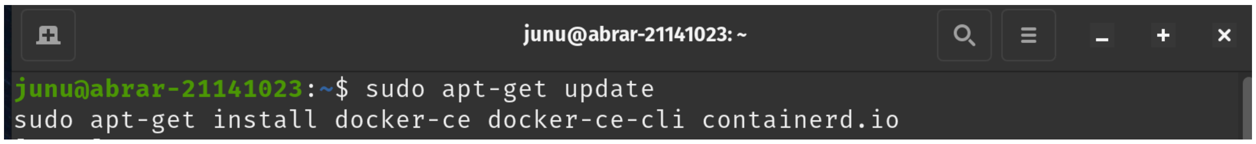
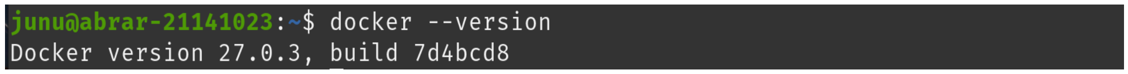
Getting Started with Docker

1. First, we will update and install the docker container using this command:



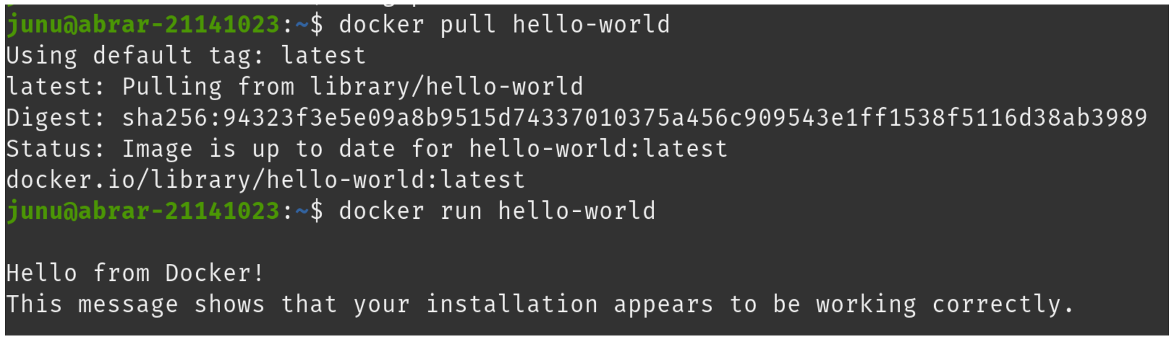
```
junu@abrar-21141023:~$ sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

   - Then using this command, we will check if docker has been successfully installed



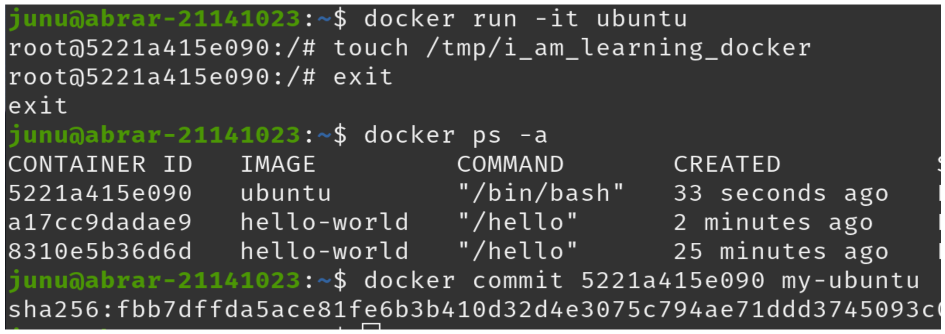
```
junu@abrar-21141023:~$ docker --version
Docker version 27.0.3, build 7d4bcd8
```

2. Let's go through some basic Docker commands:
   - Pull an image
   - Run the container using the image we pulled



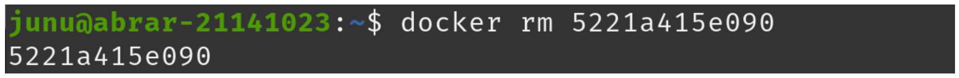
```
junu@abrar-21141023:~$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:94323f3e5e09a8b9515d74337010375a456c909543e1ff1538f5116d38ab3989
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest
junu@abrar-21141023:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

   - I will create a file named “i_am_learning_docker” in my tmp folder, and commit these changes to an Ubuntu container I just created



```
junu@abrar-21141023:~$ docker run -it ubuntu
root@5221a415e090:/# touch /tmp/i_am_learning_docker
root@5221a415e090:/# exit
exit
junu@abrar-21141023:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED
5221a415e090   ubuntu        "/bin/bash"   33 seconds ago
a17cc9dadae9   hello-world   "/hello"      2 minutes ago
8310e5b36d6d   hello-world   "/hello"      25 minutes ago
junu@abrar-21141023:~$ docker commit 5221a415e090 my-ubuntu
sha256:fbb7dffda5ace81fe6b3b410d32d4e3075c794ae71ddd3745093c
```

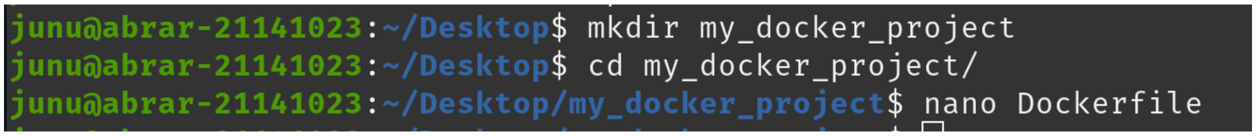
   - To remove the container



```
junu@abrar-21141023:~$ docker rm 5221a415e090
5221a415e090
```

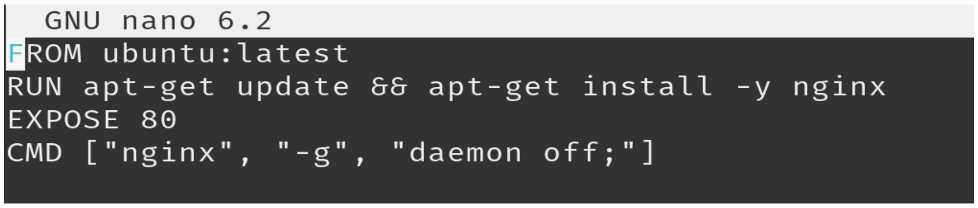
Creating a Docker Image using Dockerfile

1. Creating a new directory for my Docker project and then create a file named Dockerfile(no extension):



```
junu@abrar-21141023:~/Desktop$ mkdir my_docker_project
junu@abrar-21141023:~/Desktop$ cd my_docker_project/
junu@abrar-21141023:~/Desktop/my_docker_project$ nano Dockerfile
```

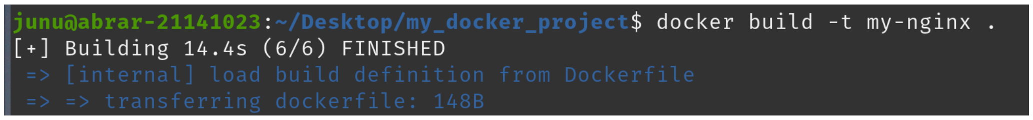
- Add the following content to the Dockerfile



```
  GNU nano 6.2
FROM ubuntu:latest
RUN apt-get update && apt-get install -y nginx
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Save and exit the file(Ctrl+X, then Y)

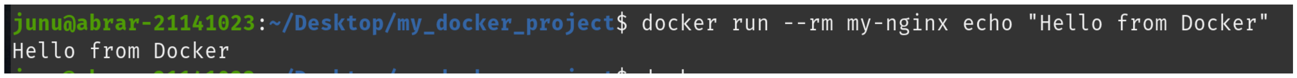
- Build the Docker image



```
junu@abrar-21141023:~/Desktop/my_docker_project$ docker build -t my-nginx .
[+] Building 14.4s (6/6) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 148B
```

  - *FROM ubuntu:latest: This sets the base image to the latest Ubuntu.*
  - *RUN apt-get update && apt-get install -y nginx: This updates the package lists and installs Nginx.*
  - *EXPOSE 80: This informs Docker that the container will listen on port 80.*
  - *CMD ["nginx", "-g", "daemon off;"]: This is the command that will run when the container starts*

Running a container as a single task:

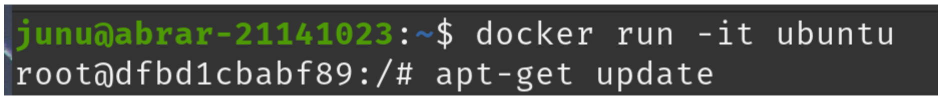


```
junu@abrar-21141023:~/Desktop/my_docker_project$ docker run --rm my-nginx echo "Hello from Docker"
Hello from Docker
```

  - *--rm: This flag tells Docker to remove the container after it exits.*
  - *echo "Hello from Docker": This is the single task we're running instead of the default CMD.*

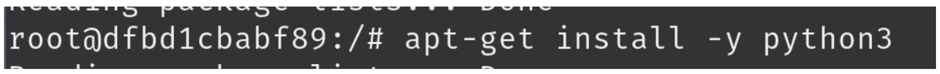
Running a container in interactive mode and installing packages:

- First I'll start an interactive mode
- Update package lists



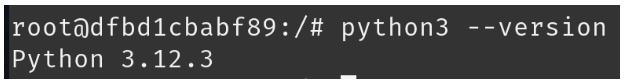
```
junu@abrar-21141023:~$ docker run -it ubuntu
root@dfbd1cbabf89:/# apt-get update
```

- Install Packages(Python-3)



```
root@dfbd1cbabf89:/# apt-get install -y python3
```

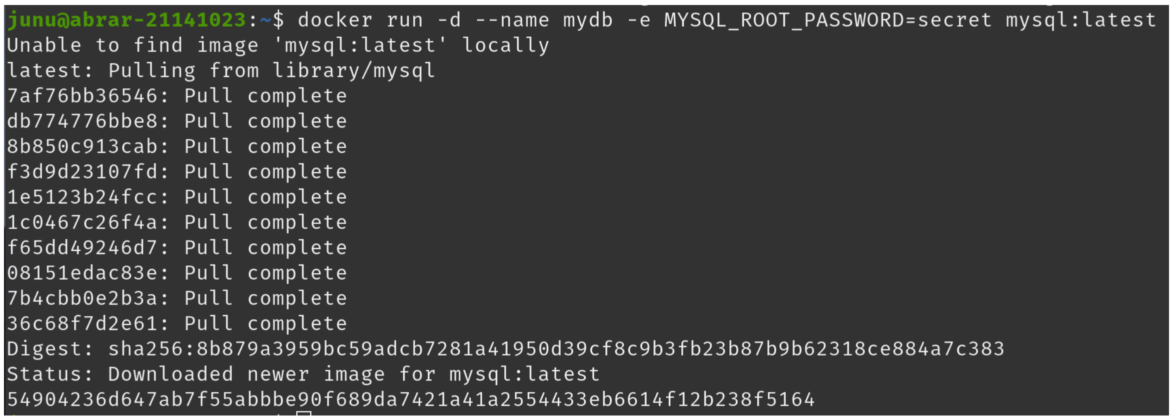
- Check the Python Version



```
root@dfbd1cbabf89:/# python3 --version
Python 3.12.3
```

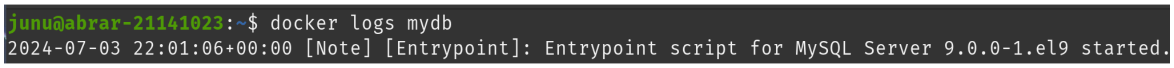
Run a database container, show logs, and access interactively

1. Start a MySQL container in the background



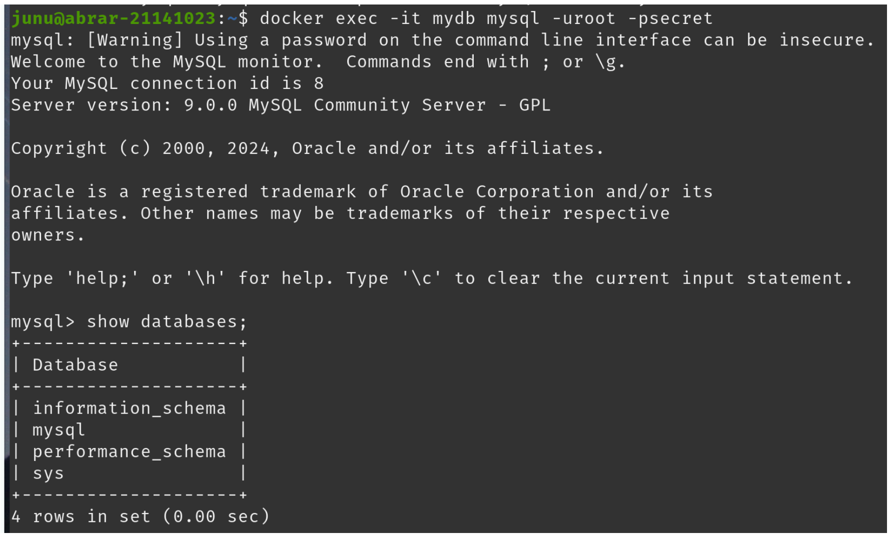
```
junu@abrar-21141023:~$ docker run -d --name mydb -e MYSQL_ROOT_PASSWORD=secret mysql:latest
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
7af76bb36546: Pull complete
db774776bbe8: Pull complete
8b850c913cab: Pull complete
f3d9d23107fd: Pull complete
1e5123b24fcc: Pull complete
1c0467c26f4a: Pull complete
f65dd49246d7: Pull complete
08151edac83e: Pull complete
7b4cbb0e2b3a: Pull complete
36c68f7d2e61: Pull complete
Digest: sha256:8b879a3959bc59adcb7281a41950d39cf8c9b3fb23b87b9b62318ce884a7c383
Status: Downloaded newer image for mysql:latest
54904236d647ab7f55abbbe90f689da7421a41a2554433eb6614f12b238f5164
```

2. Viewing the container logs



```
junu@abrar-21141023:~$ docker logs mydb
2024-07-03 22:01:06+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 9.0.0-1.el9 started.
```

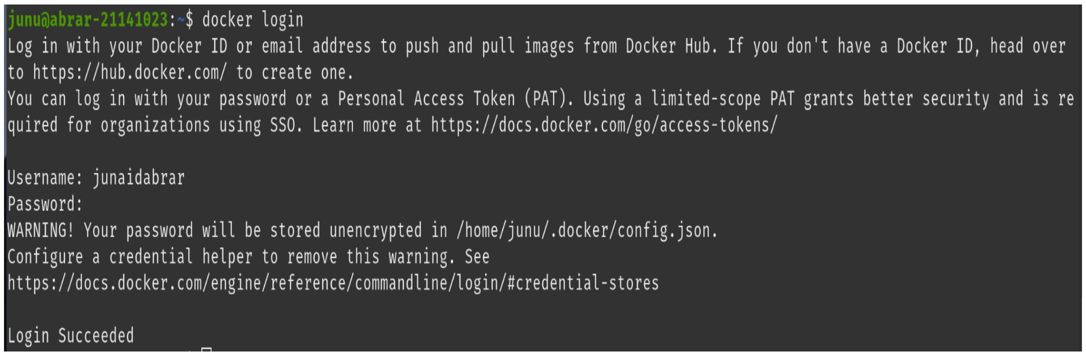
3. Accessing the MySQL shell in the container and running a SQL command



```
junu@abrar-21141023:~$ docker exec -it mydb mysql -uroot -psecret
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 9.0.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

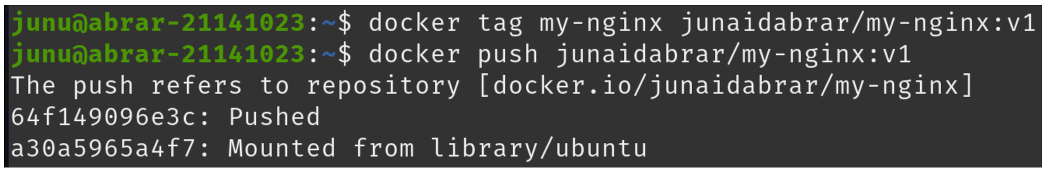
Pushing my image to Docker Hub

1. Log in to Docker Hub
    - Creating an account in **Docker Hub**
    - Going to settings>security to retrieve my personal access token(PAT)
    - Login to docker



```
junu@abrar-21141023:~$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over
to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is re
quired for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: junaidabrar
Password:
WARNING! Your password will be stored unencrypted in /home/junu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
```

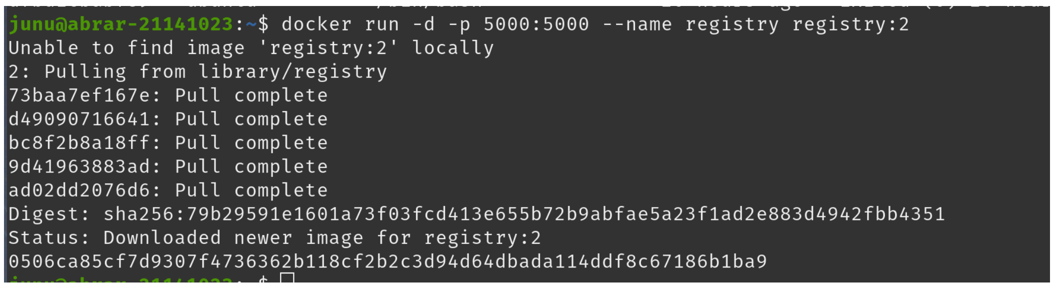
2. Tag and push the image to the **Docker Hub**



```
junu@abrar-21141023:~$ docker tag my-nginx junaidabrar/my-nginx:v1
junu@abrar-21141023:~$ docker push junaidabrar/my-nginx:v1
The push refers to repository [docker.io/junaidabrar/my-nginx]
64f149096e3c: Pushed
a30a5965a4f7: Mounted from library/ubuntu
```

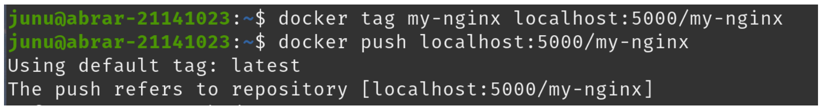
Creating a private registry

1. Running a local registry using this command, named "*registry*"



```
junu@abrar-21141023:~$ docker run -d -p 5000:5000 --name registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
73baa7ef167e: Pull complete
d49090716641: Pull complete
bc8f2b8a18ff: Pull complete
9d41963883ad: Pull complete
ad02dd2076d6: Pull complete
Digest: sha256:79b29591e1601a73f03fcd413e655b72b9abfae5a23f1ad2e883d4942fbb4351
Status: Downloaded newer image for registry:2
0506ca85cf7d9307f4736362b118cf2b2c3d94d64dbada114ddf8c67186b1ba9
```

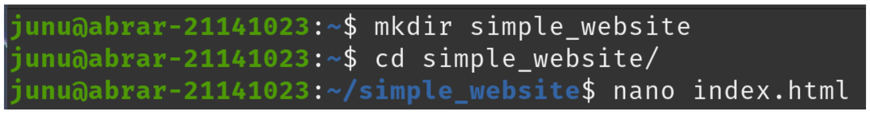
2. I am tagging my image for the local registry and then push it.



```
junu@abrar-21141023:~$ docker tag my-nginx localhost:5000/my-nginx
junu@abrar-21141023:~$ docker push localhost:5000/my-nginx
Using default tag: latest
The push refers to repository [localhost:5000/my-nginx]
```

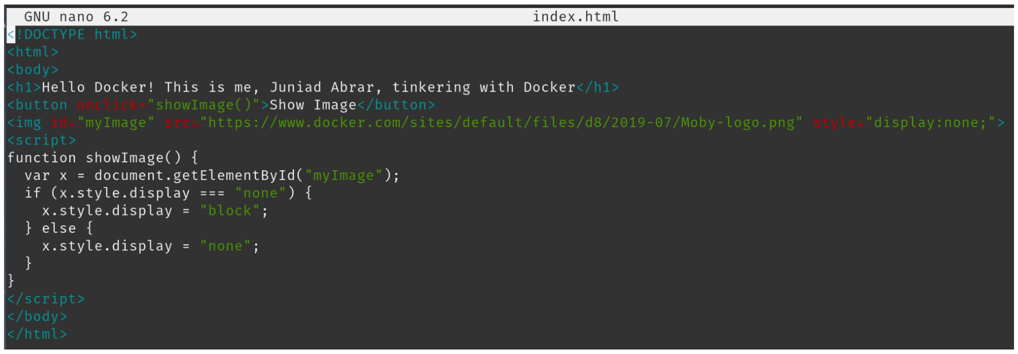
Creating a simple website in a Docker container

1. Make a directory, change to that directory, and then use *nano* to create an HTML file



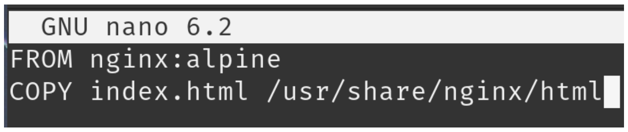
```
junu@abrar-21141023:~$ mkdir simple_website
junu@abrar-21141023:~$ cd simple_website/
junu@abrar-21141023:~/simple_website$ nano index.html
```

- Paste this in the index. html file



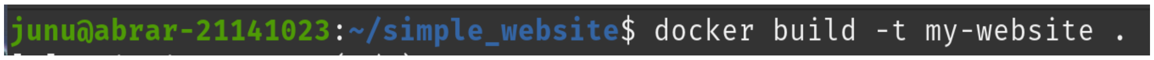
```
  GNU nano 6.2                                          index.html
<!DOCTYPE html>
<html>
<body>
<h1>Hello Docker! This is me, Juniad Abrar, tinkering with Docker</h1>
<button onclick="showImage()">Show Image</button>
<img id="myImage" src="https://www.docker.com/sites/default/files/d8/2019-07/Moby-logo.png" style="display:none;">
<script>
function showImage() {
  var x = document.getElementById("myImage");
  if (x.style.display === "none") {
    x.style.display = "block";
  } else {
    x.style.display = "none";
  }
}
</script>
</body>
</html>
```

- Create a new Dockerfile using *nano* and add this content to the *Dockerfile*



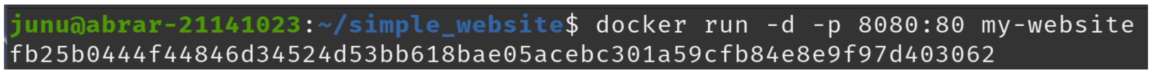
```
  GNU nano 6.2
FROM nginx:alpine
COPY index.html /usr/share/nginx/html
```

- Now let’s build the image and run the container



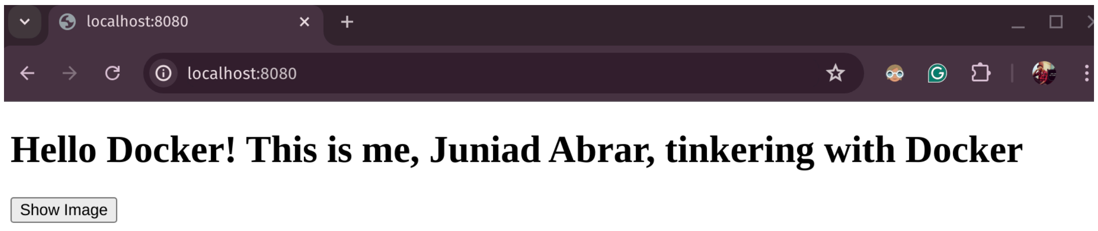
```
junu@abrar-21141023:~/simple_website$ docker build -t my-website .
```

- Run the container



```
junu@abrar-21141023:~/simple_website$ docker run -d -p 8080:80 my-website
fb25b0444f44846d34524d53bb618bae05acebc301a59cfb84e8e9f97d403062
```

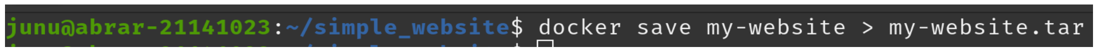
2. Now I can access the website at **http://localhost:8080**



Hello Docker! This is me, Juniad Abrar, tinkering with Docker

Migrating the container to another machine

1. On the Source the Machine
    - Saving my-website image to a file named my-website.tar



```
junu@abrar-21141023:~/simple_website$ docker save my-website > my-website.tar
```

    Now after transferring '*my-website.tar*' file, I can:
    - Load the image
      Command: *docker load < my-website.tar*
    - Run the container
      Command: *docker run -d -p 8080:80 my-website*
2. Now I should be able to access the website at `http://destination-ip:8080` in my destination machine's web browser.