

ADVANCED DEVOPS EXP 6

AIM: Creating docker image using terraform

Step 1: Check the docker functionality

```
CA Command Prompt

C:\Users\Shravani>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
```

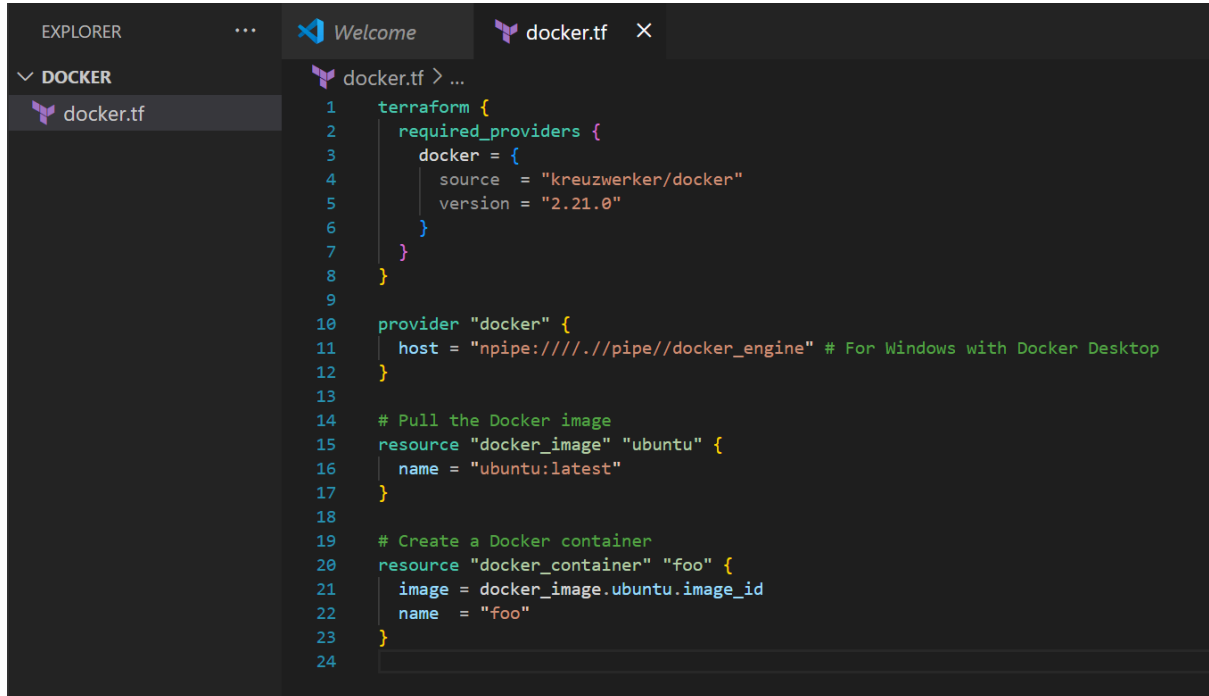
```
CA Command Prompt

Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

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

C:\Users\Shravani>
```

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.

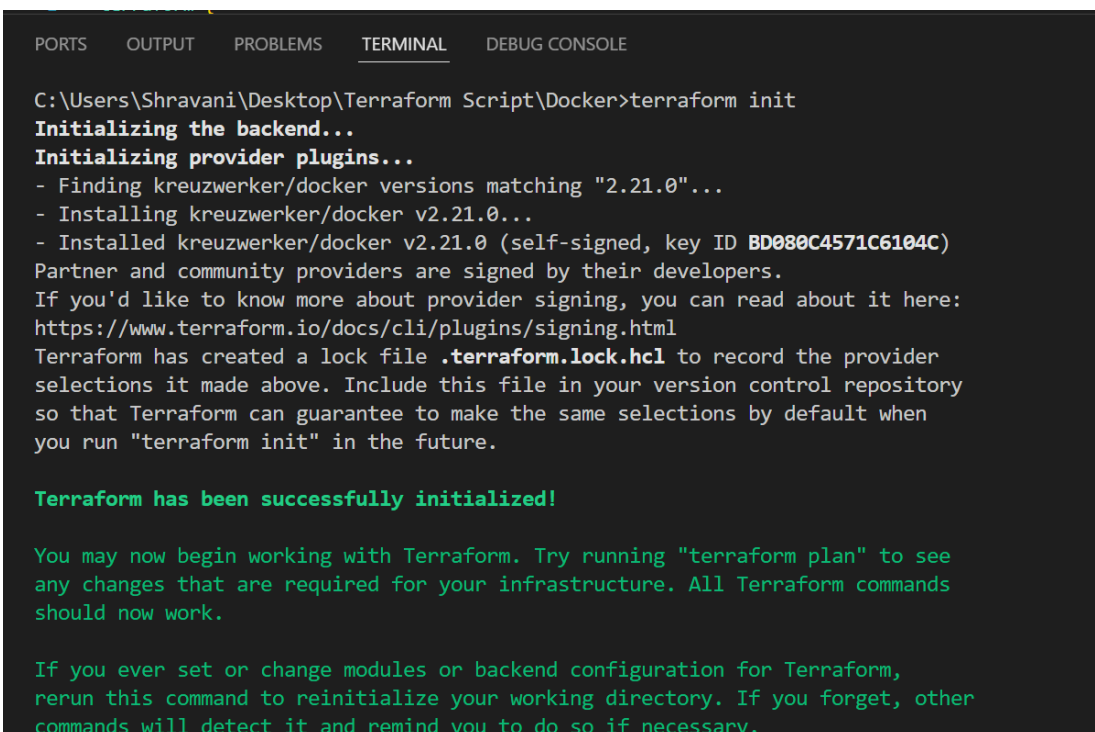


```

1 terraform {
2   required_providers {
3     docker = {
4       source = "kreuzwerker/docker"
5       version = "2.21.0"
6     }
7   }
8 }
9
10 provider "docker" {
11   host = "npipe:////./pipe/docker_engine" # For Windows with Docker Desktop
12 }
13
14 # Pull the Docker image
15 resource "docker_image" "ubuntu" {
16   name = "ubuntu:latest"
17 }
18
19 # Create a Docker container
20 resource "docker_container" "foo" {
21   image = docker_image.ubuntu.image_id
22   name = "foo"
23 }
24

```

Step 3: Execute Terraform Init command to initialize the resources



```

PORTS  OUTPUT  PROBLEMS  TERMINAL  DEBUG CONSOLE

C:\Users\Shravani\Desktop\Terraform Script\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.

```

Step 4: Execute Terraform plan to see the available resources

```

PORTS  OUTPUT  PROBLEMS  TERMINAL  DEBUG CONSOLE

C:\Users\Shravani\Desktop\Terraform Script\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)
  + 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)

```

```

PORTS  OUTPUT  PROBLEMS  TERMINAL  DEBUG CONSOLE

+ 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.

C:\Users\Shravani\Desktop\Terraform Script\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”

Docker images, Before Executing Apply step:

```
C:\Users\Shravani\Desktop\Terraform Script\Docker>docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
```

terraform apply

PORTS OUTPUT PROBLEMS TERMINAL DEBUG CONSOLE

```
C:\Users\Shravani\Desktop\Terraform Script\Docker>terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbf74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b62598aubuntu:latest]
```

Note: Objects have changed outside of Terraform

Terraform detected the following changes made outside of Terraform since the last "terraform apply" which may have affected this plan:

```
# docker_image.ubuntu has been deleted
- resource "docker_image" "ubuntu" {
    id      = "sha256:edbf74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b62598aubuntu:latest"
    image_id = "sha256:edbf74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b62598a" -> null
    name     = "ubuntu:latest"
    # (2 unchanged attributes hidden)
}
```

Unless you have made equivalent changes to your configuration, or ignored the relevant attributes using ignore_changes, the following plan may include actions to undo or respond to these changes.

Docker images, After Executing Apply step:

```
C:\Users\Shravani\Desktop\Terraform Script\Docker>docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    edbf74c41f8    3 weeks ago    78.1MB
```

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

```

C:\Users\Shravani\Desktop\Terraform Script\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]

Terraform used the selected providers to generate the following execution
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:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
  - image_id    = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
  - latest      = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
  - name        = "ubuntu:latest" -> null
  - repo_digest = "ubuntu@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee" -> 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:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.

```

Docker images After Executing Destroy step

Destroy complete! Resources: 1 destroyed.

```
C:\Users\Shravani\Desktop\Terraform Script\Docker>docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
```