Steps to replicate our Terraform workflow that mounts our host directory

1) After setting up our terraform files that has our docker resource and nginx workflow, we firstly create a docker network

```
base ~/Documents/GitHub/Exam-FullStack/q2 (0.182s)
docker network create my_network

ae9a86f007ebad5f59018bfaf479f2998961cdb68ba1d2a2f291916aab96ceca
```

2) Then we terraform init

```
base ~/Documents/GitHub/Exam-FullStack/q3 (2.491s)
terraform init

Initializing the backend...

Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 2.13.0"...
- Installing kreuzwerker/docker v2.13.0...
- Installed kreuzwerker/docker v2.13.0 (self-signed, key ID 24E54F214569A8A5)

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.
```

3) Then terraform plan

```
base ~/Documents/GitHub/Exam-FullStack/q3 (0.351s)
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.nginx will be created
  + resource "docker_container" "nginx" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = [
          + "MY_ENV_VAR=my_value",
        ]
      + 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       = "json-file"
      + logs             = false
      + must_run         = true
      + name             = "nginx_container"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + tty              = false

      + networks_advanced {
          + aliases = [
              + "nginx_server",
            ]
          + name    = "my_network"
```

```
  + networks_advanced {
      + aliases = [
          + "nginx_server",
        ]
      + name    = "my_network"
    }

  + ports {
      + external = 8080
      + internal = 80
      + ip       = "0.0.0.0"
      + protocol = "tcp"
    }

  + volumes {
      + container_path = "/usr/share/nginx/html"
      + host_path      = "/Users/Munachi/Documents/GitHub/Exam-FullStack/"
      + read_only      = false
    }
}

docker_image.nginx will be created
resource "docker_image" "nginx" {
  + id          = (known after apply)
  + latest      = (known after apply)
  + name        = "nginx:latest"
  + output      = (known after apply)
  + repo_digest = (known after apply)
}
```

4) We run terraform apply

```
base ~/Documents/GitHub/Exam-FullStack/q3 (12.396s)
terraform apply

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.nginx will be created
  + resource "docker_container" "nginx" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = [
          + "MY_ENV_VAR=my_value",
        ]
      + 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       = "json-file"
      + logs             = false
      + must_run         = true
      + name             = "nginx_container"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + tty              = false
```

5) We can verify our docker container is up by running docker ps command and executing a shell command as follows

```
base ~/Documents/GitHub/Exam-FullStack/q3 (0.102s)
docker ps
CONTAINER ID   IMAGE                                 COMMAND                  CREATED              STATU
S          PORTS
                                          NAMES
af1ee1a08397   070027a3cbe0                          "/docker-entrypoint.…"   About a minute ago   Up 59
 seconds   0.0.0.0:8080->80/tcp
                                          nginx_container
f9add35517bd   gcr.io/k8s-minikube/kicbase:v0.0.42   "/usr/local/bin/entr…"   48 minutes ago       Up 48
 minutes   127.0.0.1:51069->22/tcp, 127.0.0.1:51070->2376/tcp, 127.0.0.1:51072->5000/tcp, 127.0.0.1:5107
3->8443/tcp, 127.0.0.1:51071->32443/tcp   minikube
83bfc994b555   my-nginx-image                        "/docker-entrypoint.…"   59 minutes ago       Up 59
 minutes   0.0.0.0:8000->80/tcp
                                          my-nginx-container

base ~/Documents/GitHub/Exam-FullStack/q3 (0.186s)
docker exec nginx_container env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=af1ee1a08397
MY_ENV_VAR=my_value
NGINX_VERSION=1.25.4
NJS_VERSION=0.8.3
PKG_RELEASE=1~bookworm
HOME=/root

base ~/Documents/GitHub/Exam-FullStack/q3
```

6) We can also verify our directory was correctly mounted on path

```
base ~/Documents/GitHub/Exam-FullStack/q3 (0.038s)
mkdir -p ~/Documents/Github/Exam-FullStack/q3
echo "This is a test file." > ~/Documents/Github/Exam-FullStack/q3/testfile.txt

base ~/Documents/GitHub/Exam-FullStack/q3 (0.108s)
docker exec nginx_container
"docker exec" requires at least 2 arguments.
See 'docker exec --help'.

Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container

base ~/Documents/GitHub/Exam-FullStack/q3 (35.523s)
docker exec -it nginx_container /bin/sh

# ls
bin   dev                   docker-entrypoint.sh  home  media  opt   root  sbin  sys  usr
boot  docker-entrypoint.d  etc                   lib   mnt    proc  run   srv   tmp  var
# cd /usr/share/nginx/html
ls
# q1  q2        q3
# cd q3
# ls
main.tf  terraform.tfstate  testfile.txt
#
```

Notice how the addition of testfile, shows the same testfile in our docker working directory.