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

1. Check docker installation by running the command “docker” and “docker --version”.

```
C:\Windows\system32>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
  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 image
  scout*      Docker Scout
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Swarm Commands:
  swarm       Manage Swarm
```

```
C:\Windows\system32>docker --version
Docker version 27.0.3, build 7d4bcd8

C:\Windows\system32>
```

2. Create a folder in your system named “terraformScripts”(Do not use terraform as name as it may recognize it as a keyword). Inside it create a subfolder docker. In this create a file docker.tf and type the following code in it.

docker.tf

```
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"]
}
```

3. Run the windows powershell as administrator. Navigate to the docker folder created in the above step. Run the terraform init command.

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\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.
```

4. Terraform plan command is used to create a execution plan and see the resources.

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\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.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
```

5. Docker image is command used to see all the images currently created in docker desktop

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

6. Terraform apply command is used to execute the steps listed in terraform plan.

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]
docker_container.foo: Refreshing state... [id=4de644086273a692820f646431610cae2f095755228f61eafc2bb712b2766040]

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)
      + 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)
```

7. Terraform destroy is used to destroy all resources that are currently being managed by terraform configuration.

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]
docker_container.foo: Refreshing state... [id=6d649285ef1d861ad451273401bc2771fe4e0be9b78f232aa64230ca0f58d36e]

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
        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          = "6d649285ef1d" -> null
        id                = "6d649285ef1d861ad451273401bc2771fe4e0be9b78f232aa64230ca0f58d36e" -> null
        image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
        init              = false -> null
        ip_address        = "172.17.0.2" -> null
        ip_prefix_length  = 16 -> null
        ipc_mode          = "private" -> null
        links             = [] -> null
        log_driver        = "json-file" -> null
        log_opts          = {} -> null
        logs              = false -> null
        max_retry_count   = 0 -> null
        memory            = 0 -> null
        memory_swap       = 0 -> null
        must_run          = true -> null
        name              = "foo" -> null
        network_data      = [
            {
```

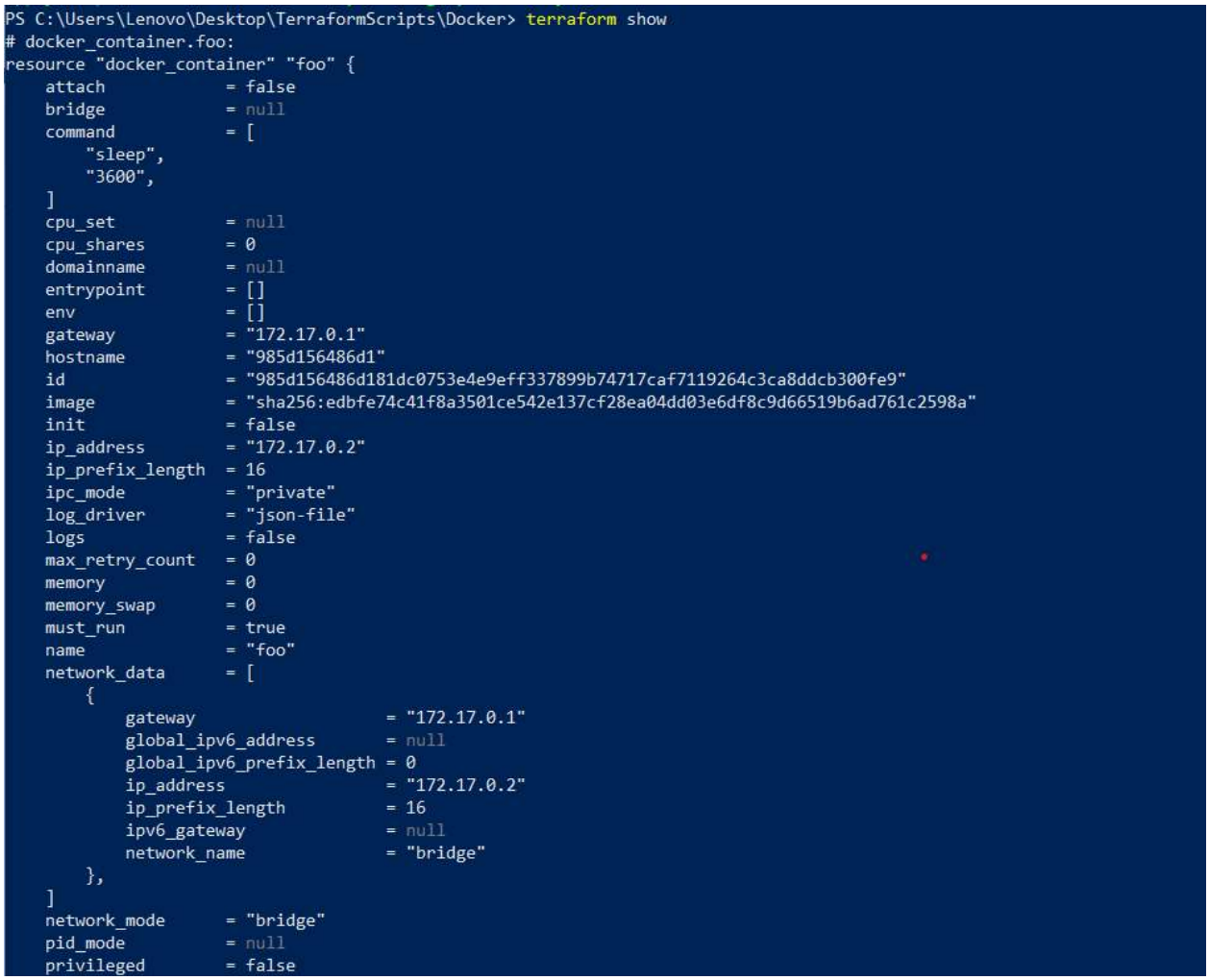
```
  # docker_image.ubuntu will be destroyed
    resource "docker_image" "ubuntu" {
        id          = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
        image_id    = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> 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=6d649285ef1d861ad451273401bc2771fe4e0be9b78f232aa64230ca0f58d36e]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:lat
est]
docker_image.ubuntu: Destruction complete after 0s
```

```
Destroy complete! Resources: 2 destroyed.
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

8. Terraform validate is a command used to check for syntax and other errors in terraform configuration files.

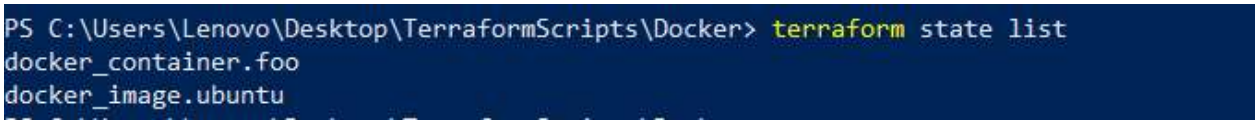
```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform validate
Success! The configuration is valid.
```

9. The terraform show command in Terraform is used to display information about the state of resources managed by Terraform.



```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform show
# docker_container.foo:
resource "docker_container" "foo" {
    attach            = false
    bridge            = null
    command           = [
        "sleep",
        "3600",
    ]
    cpu_set           = null
    cpu_shares        = 0
    domainname        = null
    entrypoint        = []
    env               = []
    gateway           = "172.17.0.1"
    hostname          = "985d156486d1"
    id                = "985d156486d181dc0753e4e9eff337899b74717caf7119264c3ca8ddcb300fe9"
    image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
    init              = false
    ip_address        = "172.17.0.2"
    ip_prefix_length  = 16
    ipc_mode          = "private"
    log_driver        = "json-file"
    logs              = false
    max_retry_count   = 0
    memory            = 0
    memory_swap       = 0
    must_run          = true
    name              = "foo"
    network_data      = [
        {
            gateway                   = "172.17.0.1"
            global_ipv6_address       = null
            global_ipv6_prefix_length = 0
            ip_address                = "172.17.0.2"
            ip_prefix_length          = 16
            ipv6_gateway              = null
            network_name              = "bridge"
        },
    ]
    network_mode      = "bridge"
    pid_mode          = null
    privileged        = false
```

10. The terraform state list command is used to list all the resources currently managed by your Terraform state



```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform state list
docker_container.foo
docker_image.ubuntu
```