

Name: Atharva Patil**Class: D15C****Roll No: 39****Experiment No.: 6****Implementation:****A. Creating docker image using terraform****Prerequisite:**

1) Download and Install Docker Desktop from <https://www.docker.com/>

Step 1: Check the docker functiona

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

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\sushmita> 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 image
scout*   Docker Scout
system   Manage Docker
trust    Manage trust on Docker images
volume   Manage volumes

Swarm Commands:
config   Manage Swarm configs
```

```
For more help on how to use Docker, head to https://docs.docker.com/go/guides/
PS C:\Users\sushmita> docker --version
Docker version 27.1.1, build 6312585
PS C:\Users\sushmita> █
```

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 = "2.21.0"
}
}
}
```

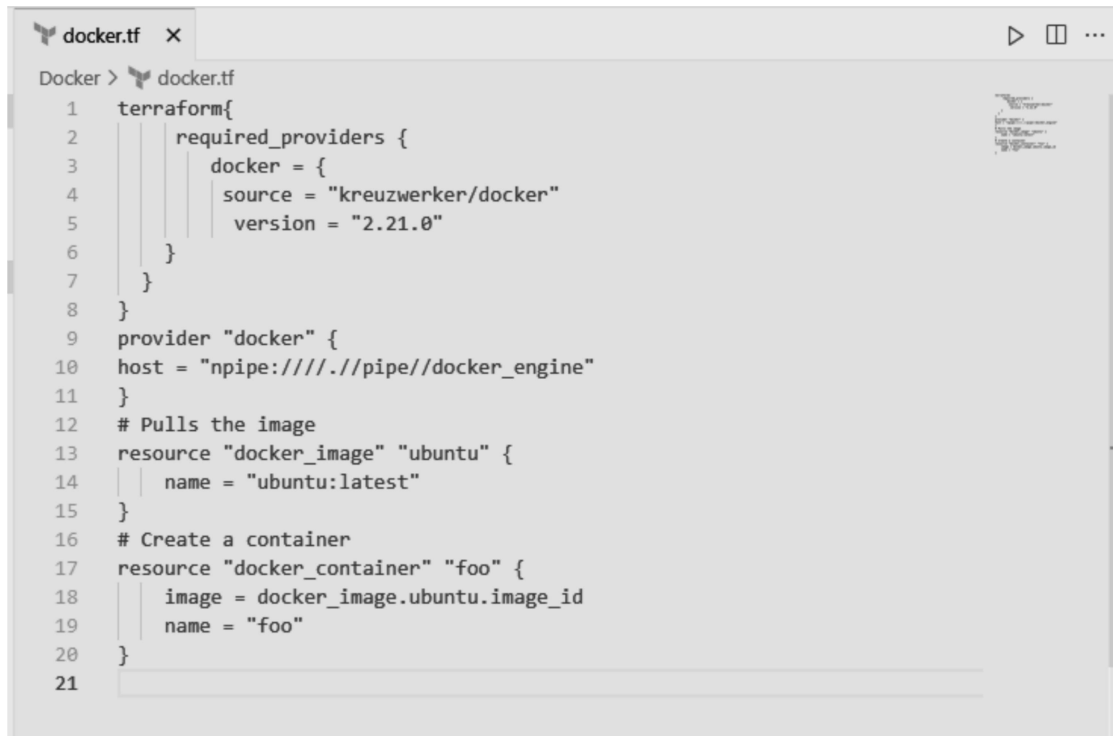
```
provider "docker" {
  host = "npipe:////./pipe//docker_engine"
}
```

Pulls the image

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

Create a container

```
resource "docker_container" "foo"
{ image =
  docker_image.ubuntu.image_idname =
  "foo"
}
```



```

1 terraform{
2     required_providers {
3         docker = {
4             source = "kreuzwerker/docker"
5             version = "2.21.0"
6         }
7     }
8 }
9 provider "docker" {
10     host = "npipe:////./pipe/docker_engine"
11 }
12 # Pulls the image
13 resource "docker_image" "ubuntu" {
14     name = "ubuntu:latest"
15 }
16 # Create a container
17 resource "docker_container" "foo" {
18     image = docker_image.ubuntu.image_id
19     name = "foo"
20 }
21

```

Step 3: Execute Terraform Init command to initialize the resources

```

PS C:\Users\sushmita\Desktop\TerraformScript\Docke> 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.
PS C:\Users\sushmita\Desktop\TerraformScript\Docke>

```

Step 4: Execute Terraform plan to see the available resources

```
PS C:\Users\sushmita\Desktop\TerraformScript\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         = (known after apply)
  + container_logs  = (known after apply)
  + endpoint        = (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.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

PS C:\Users\sushmita\Desktop\TerraformScript\Docker>

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 : “**terraform 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 5s [id=2c95700bb1f1605e21836e7f7292718f1ffdcbb296d3db0f91ce70e1511afc6]  
  
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.  
PS C:\Users\sushmita\Desktop\TerraformScript\Docke> █
```

Docker images, Before Executing Apply step:

```
PS C:\Users\sushmita\Desktop\TerraformScript\Docke> docker images  
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE  
PS C:\Users\sushmita\Desktop\TerraformScript\Docke> █
```

Docker images, After Executing Apply step:

```
PS C:\Users\sushmita\Desktop\TerraformScript\Docke> docker images  
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE  
ubuntu        latest    edbfe74c41f8   3 weeks ago    78.1MB  
PS C:\Users\sushmita\Desktop\TerraformScript\Docke> █
```

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

```

CONTAINER ID   IMAGE      COMMAND                  CREATED       STATUS       PORTS       NAMES
PS C:\Users\sushmita\Desktop\TerraformScript\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a:latest]

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.ubuntu will be destroyed
resource "docker_image" "ubuntu" {
  id          = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a: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, 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.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a:latest]
docker_image.ubuntu: Destruction complete after 2s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\sushmita\Desktop\TerraformScript\Docker>

```

Docker images After Executing Destroy step

```

PS C:\Users\sushmita\Desktop\TerraformScript\Docker> docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
PS C:\Users\sushmita\Desktop\TerraformScript\Docker>

```