

# Nnadiyekwe, Chiderah David

---

*Project- Containerizing webserver using Docker*

---

● Create a Docker hub public repository

● Containerize our Apache webserver

● Upload image to your docker hub

● Have a friend from your cluster to pull your image and build it.

● Share screenshot of app in browser, and image uploaded

● Upload it in a repo on your github



- **Create a Dockerhub Public Repository**

← ↻ <https://hub.docker.com/repository/create?namespace=chidave01> ⌵ ☆ ⚙ | 📄 ☆ 🗂 🔄 ...

🔍 Upgrade

[Repositories](#) / [Create](#)  
Using 0 of 1 private repositories. [Get more](#)

### Create repository

Namespace  Repository Name \*

Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

### Visibility

Using 0 of 1 private repositories. [Get more](#)

**Public** ☒ Appears in Docker Hub search results

**Private** ☐ Only visible to you

[Cancel](#) [Create](#)

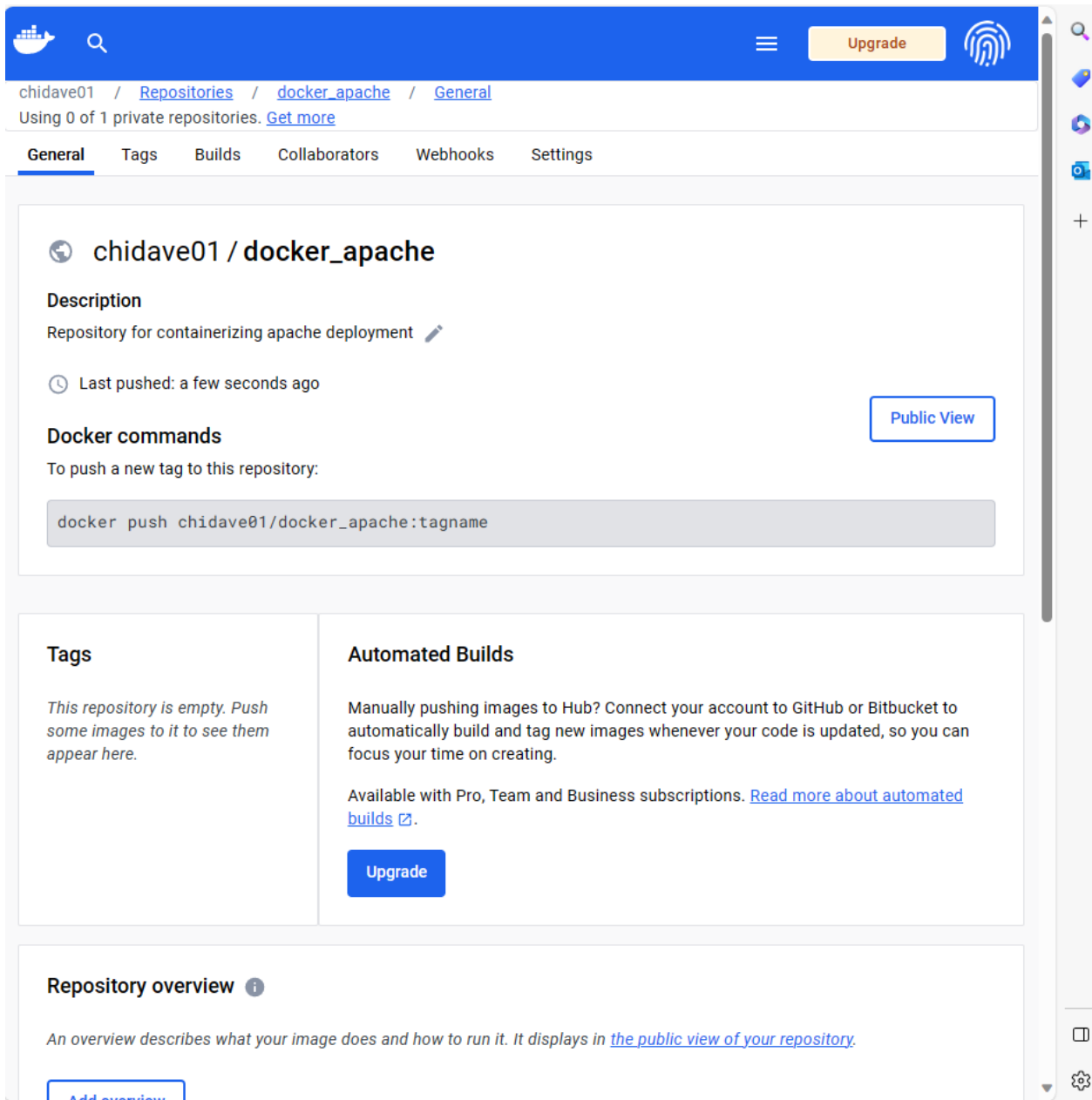
### Pushing images

You can push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to replace `tagname` with your desired image repository tag.

First, I signed up on dockerhub and created a repository titled “docker\_apache”. Note that only lowercase is accepted in creating the docker hub repository name.

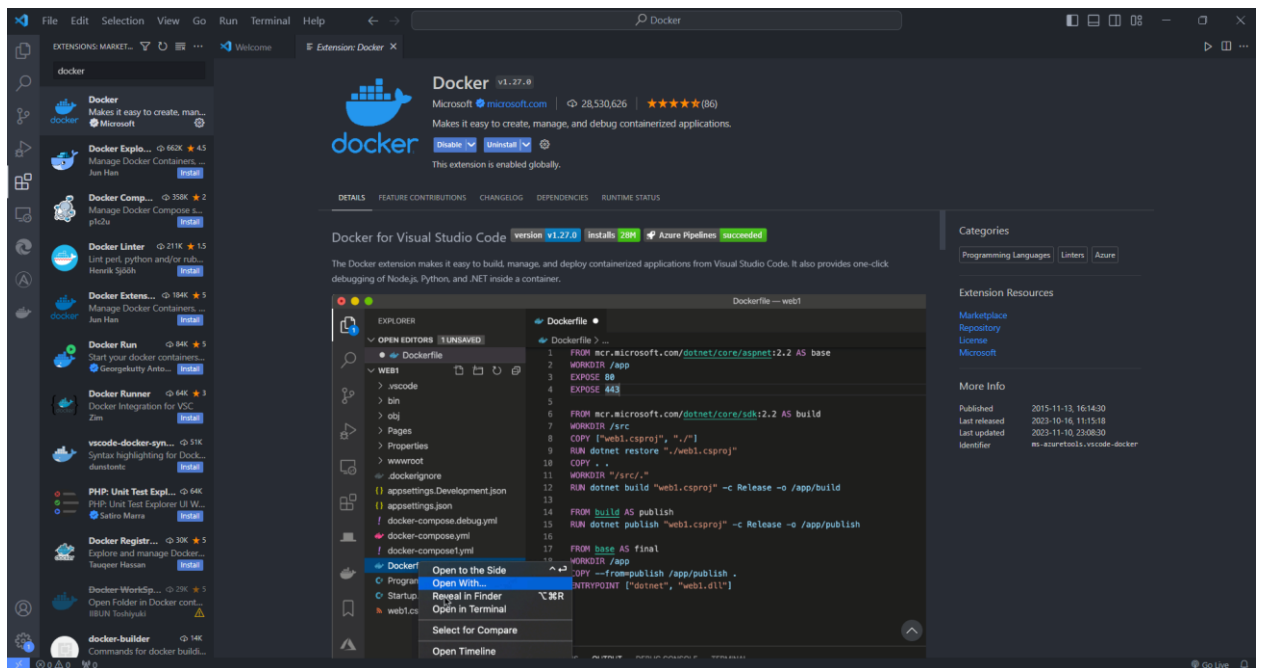


- **Containerize the apache webserver**

In order to achieve this, I need a dockerfile to create a docker image. It will save you a lot of time to download the docker extension on Vscod.

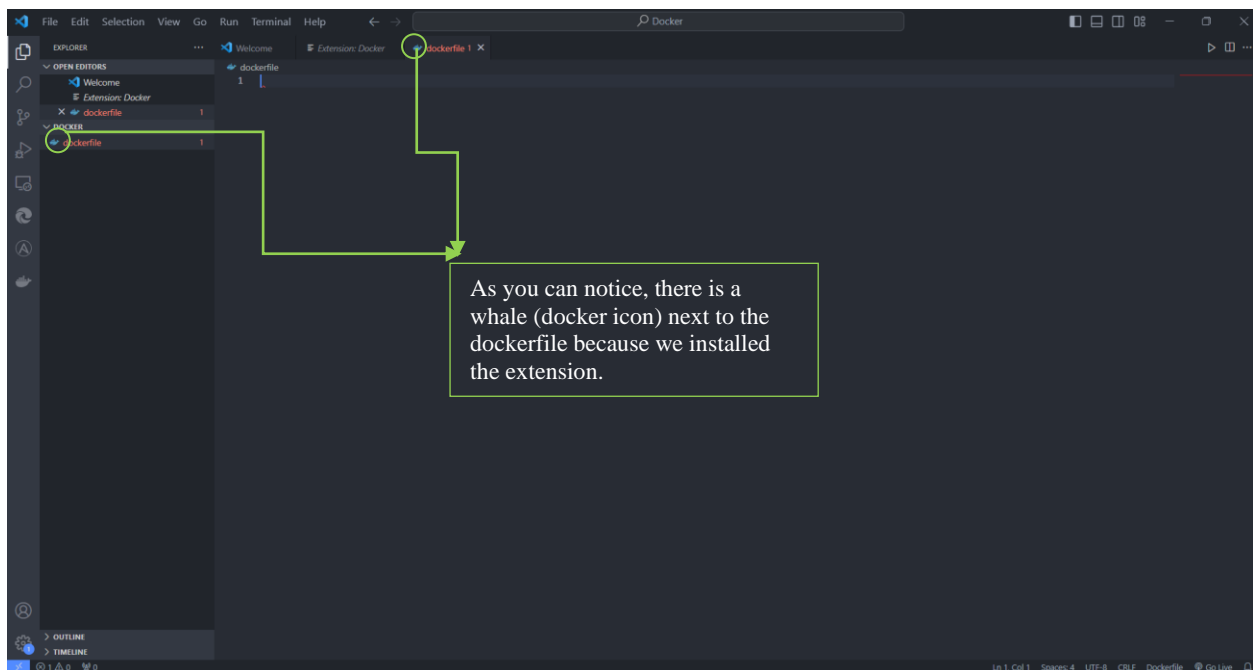
A docker file always start with a base image (something similar to an operating system for the container. You can choose centos, linux, ubuntu, etc...)

Always reference the documentation as a guide in order not to make any mistakes.

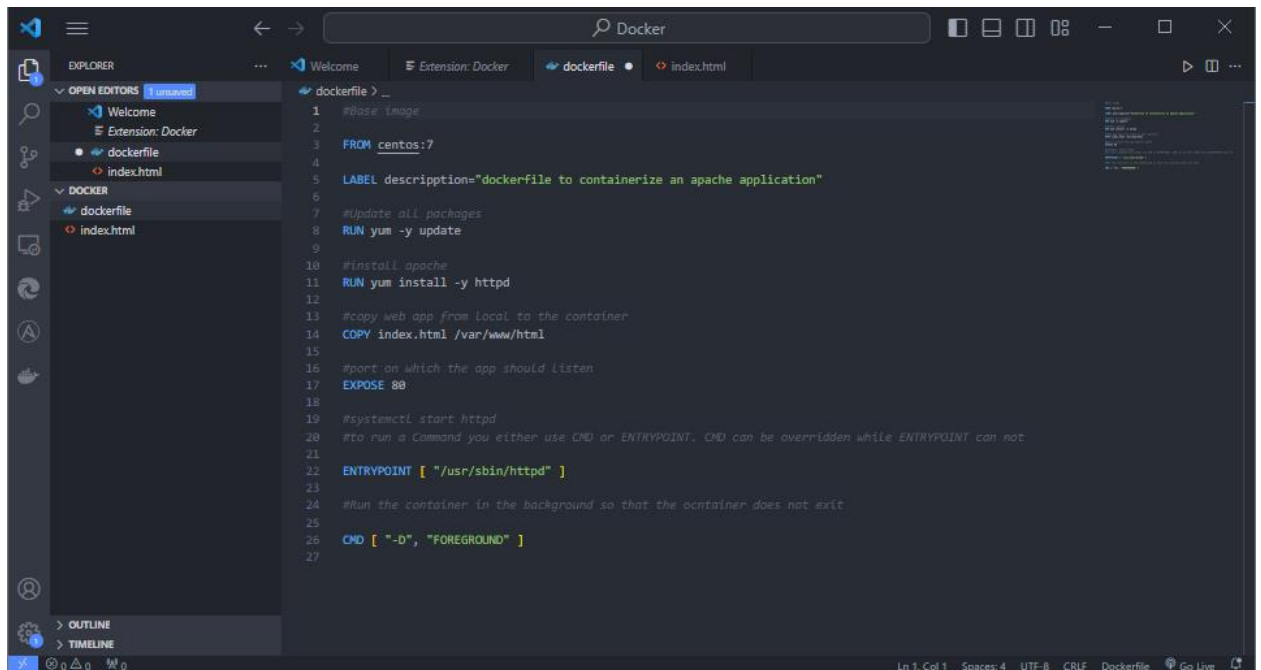


I downloaded the Docker Extension to my VsCode.

Next, I created a dockerfile...



As you can notice, there is a whale (docker icon) next to the dockerfile because we installed the extension.

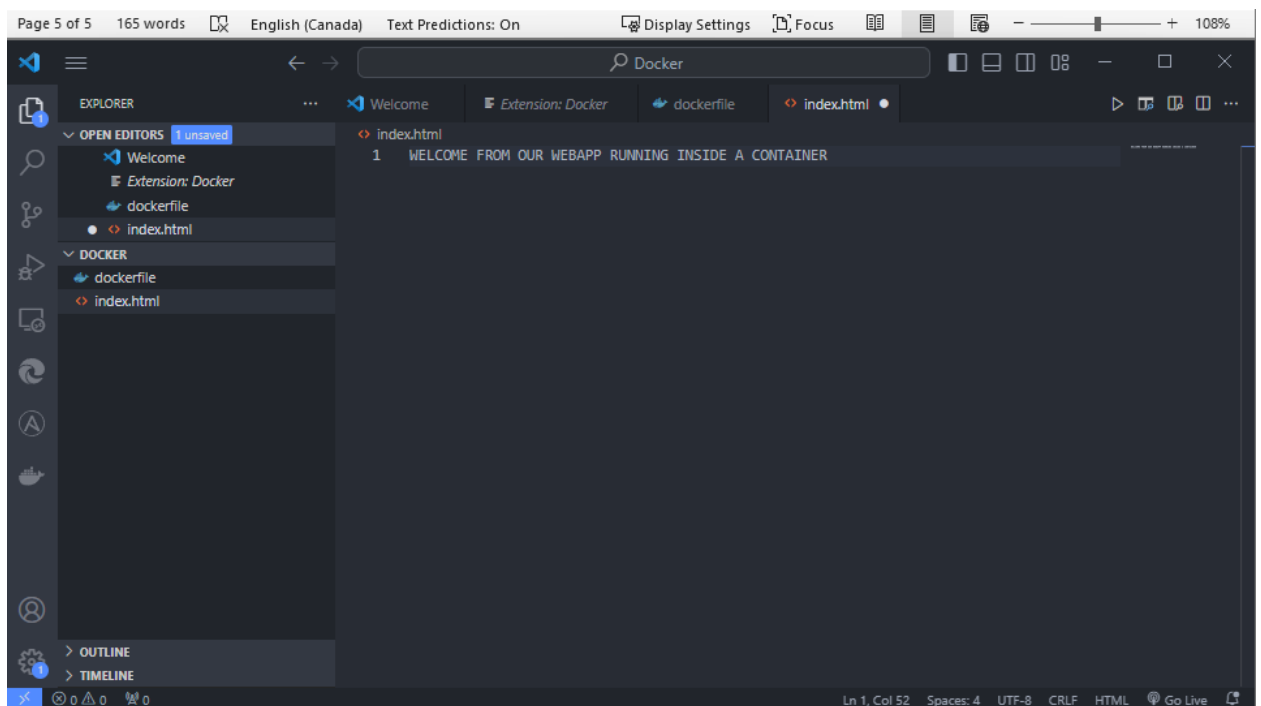


The screenshot shows the Visual Studio Code interface with the Docker extension. The Explorer sidebar on the left shows the file structure with 'dockerfile' and 'index.html' under the 'DOCKER' folder. The main editor window displays the 'dockerfile' file with the following content:

```
1 #Base image
2
3 FROM centos:7
4
5 LABEL description="dockerfile to containerize an apache application"
6
7 #Update all packages
8 RUN yum -y update
9
10 #install apache
11 RUN yum install -y httpd
12
13 #copy web app from local to the container
14 COPY index.html /var/www/html
15
16 #port on which the app should listen
17 EXPOSE 80
18
19 #systemctl start httpd
20 #to run a Command you either use CMD or ENTRYPOINT. CMD can be overridden while ENTRYPOINT can not
21
22 ENTRYPOINT [ "/usr/sbin/httpd" ]
23
24 #Run the container in the background so that the container does not exit
25
26 CMD [ "-D", "FOREGROUND" ]
27
```

The status bar at the bottom indicates 'Ln 1, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Dockerfile', and 'Go Live'.

Since we referenced the index.html file in the dockerfile, we need to create an index.html file in the same folder as the dockerfile.



The screenshot shows the Visual Studio Code interface with the Docker extension. The Explorer sidebar on the left shows the file structure with 'dockerfile' and 'index.html' under the 'DOCKER' folder. The main editor window displays the 'index.html' file with the following content:

```
1 WELCOME FROM OUR WEBAPP RUNNING INSIDE A CONTAINER
```

The status bar at the bottom indicates 'Ln 1, Col 52', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and 'Go Live'.

- **Building and uploading image to dockerhub repository**

To achieve this, we need to use an ec2, install docker in it, run the centos image (base image) and build the apache image on it.

- **Installing docker on linux instance:**

In order to achieve this, I referenced the Amazon documentation to install docker docker. Note that Amazon Linux instance is Fedora operating system based. Hence, we will use the documentation for fedora.

Documentation reference: <https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/install-docker.html>

```
Session ID: admin-008b72aee75030f12 Instance ID: i-00455a6458fff6f6d Terminate

sh-5.2$ sudo yum update -y
Last metadata expiration check: 0:04:09 ago on Sat Nov 11 17:15:20 2023.
Dependencies resolved.
Nothing to do.
Complete!
sh-5.2$ sudo amazon-linux-extras install docker
sudo: amazon-linux-extras: command not found
sh-5.2$ sudo yum install -y docker
Last metadata expiration check: 0:04:35 ago on Sat Nov 11 17:15:20 2023.
Dependencies resolved.
=====
Package                Architecture Version                                Repository                Size
=====
Installing:
docker                 x86_64      24.0.5-1.amzn2023.0.2                amazonlinux                42 M
Installing dependencies:
containerd             x86_64      1.7.2-1.amzn2023.0.4                amazonlinux                34 M
iptables-lib           x86_64      1.8.8-3.amzn2023.0.2                amazonlinux                401 k
iptables-nft           x86_64      1.8.8-3.amzn2023.0.2                amazonlinux                183 k
libcgroup              x86_64      3.0-1.amzn2023.0.1                  amazonlinux                75 k
libnetfilter_conntrack x86_64      1.0.8-2.amzn2023.0.2                amazonlinux                58 k
libnftnl                x86_64      1.0.1-19.amzn2023.0.2               amazonlinux                30 k
libnftnl                x86_64      1.2.2-2.amzn2023.0.2                amazonlinux                84 k
pigz                   x86_64      2.5-1.amzn2023.0.3                  amazonlinux                83 k
runc                   x86_64      1.1.7-1.amzn2023.0.3                amazonlinux                3.0 M

Transaction Summary
=====
Install 10 Packages

Total download size: 80 M
Installed size: 306 M
Downloading Packages:
(1/10): iptables-lib-1.8.8-3.amzn2023.0.2.x86_64.rpm           5.2 MB/s | 401 kB      00:00
(2/10): runc-1.1.7-1.amzn2023.0.3.x86_64.rpm                  28 MB/s | 3.0 MB      00:00
(3/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm                     2.3 MB/s | 83 kB      00:00
(4/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm              720 kB/s | 84 kB      00:00
(5/10): libnftnl-1.0.1-19.amzn2023.0.2.x86_64.rpm             1.9 MB/s | 30 kB      00:00
(6/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm               1.4 MB/s | 75 kB      00:00
(7/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm 3.6 MB/s | 58 kB      00:00
(8/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm          7.9 MB/s | 183 kB     00:00
(9/10): containerd-1.7.2-1.amzn2023.0.4.x86_64.rpm            46 MB/s | 34 MB      00:00
(10/10): docker-24.0.5-1.amzn2023.0.2.x86_64.rpm              42 MB/s | 42 MB      00:00
-----
Total                                           68 MB/s | 80 MB      00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing :
1/1
```

We need to start docker after installing docker in the instance:

```

(7/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm 3.6 MB/s | 58 kB 00:00
(8/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm 7.9 MB/s | 183 kB 00:00
(9/10): containerd-1.7.2-1.amzn2023.0.4.x86_64.rpm 46 MB/s | 34 MB 00:00
(10/10): docker-24.0.5-1.amzn2023.0.2.x86_64.rpm 42 MB/s | 42 MB 00:00
-----
Total 68 MB/s | 80 MB 00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : 1/1
Installing : runc-1.1.7-1.amzn2023.0.3.x86_64 1/10
Installing : containerd-1.7.2-1.amzn2023.0.4.x86_64 2/10
Running scriptlet: containerd-1.7.2-1.amzn2023.0.4.x86_64 2/10
Installing : libcgroup-3.0-1.amzn2023.0.1.x86_64 3/10
Installing : libnftnl-1.0.1-19.amzn2023.0.2.x86_64 4/10
Installing : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 5/10
Installing : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 6/10
Installing : pigz-2.5-1.amzn2023.0.3.x86_64 7/10
Installing : libnftnl-1.2.2-2.amzn2023.0.2.x86_64 8/10
Installing : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64 9/10
Running scriptlet: iptables-nft-1.8.8-3.amzn2023.0.2.x86_64 9/10
Running scriptlet: docker-24.0.5-1.amzn2023.0.2.x86_64 10/10
Installing : docker-24.0.5-1.amzn2023.0.2.x86_64 10/10
Running scriptlet: docker-24.0.5-1.amzn2023.0.2.x86_64 10/10
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

Verifying : libnftnl-1.2.2-2.amzn2023.0.2.x86_64 1/10
Verifying : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 2/10
Verifying : runc-1.1.7-1.amzn2023.0.3.x86_64 3/10
Verifying : pigz-2.5-1.amzn2023.0.3.x86_64 4/10
Verifying : libnftnl-1.0.1-19.amzn2023.0.2.x86_64 5/10
Verifying : libcgroup-3.0-1.amzn2023.0.1.x86_64 6/10
Verifying : docker-24.0.5-1.amzn2023.0.2.x86_64 7/10
Verifying : containerd-1.7.2-1.amzn2023.0.4.x86_64 8/10
Verifying : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 9/10
Verifying : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64 10/10

Installed:
containerd-1.7.2-1.amzn2023.0.4.x86_64 docker-24.0.5-1.amzn2023.0.2.x86_64
iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgroup-3.0-1.amzn2023.0.1.x86_64 libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
libnftnl-1.0.1-19.amzn2023.0.2.x86_64 libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64 runc-1.1.7-1.amzn2023.0.3.x86_64

Complete!
sh-5.2$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
sh-5.2$

```

We also have to add the ec2-user in the permission group. After we do this, we need to exit the instance and re-login to make the changes to work.

```

sh-5.2$ sudo usermod -a -G docker ec2-user
sh-5.2$

```

#### - Saving the dockerfile and the index file in the instance:

To generate an image using the dockerfile, we need to have them saved locally in the instance.

We use the command “vim dockerfile” to create a dockerfile and then paste the dockerfile script and save. We have to do the same for the index file.

Session ID: admin-0498d7ff26744b469

Instance ID: i-00455a6458fff6f6d

Terminate

```
#Base image
FROM centos:7

LABEL description="dockerfile to containerize an apache application"

#Update all packages
RUN yum -y update

#install apache
RUN yum install -y httpd

#copy web app from local to the container
COPY index.html /var/www/html

#port on which the app should listen
EXPOSE 80

#systemctl start httpd
#to run a Command you either use CMD or ENTRYPOINT. CMD can be overridden while ENTRYPOINT can not

ENTRYPOINT [ "/usr/sbin/httpd" ]

#Run the container in the background so that the ocntainer does not exit

CMD [ "-D", "FOREGROUND" ]
~
~
~
```

We also need to create and save the index file in the instance;

```
sh-5.2$ sudo su - ec2-user
Last login: Sat Nov 11 17:26:46 UTC 2023 on pts/0
[ec2-user@ip-10-0-3-121 ~]$ ls -l
total 0
[ec2-user@ip-10-0-3-121 ~]$ vim dockerfile
[ec2-user@ip-10-0-3-121 ~]$ vim index.html
[ec2-user@ip-10-0-3-121 ~]$
```

Session ID: admin-069df29cf92fa5e95

Instance ID: i-00455a6458fff6f6d

Terminate

```
[ec2-user@ip-10-0-3-121 bin]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-10-0-3-121 bin]$
```



- Next, we need to build the image using the dockerfile and the index file.

We use the command: “docker build -t apacheimage:v1 .” where the “apacheimage” is the name of the docker image we want to build while “v1” is a tag. The “.” Signifies the location where the dockerfile is saved in the local computer. Alternatively if the dockerfile was located in another location, you need to state the path of the directory where the docker file is saved.

```
[ec2-user@ip-10-0-9-10 ~]$ docker build --ulimit nofile=1024 -t apacheimage:v1 .
[+] Building 39.2s (9/9) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 656B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/centos:7
=> CACHED [1/4] FROM docker.io/library/centos:7@sha256:be65f488b7764ad3638f236b7b515b3678369a5124c47b8d32916d6487418ea4
=> [internal] load build context
=> => transferring context: 86B
=> [2/4] RUN yum -y update
=> [3/4] RUN yum install -y httpd
=> [4/4] COPY index.html /var/www/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:8c3323a18bb20931a1d6c867e91df893122df3de337fad3fb7113828f0ff0cc9
=> => naming to docker.io/library/apacheimage:v1
[ec2-user@ip-10-0-9-10 ~]$
```

**Note:** My build process was taking too long so I used an additional command to execute the build process faster. The command is; “docker build –ulimit nofile=1024 -t apacheimage:v1 .” and everything ran smoothly and faster.

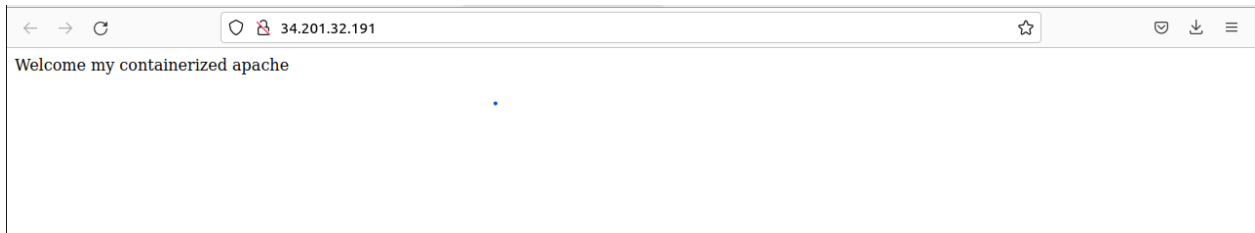
To verify that the image was built, I used the “docker images” command

```
[ec2-user@ip-10-0-9-10 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
apacheimage         v1                 8c3323a18bb2       5 minutes ago      802MB
ubuntu              latest             e4c58958181a       7 weeks ago        77.8MB
hello-world         latest             9c7a54a9a43c       6 months ago       13.3kB
ubuntu              16.04             b6f507652425       2 years ago        135MB
[ec2-user@ip-10-0-9-10 ~]$
```

To run the image, I used the command “docker run -t -d -p 80:80 apacheimage:v1” . The “-d” will allow the apache run in the background

```
[ec2-user@ip-10-0-9-10 ~]$ docker run -t -d -p 80:80 apacheimage:v1
54c030f28a8054a31ecb6b9b9f1231301396c26aa698b91bc862c4bec2c1c2f9
[ec2-user@ip-10-0-9-10 ~]$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                               NAMES
54c030f28a80   apacheimage:v1       "/usr/sbin/httpd -D ..." 6 seconds ago  Up 5 seconds  0.0.0.0:80->80/tcp, :::80->80/tcp  zealous_newton
[ec2-user@ip-10-0-9-10 ~]$
```

To verify that the image is running, I’ll try to access the public IP of the instance used in creating the image.



It works! :)

- **Uploading image to dockerhub repository**

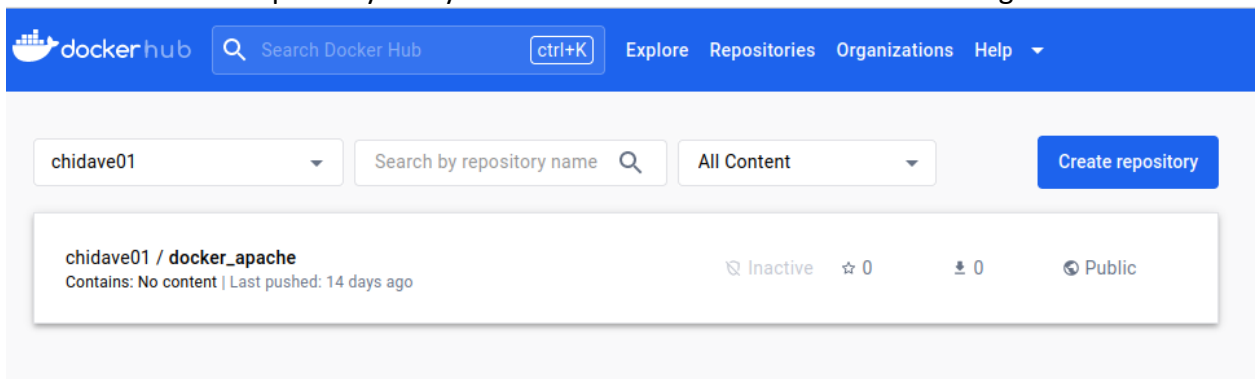
```
[ec2-user@ip-10-0-9-10 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: chidave01
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-10-0-9-10 ~]$
```

First, I need to login to docker hub using the “docker login” command

Next, I’ll reference the docker documentation to guide me to tag the image properly for docker upload.

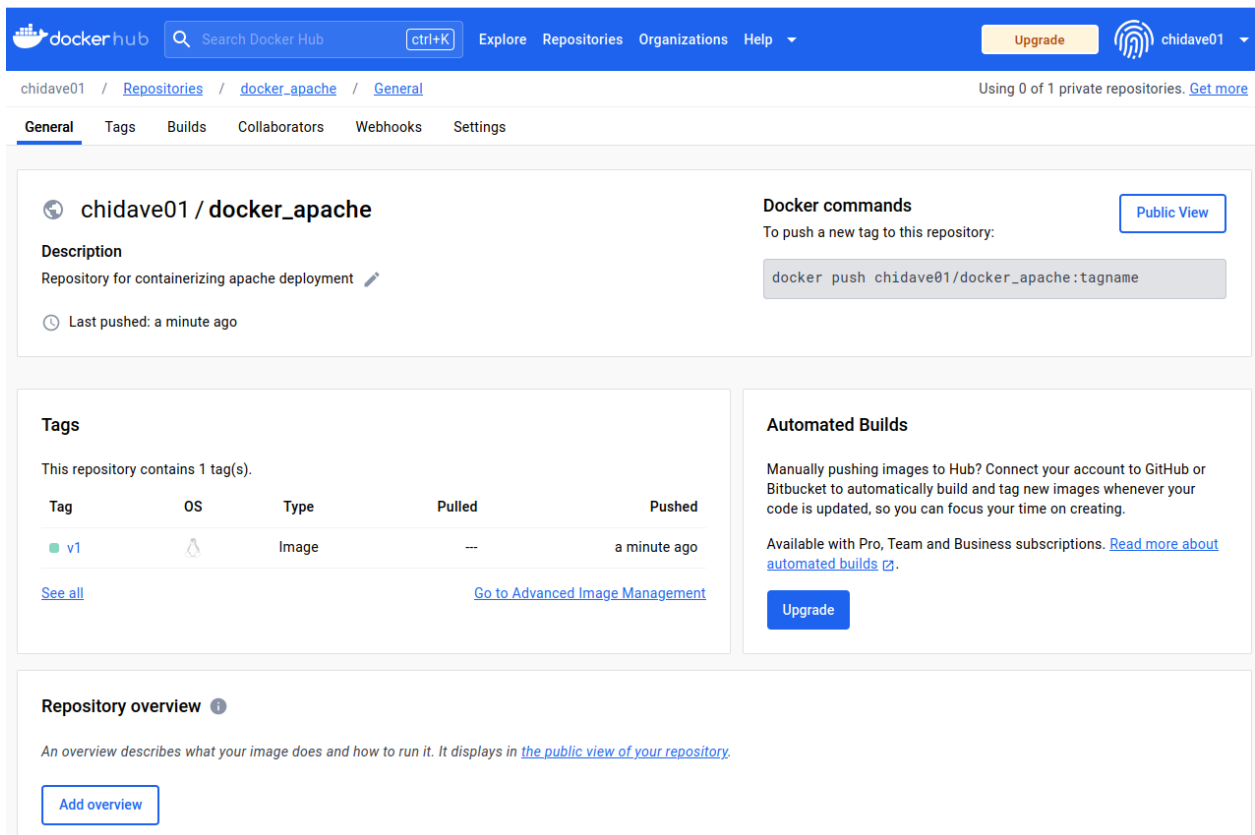
I’ll need to have a repository in my dockerhub account where I want the image to be stored in.



I will be using the docker\_apache repository in my dockerhub account.

Next, I need to tag the image using the “docker tag” command. And push to my dockerhub repository.

```
[ec2-user@ip-10-0-9-10 ~]$ docker images
REPOSITORY      TAG       IMAGE ID       CREATED        SIZE
apacheimage      v1        8c3323a18bb2   41 minutes ago 802MB
ubuntu           latest    e4c58958181a   7 weeks ago   77.8MB
hello-world      latest    9c7a54a9a43c   6 months ago  13.3kB
ubuntu           16.04     b6f507652425   2 years ago   135MB
[ec2-user@ip-10-0-9-10 ~]$ docker tag 8c3323a18bb2 chidave01/docker_apache:v1
[ec2-user@ip-10-0-9-10 ~]$ docker push chidave01/docker_apache:v1
The push refers to repository [docker.io/chidave01/docker_apache]
2868607cc1cf: Pushed
bcaad06ea8e7: Pushed
f10028744f66: Pushed
174f56854903: Pushed
v1: digest: sha256:ceb35cffb1a88f4abe50a849174d48ced19350953b12f31e97dd109a72631898 size: 1161
[ec2-user@ip-10-0-9-10 ~]$
```



The screenshot shows the Docker Hub interface for the repository `chidave01/docker_apache`. The page includes a search bar, navigation links, and a header with the user's profile. The repository page itself has tabs for General, Tags, Builds, Collaborators, Webhooks, and Settings. The General tab is active, showing the repository name, description, and a list of tags. The description is "Repository for containerizing apache deployment". The tags section shows a single tag `v1` with a status of "a minute ago". The Docker commands section shows the command `docker push chidave01/docker_apache:tagname`. The Automated Builds section is also visible, along with a repository overview section.

**chidave01 / docker\_apache**

**Description**  
Repository for containerizing apache deployment

Last pushed: a minute ago

**Docker commands**  
To push a new tag to this repository:  
`docker push chidave01/docker_apache:tagname`

**Tags**  
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
v1		Image	---	a minute ago

[See all](#) [Go to Advanced Image Management](#)

**Automated Builds**  
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.  
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

**Repository overview**  
An overview describes what your image does and how to run it. It displays in [the public view of your repository](#).  
[Add overview](#)

Successfully pushed to the docker hub repo.

**Note:** If you don't have a repo on your dockerhub and you push, dockerhub will create a repo automatically for you.