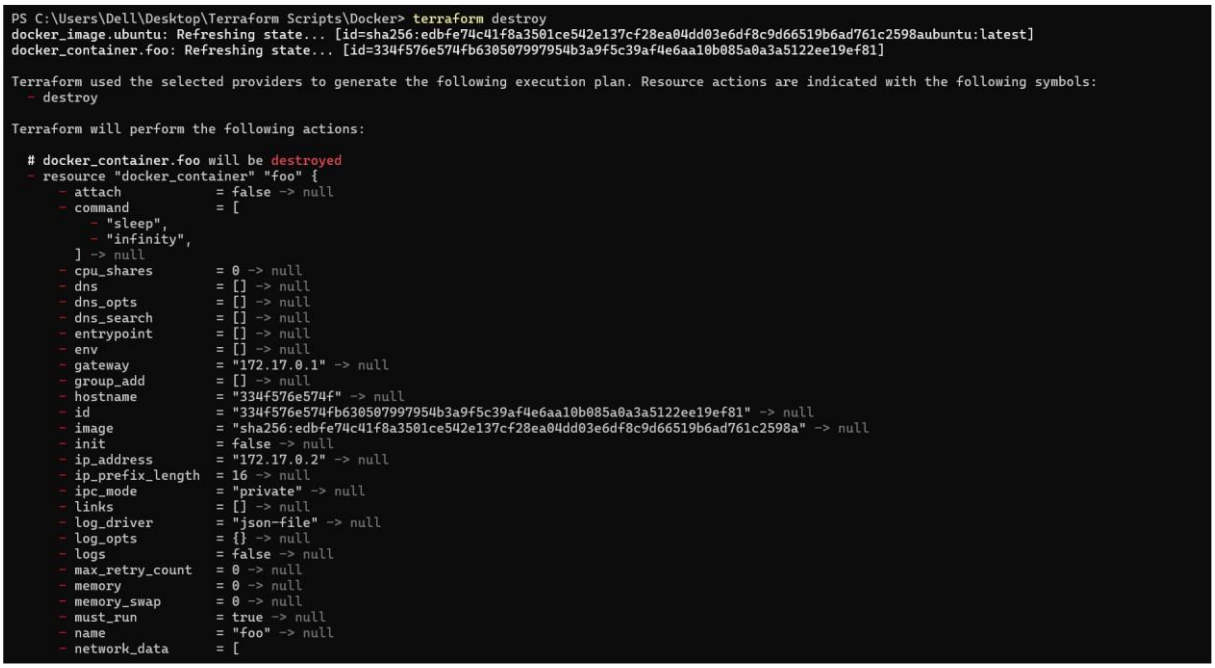
```
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + 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 0s [id=334f576e574fb630507997954b3a9f5c39af4e6aa10b085a0a3a5122ee19ef81]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

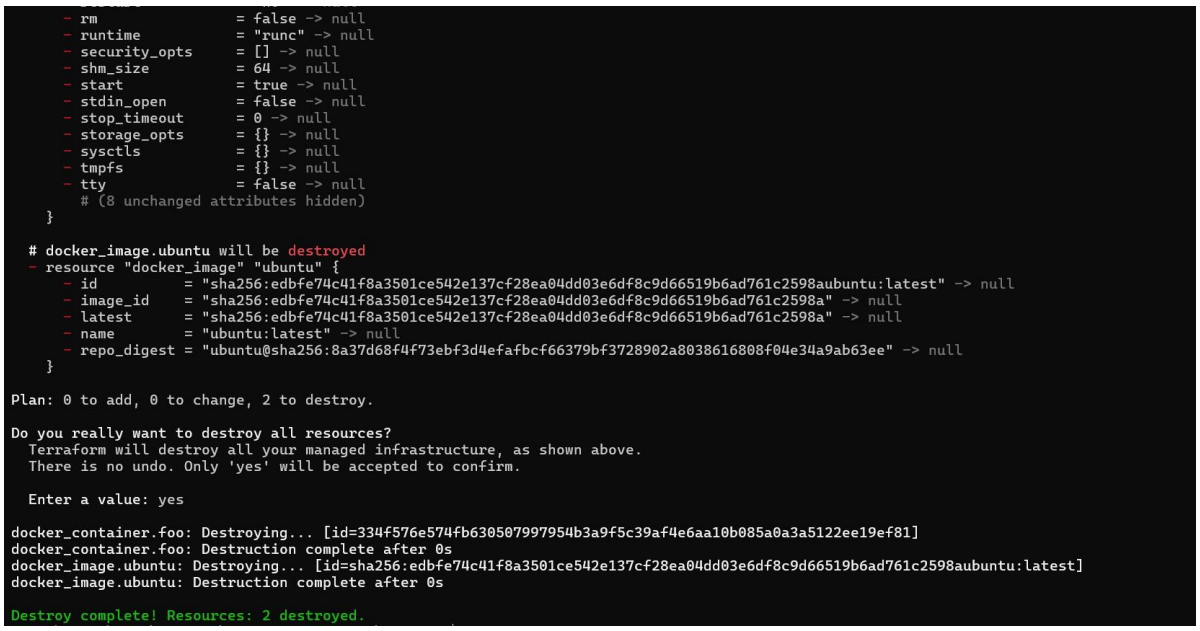
Docker image has been created.

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



10. Using terraform destroy we can destroy the container.





Now when we run docker images we will see that the container information is not visible that means it has been deleted successfully.