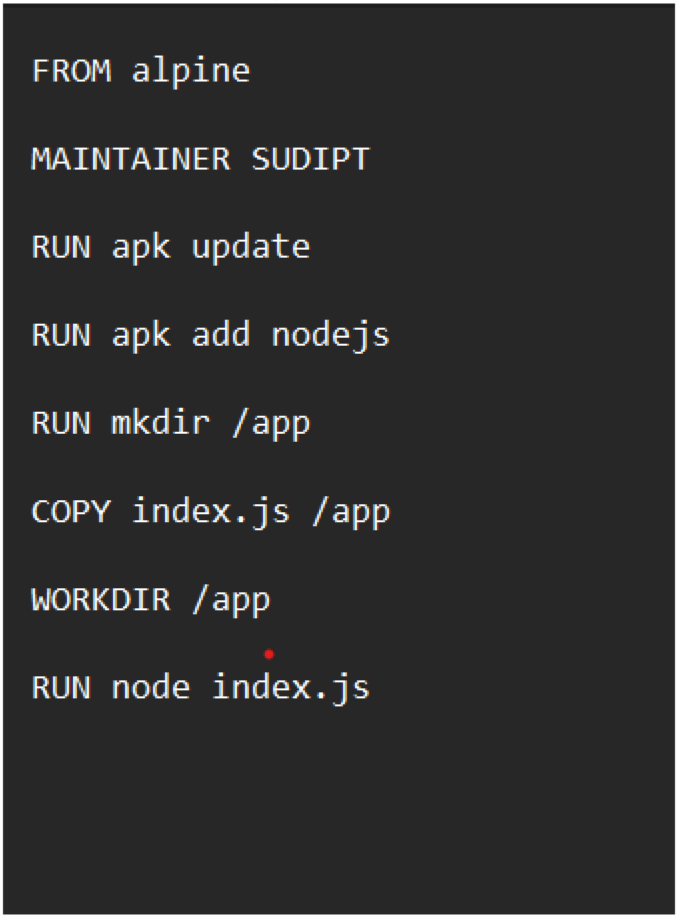
EXPERIMENT 5

AIM: Working with Dockerfile to Build and Push Docker Image

Steps to Complete:

1. Create following Dockerfile

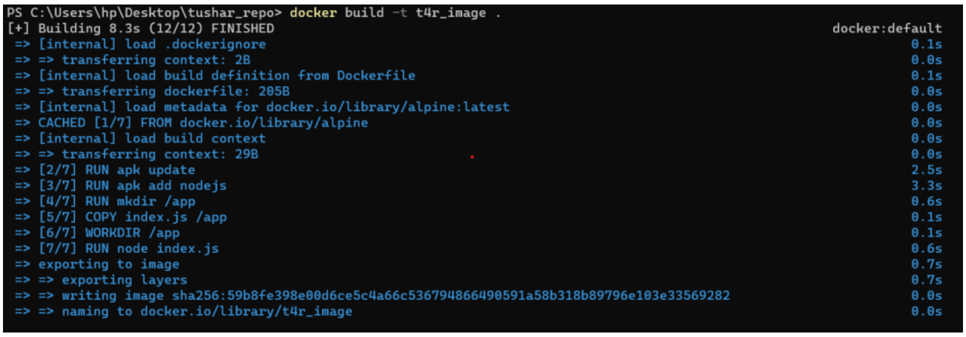
```
FROM alpine
MAINTAINER SUDIPT
RUN apk update
RUN apk add nodejs
RUN mkdir /app
COPY index.js /app
WORKDIR /app
RUN node index.js
```



```
FROM alpine

MAINTAINER SUDIPT

RUN apk update

RUN apk add nodejs

RUN mkdir /app

COPY index.js /app

WORKDIR /app

RUN node index.js
```

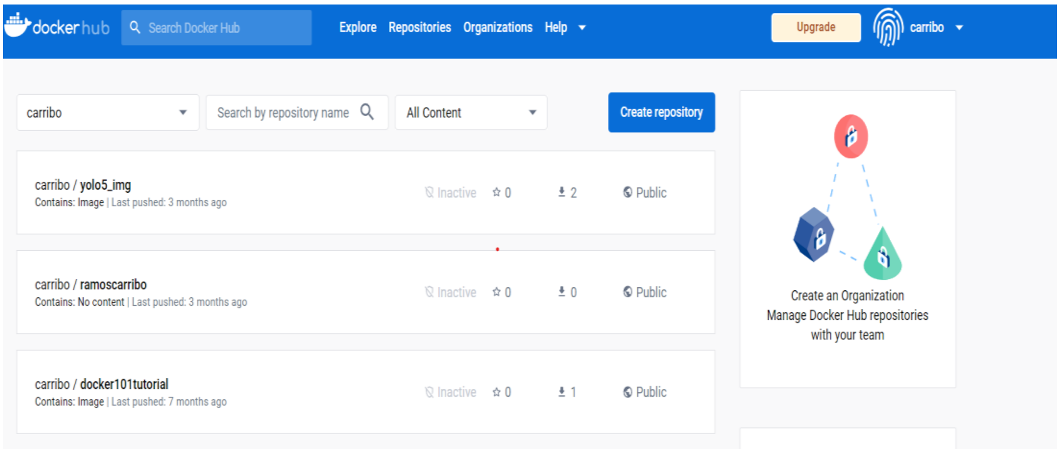
2. Now we have dockerized the app, we will Build image from Dockerfile.

```
docker build -t myimage:0.1 .
```



```
PS C:\Users\hp\Desktop\tushar_repo> docker build -t t4r_image .
[+] Building 8.3s (12/12) FINISHED                                                              docker:default
 => [internal] load .dockerignore                                                                         0.1s
 => => transferring context: 2B                                                                           0.0s
 => [internal] load build definition from Dockerfile                                                      0.1s
 => => transferring dockerfile: 205B                                                                      0.0s
 => [internal] load metadata for docker.io/library/alpine:latest                                          0.0s
 => CACHED [1/7] FROM docker.io/library/alpine                                                            0.0s
 => [internal] load build context                                                                         0.0s
 => => transferring context: 29B                                                                          0.0s
 => [2/7] RUN apk update                                                                                  2.5s
 => [3/7] RUN apk add nodejs                                                                              3.3s
 => [4/7] RUN mkdir /app                                                                                  0.6s
 => [5/7] COPY index.js /app                                                                              0.1s
 => [6/7] WORKDIR /app                                                                                    0.1s
 => [7/7] RUN node index.js                                                                               0.6s
 => exporting to image                                                                                    0.7s
 => => exporting layers                                                                                   0.7s
 => => writing image sha256:59b8fe398e00d6ce5c4a66c536794866490591a58b318b89796e103e33569282              0.0s
 => => naming to docker.io/library/t4r_image                                                              0.0s
```

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

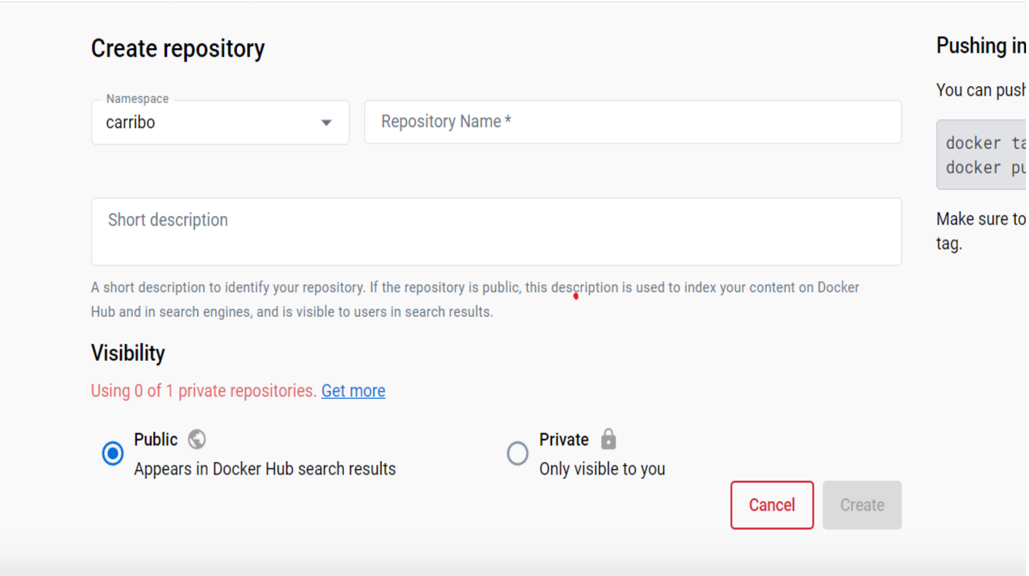


carribo / yolo5_img
Contains: Image | Last pushed: 3 months ago
Inactive ☆ 0 ⭳ 2 Public

carribo / ramoscarribo
Contains: No content | Last pushed: 3 months ago
Inactive ☆ 0 ⭳ 0 Public

carribo / docker101tutorial
Contains: Image | Last pushed: 7 months ago
Inactive ☆ 0 ⭳ 1 Public

Create an Organization
Manage Docker Hub repositories with your team



Create repository

Namespace
carribo

Repository Name *

Short description

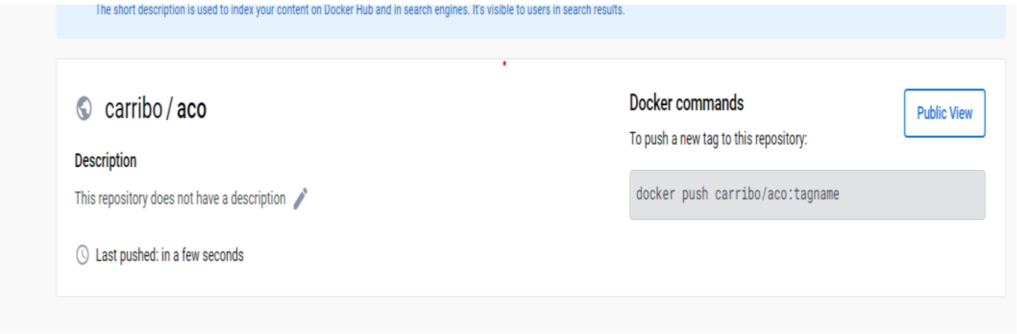
A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

Visibility

Using 0 of 1 private repositories. Get more

● Public
Appears in Docker Hub search results

○ Private
Only visible to you



The short description is used to index your content on Docker Hub and in search engines. It's visible to users in search results.

carribo / aco

Description

This repository does not have a description

Last pushed: in a few seconds

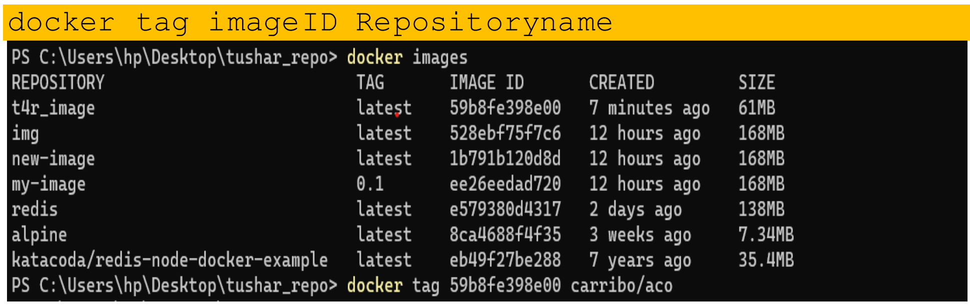
Docker commands

To push a new tag to this repository:

```
docker push carribo/aco:tagname
```

4. Tag the recently created image using following command.

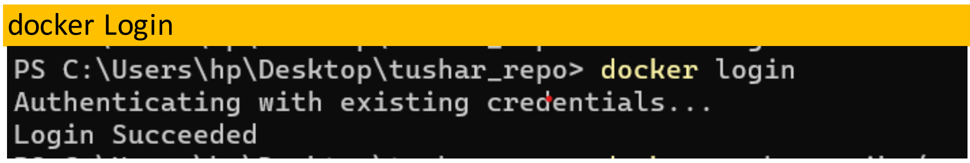
```
docker tag imageID Repositoryname
```



```
PS C:\Users\hp\Desktop\tushar_repo> docker images
REPOSITORY                         TAG       IMAGE ID       CREATED         SIZE
t4r_image                          latest    59b8fe398e00   7 minutes ago   61MB
img                                latest    528ebf75f7c6   12 hours ago    168MB
new-image                          latest    1b791b120d8d   12 hours ago    168MB
my-image                           0.1       ee26eedad720   12 hours ago    168MB
redis                              latest    e579380d4317   2 days ago      138MB
alpine                             latest    8ca4688f4f35   3 weeks ago     7.34MB
katacoda/redis-node-docker-example latest    eb49f27be288   7 years ago     35.4MB
PS C:\Users\hp\Desktop\tushar_repo> docker tag 59b8fe398e00 carribo/aco
```

5. Login to Dockerhub from console using following command.

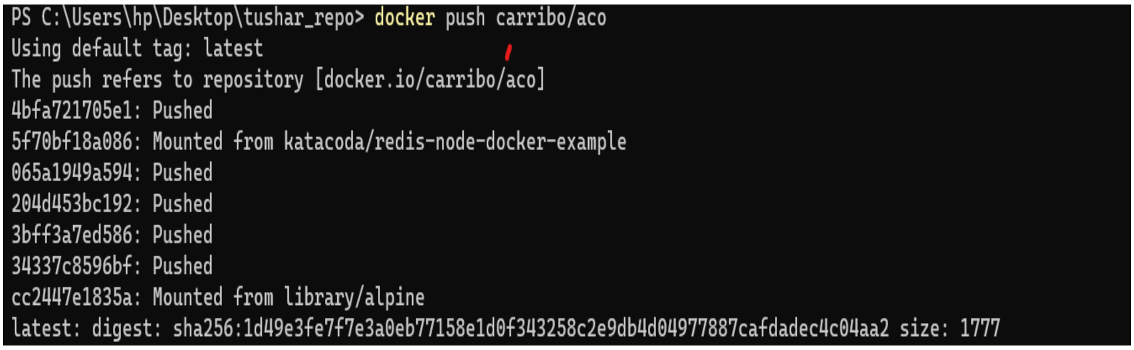
```
docker Login
```



```
PS C:\Users\hp\Desktop\tushar_repo> docker login
Authenticating with existing credentials...
Login Succeeded
```

6. Now push the image on Dockerhub using following command.

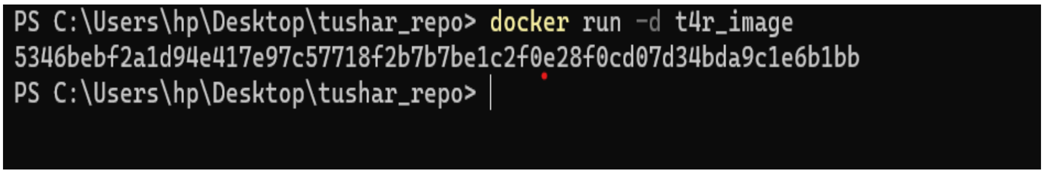
```
docker push Repositoryname
```



```
PS C:\Users\hp\Desktop\tushar_repo> docker push carribo/aco
Using default tag: latest
The push refers to repository [docker.io/carribo/aco]
4bfa721705e1: Pushed
5f70bf18a086: Mounted from katacoda/redis-node-docker-example
065a1949a594: Pushed
204d453bc192: Pushed
3bff3a7ed586: Pushed
34337c8596bf: Pushed
cc2447e1835a: Mounted from library/alpine
latest: digest: sha256:1d49e3fe7f7e3a0eb77158e1d0f343258c2e9db4d04977887cafdadec4c04aa2 size: 1777
```

7. Pull and Run the container of your deployed image on docker hub.

```
Docker run -d image_name
```



```
PS C:\Users\hp\Desktop\tushar_repo> docker run -d t4r_image
5346bebf2a1d94e417e97c57718f2b7b7be1c2f0e28f0cd07d34bda9c1e6b1bb
PS C:\Users\hp\Desktop\tushar_repo> |
```