



UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Dehradun

ACO LAB

Name- Harsha Agarwal

Sap id- 500096741

Roll no- R2142211158

Batch- B-4

Course- Btech Devops

Submitted to- Dr Hitesh Kumar Sharma

EXPERIMENT-5

AIM : Working with Dockerfile to Build and Push Docker Image

1. Create following Dockerfile

FROM alpine

MAINTAINER SUDIPT

RUN apk update

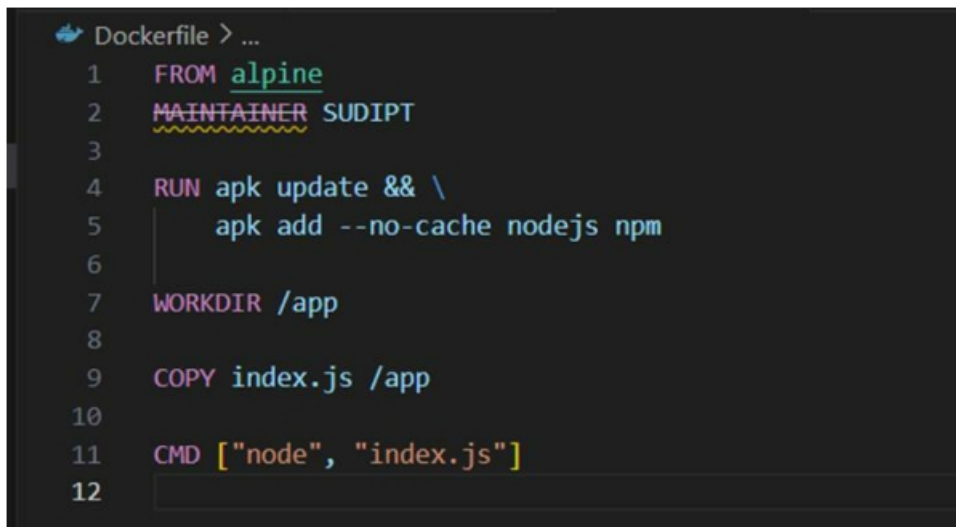
RUN apk add nodejs

RUN mkdir /app

COPY index.js /app

WORKDIR /app

RUN node index.js

A screenshot of a code editor showing a Dockerfile. The file is named 'Dockerfile' and is located in a directory. The code is as follows:

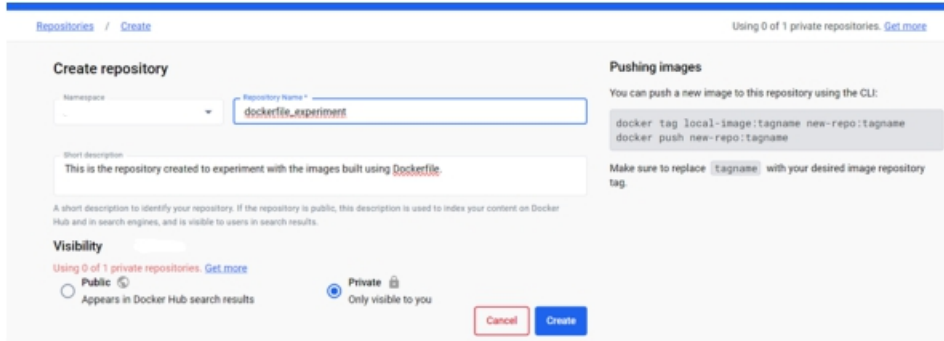
```
1 FROM alpine
2 MAINTAINER SUDIPT
3
4 RUN apk update && \
5     apk add --no-cache nodejs npm
6
7 WORKDIR /app
8
9 COPY index.js /app
10
11 CMD ["node", "index.js"]
12
```

2. Now we have dockerized the app, we will Build image from Dockerfile.
“ **docker build -t myimage:1.0.0 .** ”

```
91983@DELL MINGW64 ~/OneDrive/Desktop/SEM5/ACO-LAB-2021-25(Local)/my-web-app (main)
$ docker build -t myimage:1.0.0 .
[+] Building 65.6s (9/9) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 195B                                0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 125B                                     0.0s
=> [internal] load metadata for docker.io/library/alpine:latest    2.5s
=> CACHED [1/4] FROM docker.io/library/alpine@sha256:eece025e432126ce23f223450a0326fbede39cdf496a85d8c016293fc851978  0.0s
=> [internal] load build context                                   0.0s
=> => transferring context: 29B                                       0.0s
=> [2/4] RUN apk update && apk add --no-cache nodejs npm          62.5s
=> [3/4] WORKDIR /app                                              0.0s
=> [4/4] COPY index.js /app                                        0.0s
=> exporting to image                                              0.5s
=> => exporting layers                                              0.5s
=> => writing image sha256:f7f74f83583bee05f10b14c410ffce181180833039f324df89940bb7f6957957  0.0s
=> => naming to docker.io/library/myimage:1.0.0                   0.0s

What's Next?
View a summary of image vulnerabilities and recommendations + docker scout quickview
```

3. Create account on Dockerhub and create a repository in it.



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


Create repository


Repository Name *

Short description
This is the repository created to experiment with the images built using Dockerfile.

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.

4. Tag the recently created image using following command.
“ **docker tag imageID Repositoryname** ”

```
91983@DELL MINGW64 ~/OneDrive/Desktop/SEM5/ACO-LAB-2021-25(Local)/my-web-app (main)
$ docker tag f7f74f83583b dockerfile_experiment
```

5. Login to Dockerhub from console using following command.
“ **docker login** ”

```
91983@DELL MINGW64 ~/OneDrive/Desktop/SEM5/ACO-LAB-2021-25(Local)/my-web-app (main)
$ docker login
Authenticating with existing credentials...
Login Succeeded
```

6. Now push the image on Dockerhub using following command.
“ **docker push RepositoryName** ”

