Rujuta Medhi **D15A-28**

EXPERIMENT NO. 6

Aim : To Build, change, and destroy AWS infrastructure Using Terraform (S3 bucket or Docker) .

Theory:

What is Terraform?

Terraform is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp. It allows you to define, provision, and manage infrastructure resources in a consistent and repeatable way. Terraform uses a high-level configuration language called HashiCorp Configuration Language (HCL) or JSON to describe the desired state of your infrastructure, and it can manage resources across various cloud providers, including AWS, Azure, Google Cloud, and others.

Benefits of Using Terraform with AWS

1.Multi-Cloud Support: Terraform can manage infrastructure across multiple cloud providers, including AWS. This allows you to create a consistent infrastructure environment across different clouds or migrate between them.

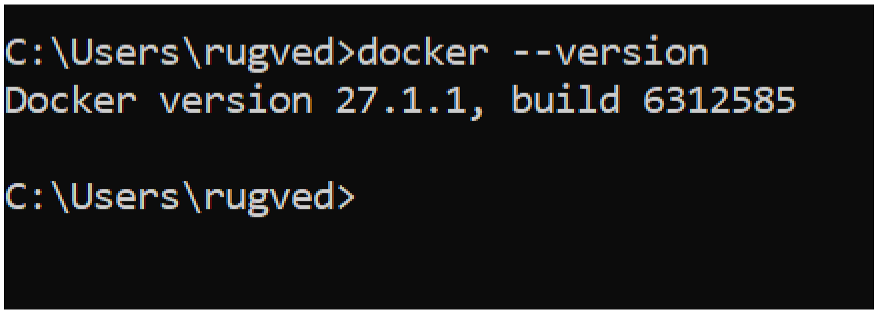
2.Infrastructure as Code: By using Terraform, you can define your AWS infrastructure as code. This makes it easier to version control your infrastructure, automate deployments, and collaborate with others.

3. Scalability and Flexibility: Terraform can manage infrastructure of any size, from small projects to large-scale, complex environments. It allows you to define reusable modules, which can be shared across different projects.

4. State Management: Terraform keeps track of the current state of your infrastructure in a state file. This allows Terraform to determine the actions required to achieve the desired state, ensuring that your infrastructure remains consistent.

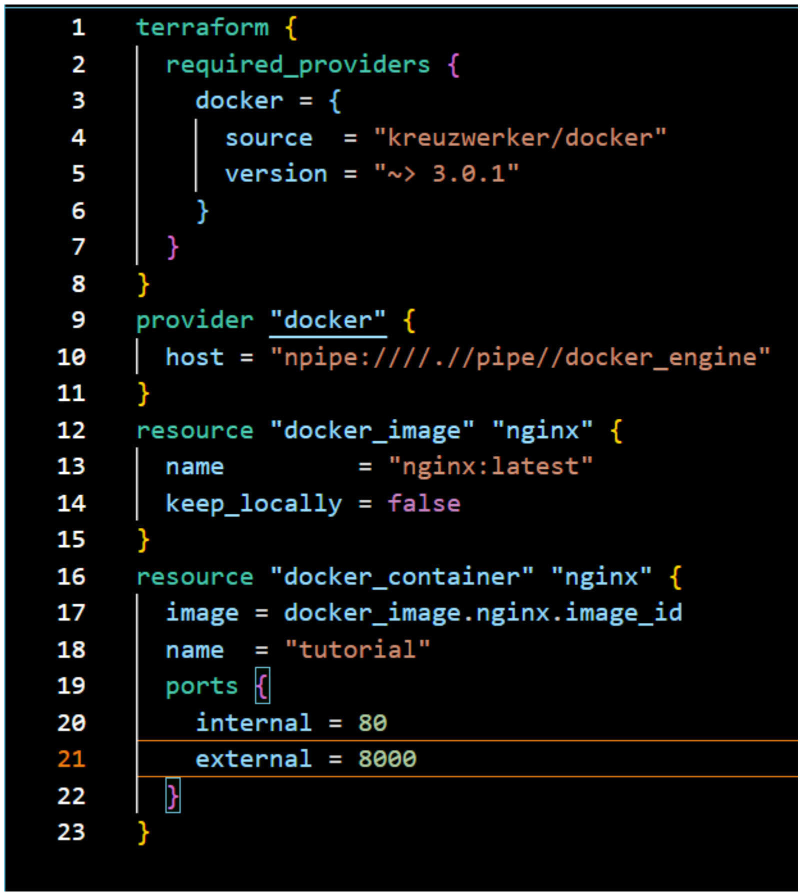
Implementation :

Step 1 : check docker installation and version



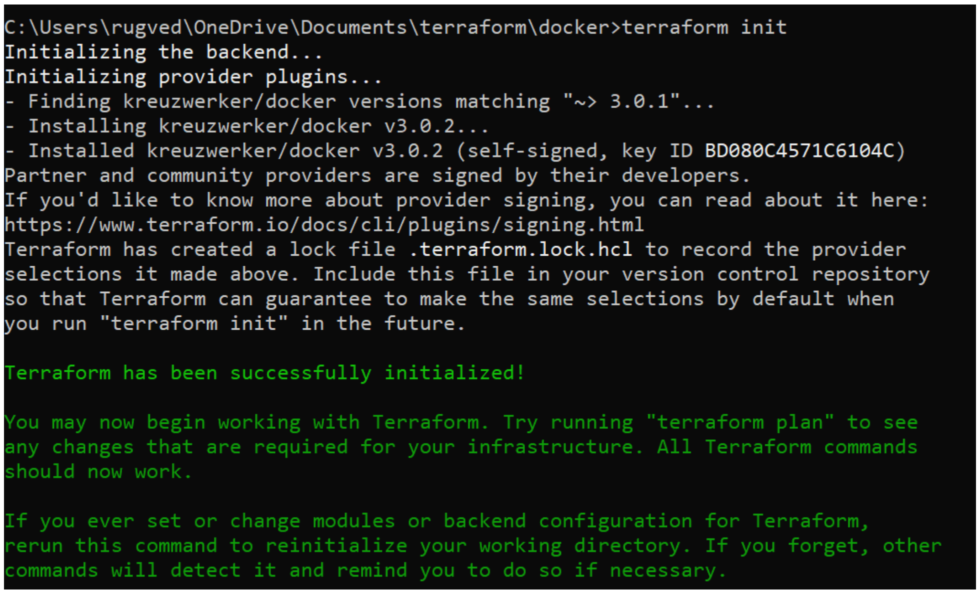
```
C:\Users\rugved>docker --version
Docker version 27.1.1, build 6312585

C:\Users\rugved>
```

Step 2 : create docker.tf file and write following code for terraform and docker



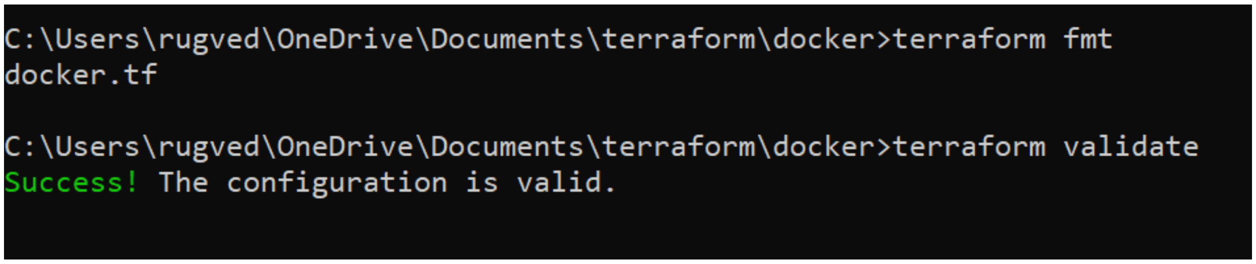
```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}
provider "docker" {
  host = "npipe:////.//pipe//docker_engine"
}
resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}
resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8000
  }
}
```

Step 3 : Type terraform init command to initialize terraform backend



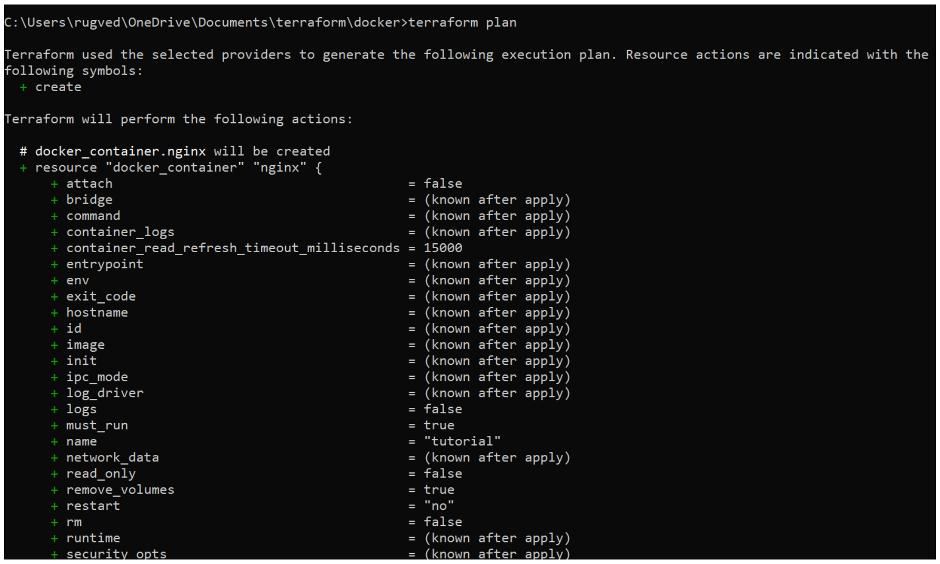
```
C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 3.0.1"...
- Installing kreuzwerker/docker v3.0.2...
- Installed kreuzwerker/docker v3.0.2 (self-signed, key ID BD080C4571C6104C)
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(EXTRA) : type terraform fmt and validate commands . The two Terraform commands – terraform validate and terraform fmt – are used to maintain a clean, error-free, and well-structured Terraform codebase.



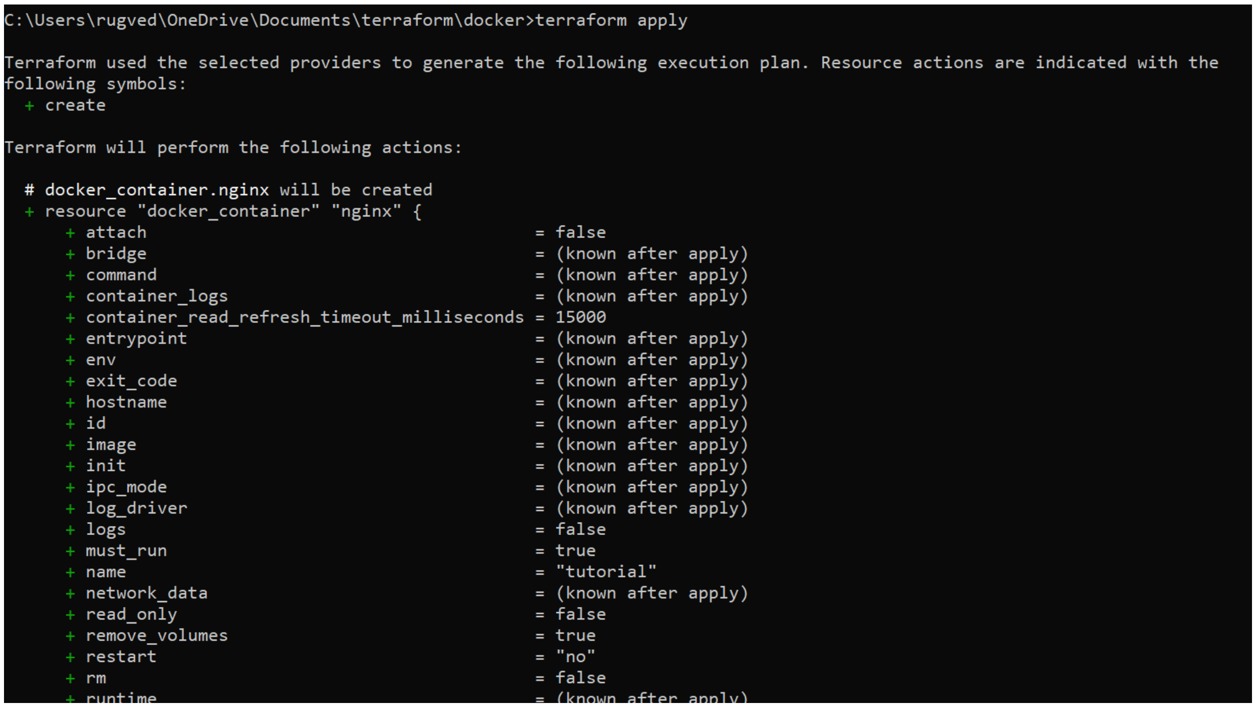
```
C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform fmt
docker.tf

C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform validate
Success! The configuration is valid.
```

Step 5 : Type Terraform plan command to create execution plan .

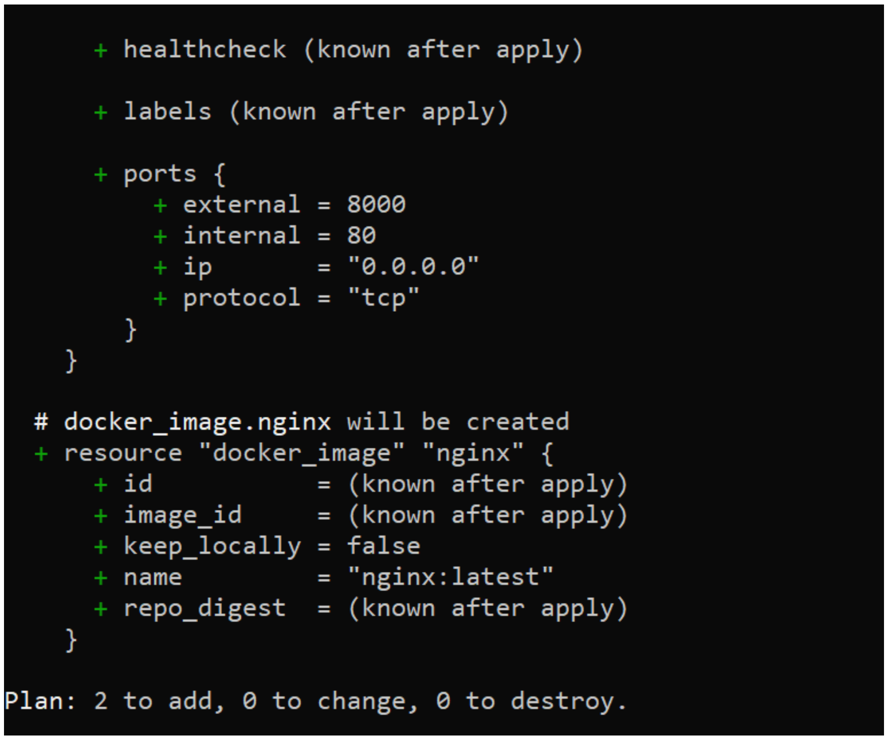


```
C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform plan

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.nginx will be created
  + resource "docker_container" "nginx" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
      + exit_code                                   = (known after apply)
      + hostname                                    = (known after apply)
      + id                                          = (known after apply)
      + image                                       = (known after apply)
      + init                                        = (known after apply)
      + ipc_mode                                    = (known after apply)
      + log_driver                                  = (known after apply)
      + logs                                        = false
      + must_run                                    = true
      + name                                        = "tutorial"
      + network_data                                = (known after apply)
      + read_only                                   = false
      + remove_volumes                              = true
      + restart                                     = "no"
      + rm                                          = false
      + runtime                                     = (known after apply)
      + security_opts                               = (known after apply)
```

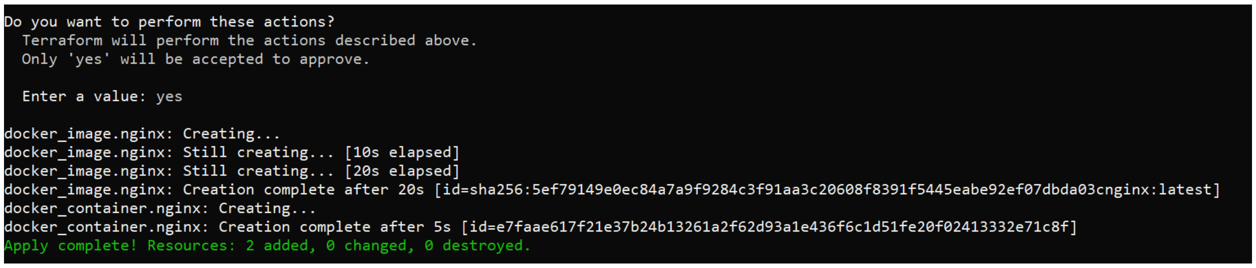
Step 6 : Type terraform apply to apply changes .



```
C:\Users\rugved\OneDrive\Documents\terraform\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.nginx will be created
  + resource "docker_container" "nginx" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
      + exit_code                                   = (known after apply)
      + hostname                                    = (known after apply)
      + id                                          = (known after apply)
      + image                                       = (known after apply)
      + init                                        = (known after apply)
      + ipc_mode                                    = (known after apply)
      + log_driver                                  = (known after apply)
      + logs                                        = false
      + must_run                                    = true
      + name                                        = "tutorial"
      + network_data                                = (known after apply)
      + read_only                                   = false
      + remove_volumes                              = true
      + restart                                     = "no"
      + rm                                          = false
      + runtime                                     = (known after apply)
```

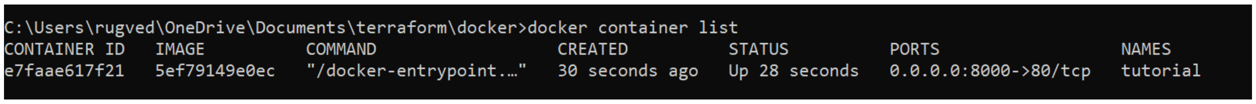


```
      + healthcheck (known after apply)

      + labels (known after apply)

      + ports {
          + external = 8000
          + internal = 80
          + ip       = "0.0.0.0"
          + protocol = "tcp"
        }
    }

  # docker_image.nginx will be created
  + resource "docker_image" "nginx" {
      + id           = (known after apply)
      + image_id     = (known after apply)
      + keep_locally = false
      + name         = "nginx:latest"
      + repo_digest  = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.
```

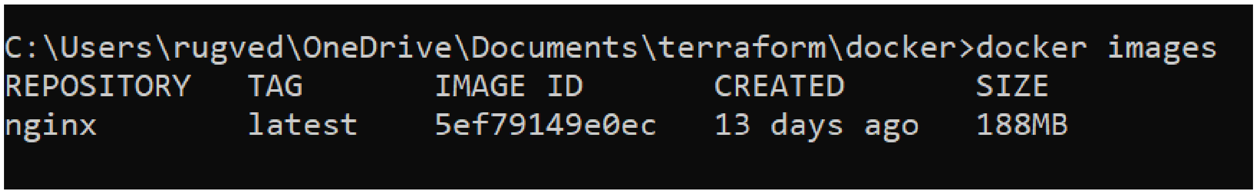


```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

docker_image.nginx: Creating...
docker_image.nginx: Still creating... [10s elapsed]
docker_image.nginx: Still creating... [20s elapsed]
docker_image.nginx: Creation complete after 20s [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 5s [id=e7faae617f21e37b24b13261a2f62d93a1e436f6c1d51fe20f02413332e71c8f]
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Step 7 : Docker container before and after step 6 execution

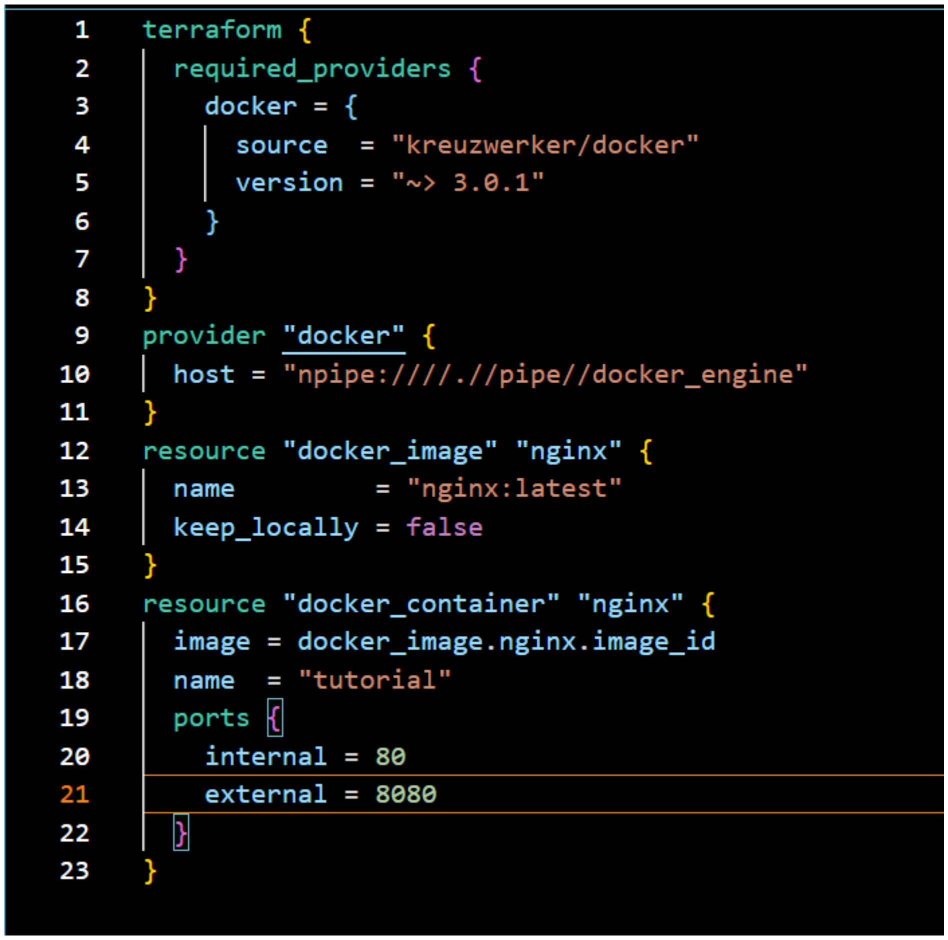


```
C:\Users\rugved\OneDrive\Documents\terraform\docker>docker container list
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                  NAMES
e7faae617f21   5ef79149e0ec   "/docker-entrypoint.…"   30 seconds ago   Up 28 seconds   0.0.0.0:8000->80/tcp   tutorial
```

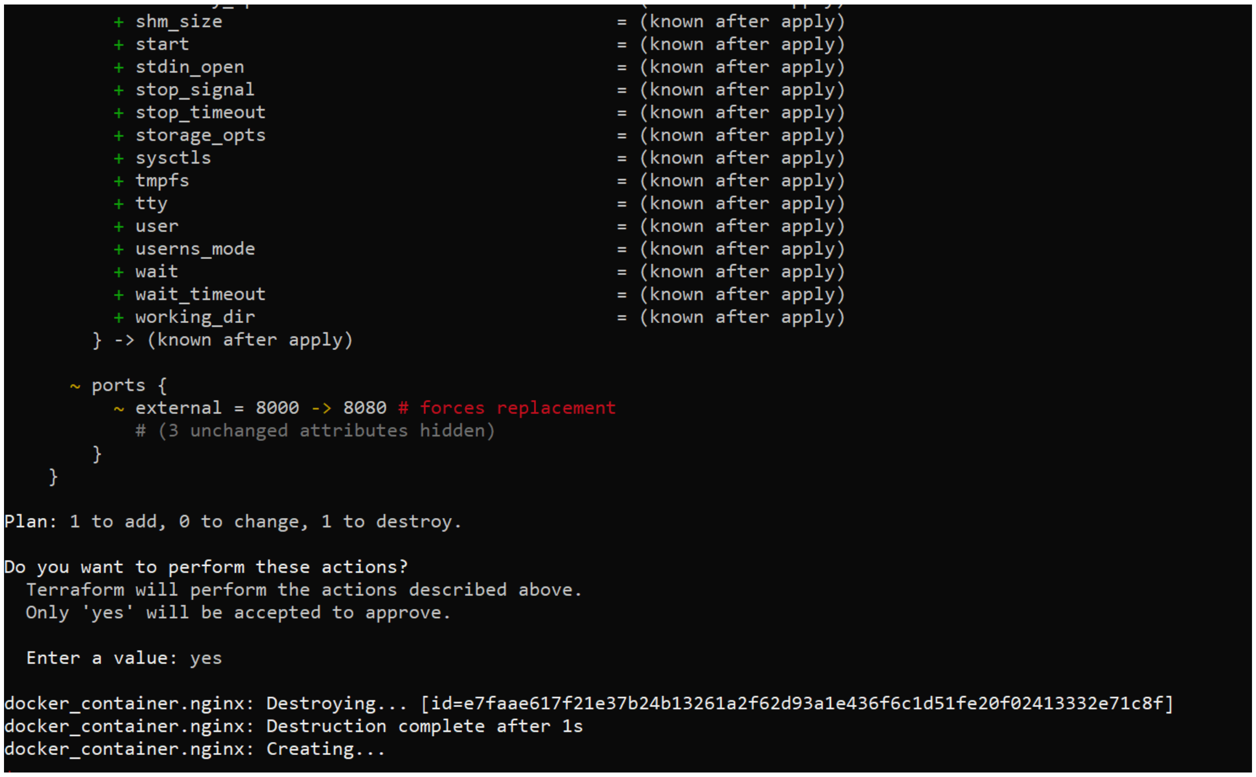


```
C:\Users\rugved\OneDrive\Documents\terraform\docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
nginx        latest    5ef79149e0ec   13 days ago    188MB
```

Step 8 (EXTRA) : Execution of change .

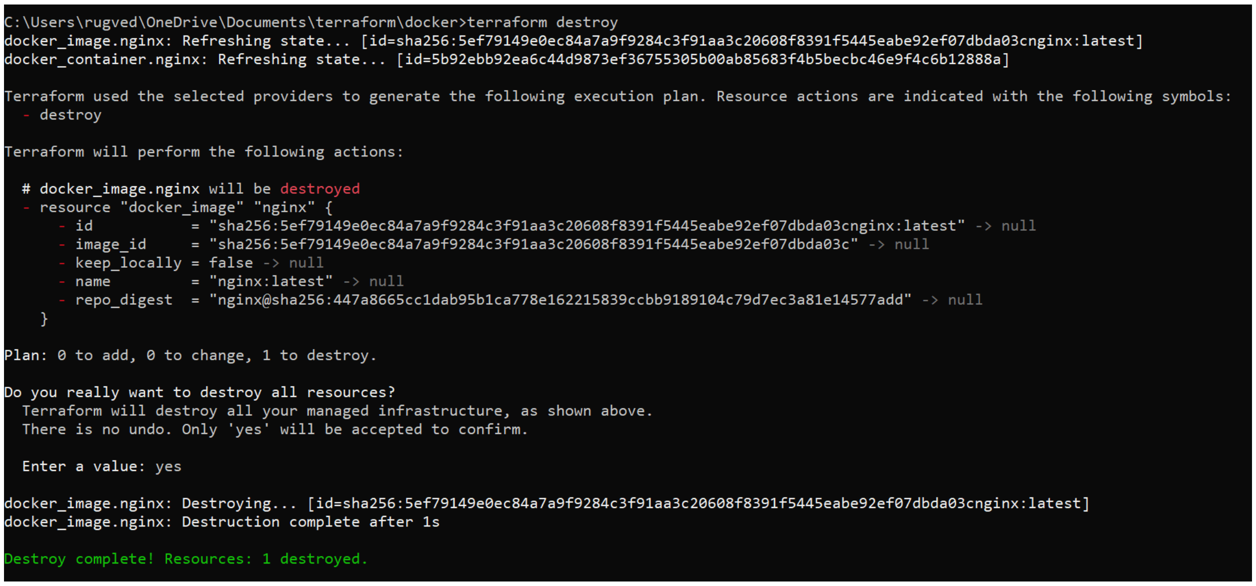


```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}
provider "docker" {
  host = "npipe:////.//pipe//docker_engine"
}
resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}
resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8080
  }
}
```



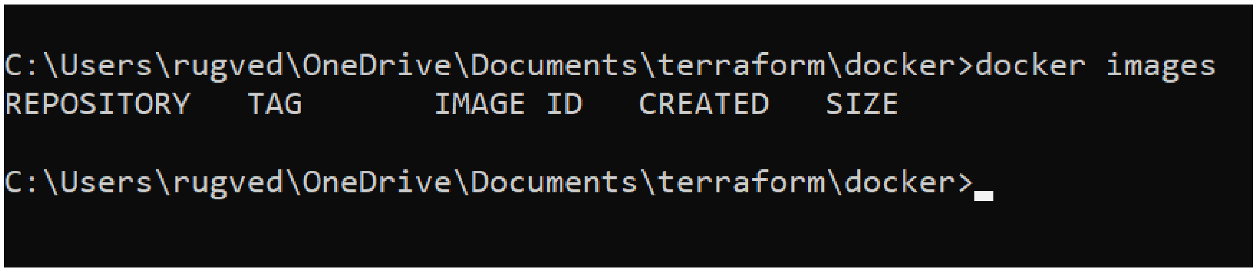
```
          + shm_size                                        = (known after apply)
          + start                                           = (known after apply)
          + stdin_open                                      = (known after apply)
          + stop_signal                                     = (known after apply)
          + stop_timeout                                    = (known after apply)
          + storage_opts                                    = (known after apply)
          + sysctls                                         = (known after apply)
          + tmpfs                                           = (known after apply)
          + tty                                             = (known after apply)
          + user                                            = (known after apply)
          + userns_mode                                     = (known after apply)
          + wait                                            = (known after apply)
          + wait_timeout                                    = (known after apply)
          + working_dir                                     = (known after apply)
        } -> (known after apply)

      ~ ports {
          ~ external = 8000 -> 8080 # forces replacement
            # (3 unchanged attributes hidden)
        }
    }

Plan: 1 to add, 0 to change, 1 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.nginx: Destroying... [id=e7faae617f21e37b24b13261a2f62d93a1e436f6c1d51fe20f02413332e71c8f]
docker_container.nginx: Destruction complete after 1s
docker_container.nginx: Creating...
```

Step 9 : terraform destroy to destroy infrastructure.



```
C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform destroy
docker_image.nginx: Refreshing state... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_container.nginx: Refreshing state... [id=5b92ebb92ea6c44d9873ef36755305b00ab85683f4b5becbc46e9f4c6b12888a]

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_image.nginx will be destroyed
  - resource "docker_image" "nginx" {
      - id           = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest" -> null
      - image_id     = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c" -> null
      - keep_locally = false -> null
      - name         = "nginx:latest" -> null
      - repo_digest  = "nginx@sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577add" -> null
    }

Plan: 0 to add, 0 to change, 1 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_image.nginx: Destroying... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_image.nginx: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
```

Step 10 : Docker after destroy command.



```
C:\Users\rugved\OneDrive\Documents\terraform\docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

C:\Users\rugved\OneDrive\Documents\terraform\docker>
```