

Experiment 6

A. Creating docker image using terraform

Prerequisite:

Step 1: Download and Install Docker Desktop from <https://www.docker.com/>
Check the docker functionality

```
PS C:\Users\veyda> 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
```

```
Command Prompt

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

C:\Users\veyda>docker --version
Docker version 27.0.3, build 7d4bcd8
```

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

C:\Users\veyda>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
31e907dcc94a: Pull complete
Digest: sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview ubuntu

C:\Users\veyda>docker run -it ubuntu
root@0ea95fa5b779:/#
root@0ea95fa5b779:/#
```

Step 2: Create a folder called terra (do not use the word terraform or scripts as that may be a keyword.) and then create a folder called docker inside it.

Then create a file called docker.tf

```

docker.tf
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"
12 }
13
14 # Pull the image
15 resource "docker_image" "ubuntu" {
16   name = "ubuntu:latest"
17 }
18
19 # Create a container
20 resource "docker_container" "foo" {
21   image = docker_image.ubuntu.image_id
22   name = "foo"
23   command = ["sleep", "3600"]
24 }
25

```

Step 3: Go to powershell, switch directory to where the docker.tf script was saved, then run the terraform init command.

```

PS C:\Users\veyda\Desktop\terra> cd docker
PS C:\Users\veyda\Desktop\terra\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 view resources.

```

PS C:\Users\veyda\Desktop\terra\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)

```

Step 5: terraform show command for viewing all the state files

```

PS C:\Users\veyda\Desktop\terra\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    = "d4aafae11b3b"
  id         = "d4aafae11b3bf9d4a108b967b3c24943543a9b381fc75576cc06cd4fce85ee72"
  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
  publish_all_ports = false
  read_only    = false
  remove_volumes = true
  restart      = "no"
  rm          = false
  runtime      = "runc"
  security_opts = []
  shm_size     = 64

```

Step 6: 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 apply step:

```
PS C:\Users\veyda\Desktop\terra\docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
PS C:\Users\veyda\Desktop\terra\docker>
```

Docker images after apply step:

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

Step 7: terraform refresh command is helpful to sync your state file with the real-world infrastructure without applying any changes.

```
PS C:\Users\veyda\Desktop\terra\docker> terraform refresh
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=dfaafae11b3bf9d4a108b967b3c24943543a9b381fc75576cc06cd4fce85ee72]
```

Step 8: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
PS C:\Users\veyda\Desktop\terra\docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=dfaafae11b3bf9d4a108b967b3c24943543a9b381fc75576cc06cd4fce85ee72]

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",
  ]
}
```

Docker images After Executing Destroy step

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

