

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 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"
```

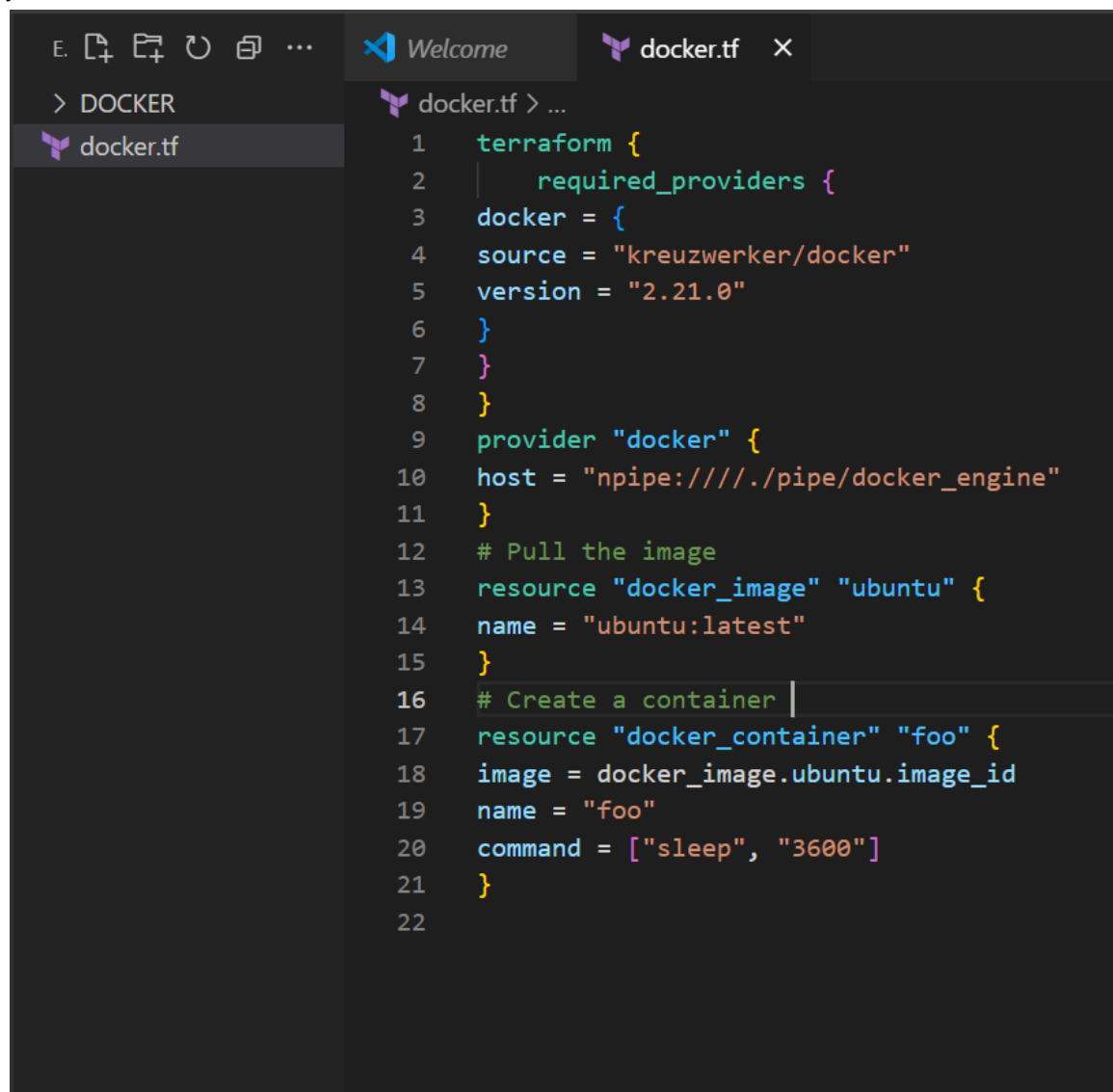
```
}
```

```
# Pull the image resource
```

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

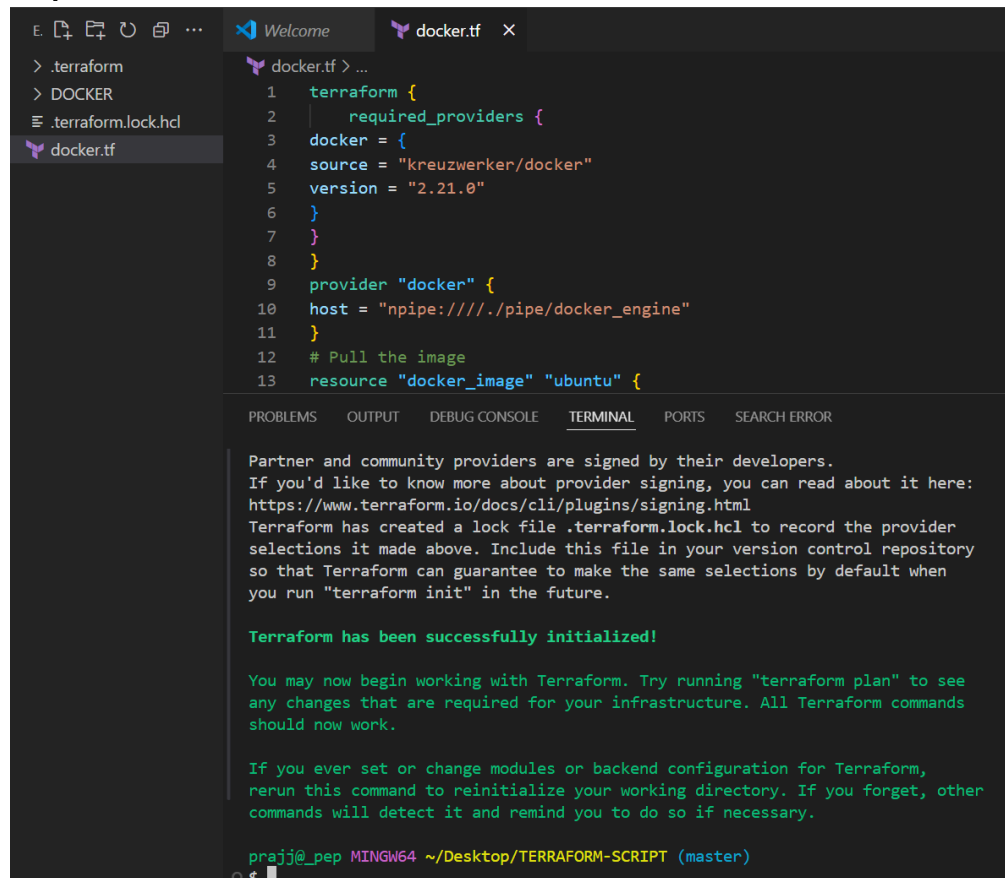
```
# Create a container resource
```

```
"docker_container" "foo" {  
  image = docker_image.ubuntu.image_id  
  name = "foo" command = ["sleep", "3600"]  
}
```



```
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 # Pull 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  command = ["sleep", "3600"]  
21 }  
22
```

Step 3: Execute Terraform Init command to initialize the resources



The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files `.terraform`, `DOCKER`, `.terraform.lock.hcl`, and `docker.tf`. The `docker.tf` file is selected. The code editor displays the contents of `docker.tf`, which is a Terraform configuration for the Docker provider and a resource. Below the code editor, the `TERMINAL` tab is active, showing the output of the `terraform init` command. The output indicates that Terraform has been successfully initialized and provides instructions on how to use the configuration.

```
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 # Pull the image
13 resource "docker_image" "ubuntu" {
```

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.

prajj@_pep MINGW64 ~/Desktop/TERRAFORM-SCRIPT (master)
\$

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 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
  + 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
------------	-----	----------	---------	------