

## Experiment No. 6

- Creating a docker image using terraform

```
PS C:\ProgramData\Microsoft\Windows\Start Menu> Docker --version
Docker version 27.1.1, build 6312585
PS C:\ProgramData\Microsoft\Windows\Start Menu> 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

- Create a folder named 'Terraform Scripts' in which we save our different types of scripts which will be further used in this experiment.



Terraform\_Scripts

8/22/2024 4:10 PM

File folder

- Create a new folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file using a text editor and write the following contents into it to create a Ubuntu Linux container.



docker.tf - Notepad

File Edit Format View Help

```
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_id  
    name = "foo"  
}
```

- Execute Terraform Init command to initialize the resources

```
D:\Terraform_Scripts\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.

D:\Terraform_Scripts\Docker>
```

- Execute Terraform plan to see the available resources

```
D:\Terraform_Scripts\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)
  + 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.

- 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”
- Check Docker images, Before and After Executing Apply step

```
D:\Terraform_Scripts\Docker>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mcr.microsoft.com/dotnet/framework/aspnet	4.8-windowsservercore-ltsc2022	0b1ef1176a57	6 weeks ago	5.43GB
mcr.microsoft.com/dotnet/framework/sdk	4.8-windowsservercore-ltsc2022	c3f8c2735565	6 weeks ago	9.04GB
mcr.microsoft.com/dotnet/framework/runtime	4.8-windowsservercore-ltsc2022	e69ea8a5ec1b	6 weeks ago	5.1GB
mcr.microsoft.com/windows/servercore	ltsc2022	e60f47e635b7	7 weeks ago	4.84GB
mcr.microsoft.com/windows/nanoserver	ltsc2022	f0ca29645006	7 weeks ago	292MB

```
D:\Terraform_Scripts\Docker>_
```

```
D:\Terraform_Scripts\Docker>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mcr.microsoft.com/dotnet/framework/aspnet	4.8-windowsservercore-ltsc2022	0b1ef1176a57	6 weeks ago	5.43GB
mcr.microsoft.com/dotnet/framework/sdk	4.8-windowsservercore-ltsc2022	c3f8c2735565	6 weeks ago	9.04GB
mcr.microsoft.com/dotnet/framework/runtime	4.8-windowsservercore-ltsc2022	e69ea8a5ec1b	6 weeks ago	5.1GB
mcr.microsoft.com/windows/servercore	ltsc2022	e60f47e635b7	7 weeks ago	4.84GB
mcr.microsoft.com/windows/nanoserver	ltsc2022	f0ca29645006	7 weeks ago	292MB
ubuntu	Latest	2dc39ba859dc	2 minutes ago	77.8MB

- Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.
- Check Docker images, After Executing Destroy step

```
D:\Terraform_Scripts\Docker>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mcr.microsoft.com/dotnet/framework/aspnet	4.8-windowsservercore-ltsc2022	0b1ef1176a57	6 weeks ago	5.43GB
mcr.microsoft.com/dotnet/framework/sdk	4.8-windowsservercore-ltsc2022	c3f8c2735565	6 weeks ago	9.04GB
mcr.microsoft.com/dotnet/framework/runtime	4.8-windowsservercore-ltsc2022	e69ea8a5ec1b	6 weeks ago	5.1GB
mcr.microsoft.com/windows/servercore	ltsc2022	e60f47e635b7	7 weeks ago	4.84GB
mcr.microsoft.com/windows/nanoserver	ltsc2022	f0ca29645006	7 weeks ago	292MB

```
D:\Terraform_Scripts\Docker>_
```