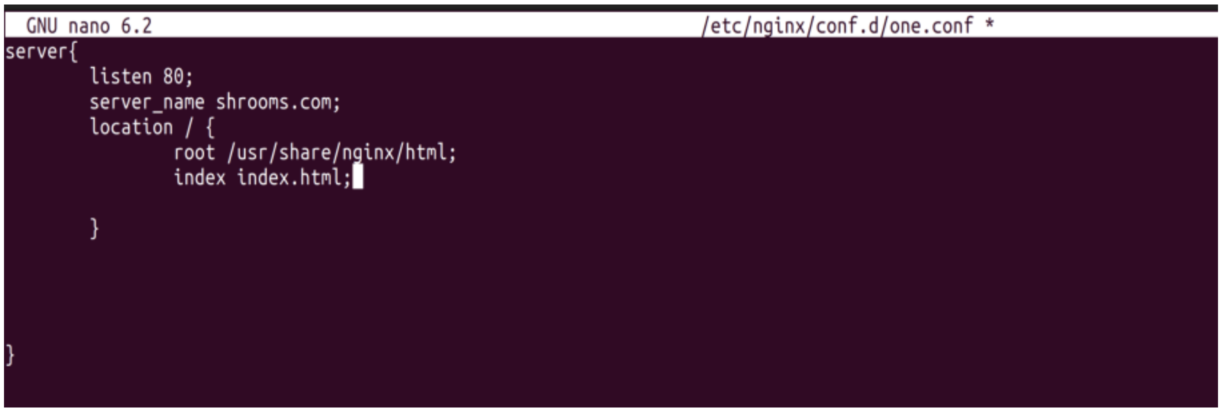
# Assignment 7- Sharams Kunwar – Docker Compose

## Hands-On:

## Number 1: Installing Docker and creating a container (nginx) with docker cli (volume, network, port).

At first, a nginx configuration file was created, which basically serves the default nginx page.

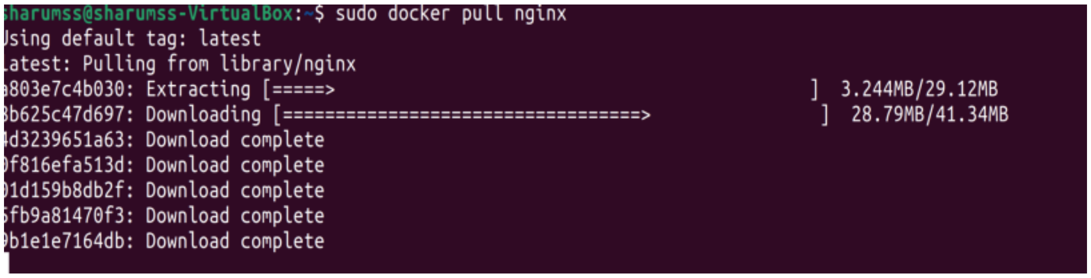
**Screenshot:**



```
  GNU nano 6.2                                         /etc/nginx/conf.d/one.conf *
server{
        listen 80;
        server_name shrooms.com;
        location / {
                root /usr/share/nginx/html;
                index index.html;

        }



}
```

After that, the nginx image was pulled to create a container.

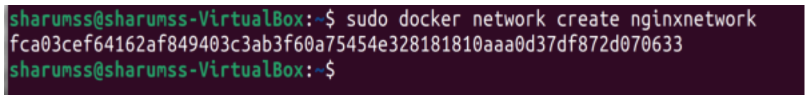
**Screenshot:**



```
harumss@sharumss-VirtualBox:~$ sudo docker pull nginx
Using default tag: latest
atest: Pulling from library/nginx
803e7c4b030: Extracting [=====>                                              ]  3.244MB/29.12MB
b625c47d697: Downloading [=================================>                 ]  28.79MB/41.34MB
d3239651a63: Download complete
f816efa513d: Download complete
1d159b8db2f: Download complete
fb9a81470f3: Download complete
b1e1e7164db: Download complete
```

Then a network named 'nginxnetwork' was created.

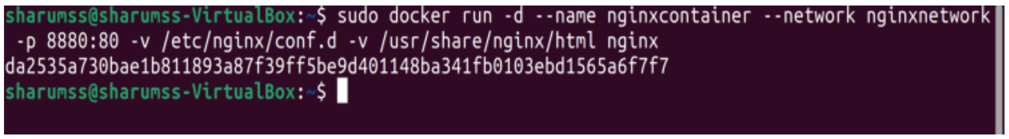
**Screenshot:**



```
sharumss@sharumss-VirtualBox:~$ sudo docker network create nginxnetwork
fca03cef64162af849403c3ab3f60a75454e328181810aaa0d37df872d070633
sharumss@sharumss-VirtualBox:~$
```

Following the network creation, a container was created using the image pulled above and run in detached mode. The container was named 'nginxcontainer', connected to Docker network created above, mapped to port 8880 on the host machine to port 80 inside the container. Similarly, the /etc/nginx/conf.d directory was mounted to the one inside the container and /usr/share/nginx/html was mounted to that of container as well.

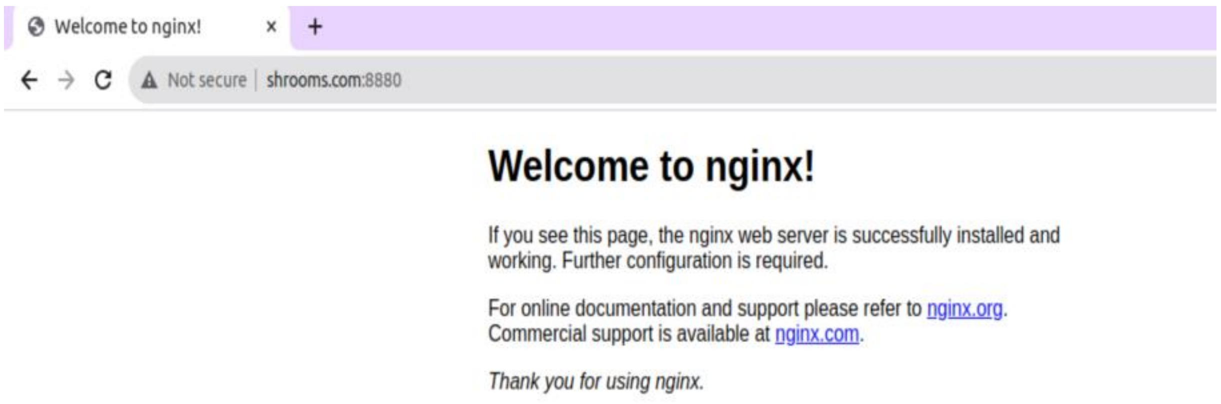
**Screenshot:**



```
sharumss@sharumss-VirtualBox:~$ sudo docker run -d --name nginxcontainer --network nginxnetwork
 -p 8880:80 -v /etc/nginx/conf.d -v /usr/share/nginx/html nginx
da2535a730bae1b811893a87f39ff5be9d401148ba341fb0103ebd1565a6f7f7
sharumss@sharumss-VirtualBox:~$
```

After the container was running, we could access the nginx welcome page from host machine by navigating to http://localhost:8880.

**Screenshot:**



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

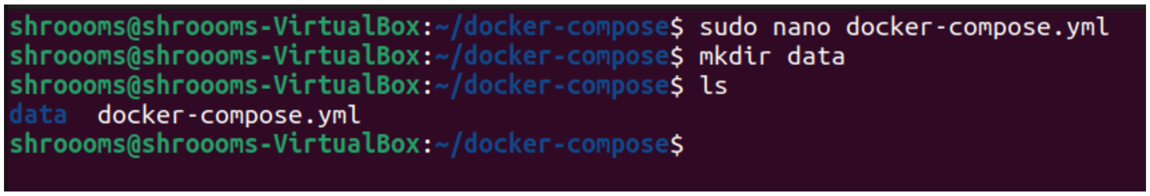
*Thank you for using nginx.*

## Number 2: creating a docker container with docker-compose file for mysql8.

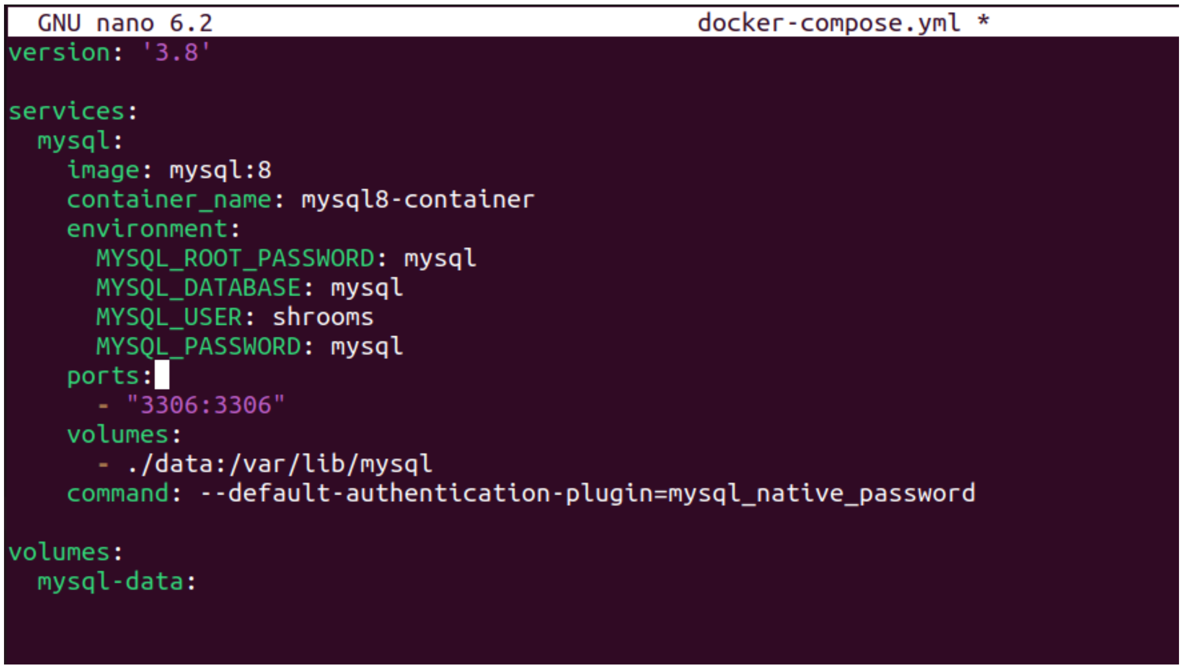
A docker-compose file was created for mysql8 in docker-compose directory. Below is the summary of configuration of the file.

- Version 3.8 of Docker Compose file was used.
- ‘mysql’ service was used defining a mysql container which would use ‘mysql:8’ image from DockerHub, exposing port 3306 and mounting the /data directory on host machine to /var/lib/mysql directory inside the container.
- Similarly, the service ‘mysql-data’ defines the volume where MySQL database data is stored.

**Screenshot:**



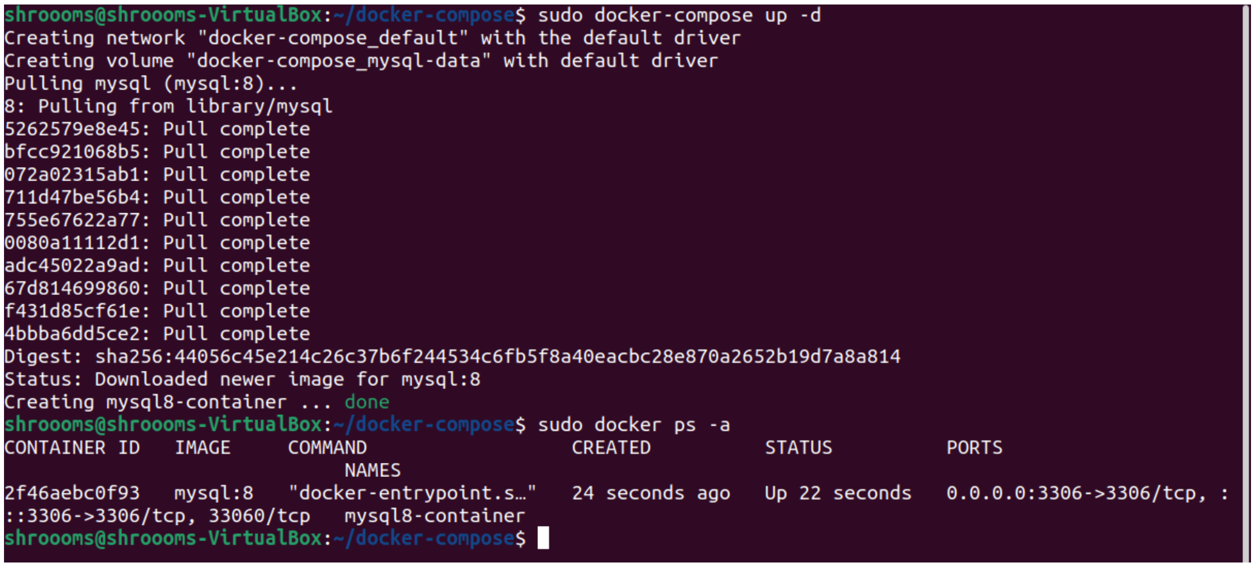
```
shroooms@shroooms-VirtualBox:~/docker-compose$ sudo nano docker-compose.yml
shroooms@shroooms-VirtualBox:~/docker-compose$ mkdir data
shroooms@shroooms-VirtualBox:~/docker-compose$ ls
data  docker-compose.yml
shroooms@shroooms-VirtualBox:~/docker-compose$
```



```
  GNU nano 6.2                                    docker-compose.yml *
version: '3.8'

services:
  mysql:
    image: mysql:8
    container_name: mysql8-container
    environment:
      MYSQL_ROOT_PASSWORD: mysql
      MYSQL_DATABASE: mysql
      MYSQL_USER: shrooms
      MYSQL_PASSWORD: mysql
    ports:
      - "3306:3306"
    volumes:
      - ./data:/var/lib/mysql
    command: --default-authentication-plugin=mysql_native_password

volumes:
  mysql-data:
```

Then, the command sudo docker-compose up -d was used to start the container in detached mode.

**Screenshot:**

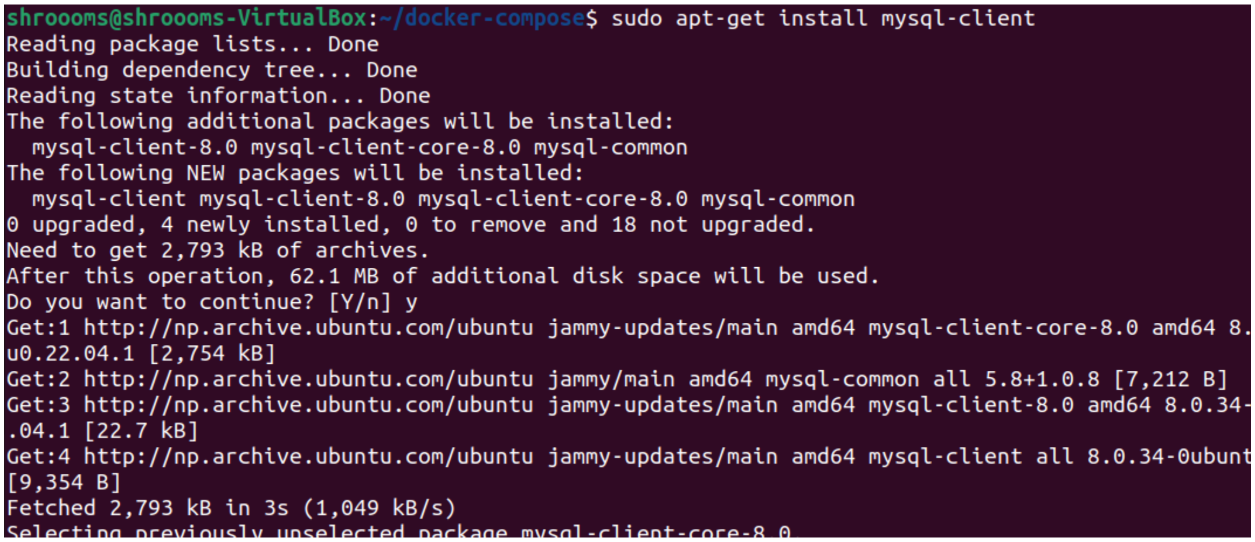


```
shroooms@shroooms-VirtualBox:~/docker-compose$ sudo docker-compose up -d
Creating network "docker-compose_default" with the default driver
Creating volume "docker-compose_mysql-data" with default driver
Pulling mysql (mysql:8)...
8: Pulling from library/mysql
5262579e8e45: Pull complete
bfcc921068b5: Pull complete
072a02315ab1: Pull complete
711d47be56b4: Pull complete
755e67622a77: Pull complete
0080a11112d1: Pull complete
adc45022a9ad: Pull complete
67d814699860: Pull complete
f431d85cf61e: Pull complete
4bbba6dd5ce2: Pull complete
Digest: sha256:44056c45e214c26c37b6f244534c6fb5f8a40eacbc28e870a2652b19d7a8a814
Status: Downloaded newer image for mysql:8
Creating mysql8-container ... done
shroooms@shroooms-VirtualBox:~/docker-compose$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
                              NAMES
2f46aebc0f93   mysql:8   "docker-entrypoint.s…"   24 seconds ago   Up 22 seconds   0.0.0.0:3306->3306/tcp, :
::3306->3306/tcp, 33060/tcp   mysql8-container
shroooms@shroooms-VirtualBox:~/docker-compose$ 
```

As we can see, the container is up and running on using command 'sudo docker ps -a''

Then, to access MySQL database, mysql-client was installed.

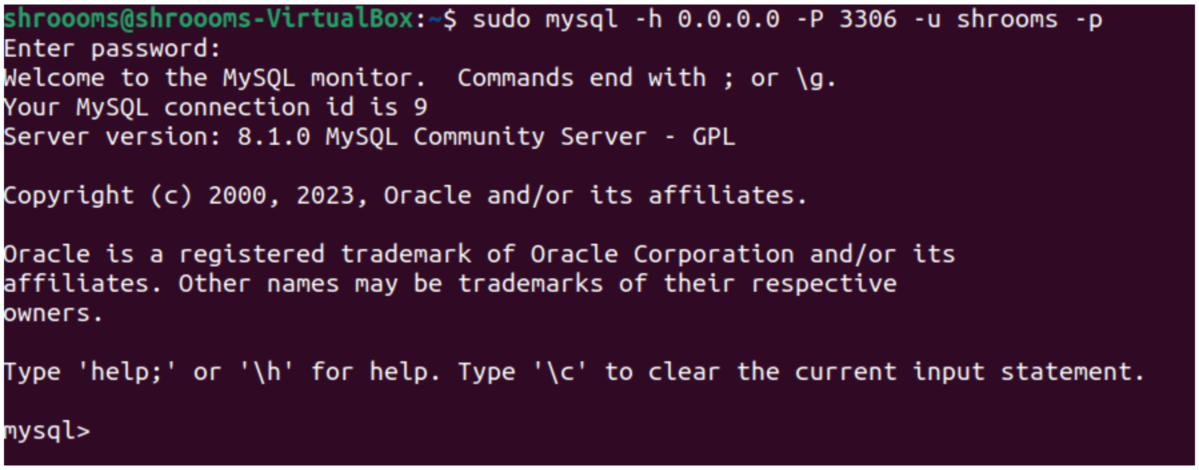
**Screenshot:**



```
shroooms@shroooms-VirtualBox:~/docker-compose$ sudo apt-get install mysql-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  mysql-client-8.0 mysql-client-core-8.0 mysql-common
The following NEW packages will be installed:
  mysql-client mysql-client-8.0 mysql-client-core-8.0 mysql-common
0 upgraded, 4 newly installed, 0 to remove and 18 not upgraded.
Need to get 2,793 kB of archives.
After this operation, 62.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client-core-8.0 amd64 8.
u0.22.04.1 [2,754 kB]
Get:2 http://np.archive.ubuntu.com/ubuntu jammy/main amd64 mysql-common all 5.8+1.0.8 [7,212 B]
Get:3 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client-8.0 amd64 8.0.34-
.04.1 [22.7 kB]
Get:4 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client all 8.0.34-0ubunt
[9,354 B]
Fetched 2,793 kB in 3s (1,049 kB/s)
Selecting previously unselected package mysql-client-core-8.0
```

Then, MySQL database was accessed using the credentials defined in the docker-compose file created above:

**Screenshot:**

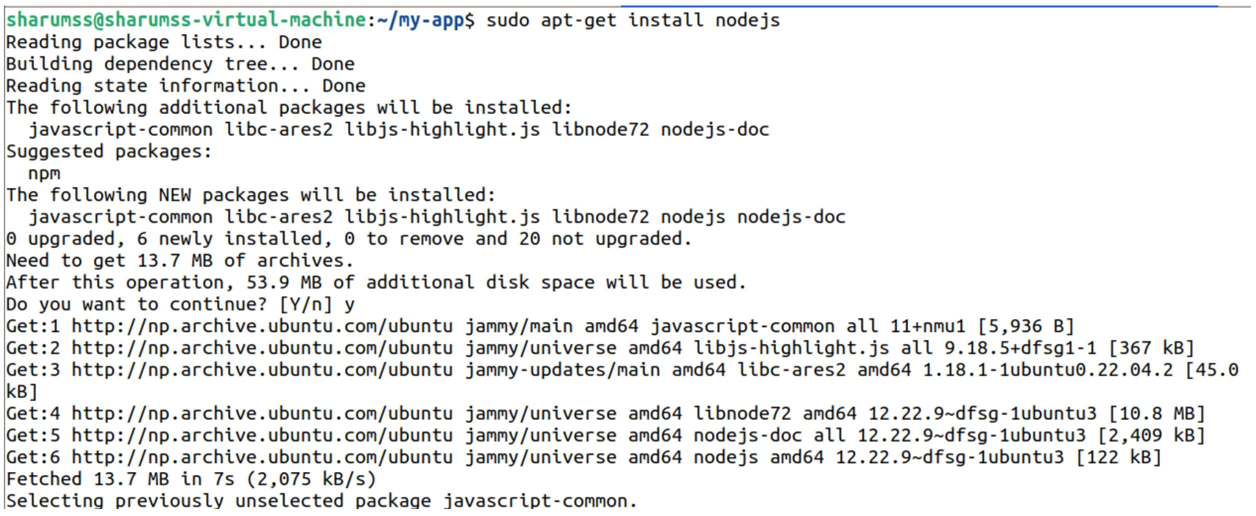


```
shroooms@shroooms-VirtualBox:~$ sudo mysql -h 0.0.0.0 -P 3306 -u shrooms -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, 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>
```

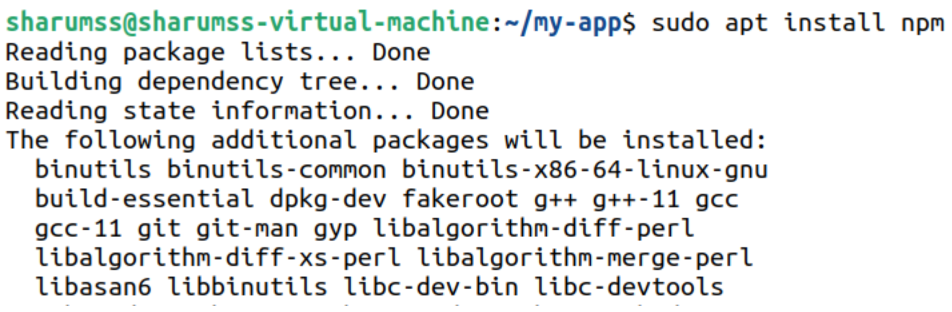
## Number 3: creating a react app with node js and nginx using dockerfile with docker compose file.

At first, nodejs and npm was installed in host machine to create a react project.

**Screenshot:**



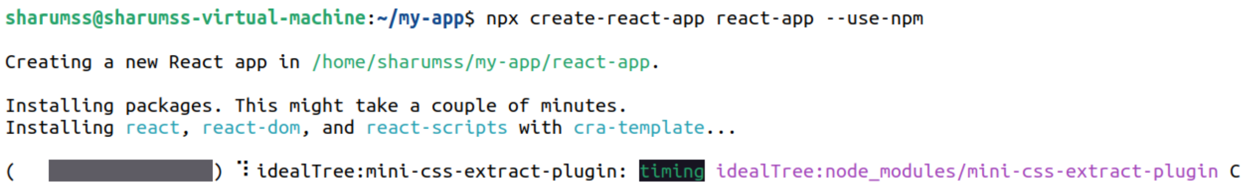
```
sharumss@sharumss-virtual-machine:~/my-app$ sudo apt-get install nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  javascript-common libc-ares2 libjs-highlight.js libnode72 nodejs-doc
Suggested packages:
  npm
The following NEW packages will be installed:
  javascript-common libc-ares2 libjs-highlight.js libnode72 nodejs nodejs-doc
0 upgraded, 6 newly installed, 0 to remove and 20 not upgraded.
Need to get 13.7 MB of archives.
After this operation, 53.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://np.archive.ubuntu.com/ubuntu jammy/main amd64 javascript-common all 11+nmu1 [5,936 B]
Get:2 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 libjs-highlight.js all 9.18.5+dfsg1-1 [367 kB]
Get:3 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libc-ares2 amd64 1.18.1-1ubuntu0.22.04.2 [45.0
kB]
Get:4 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 libnode72 amd64 12.22.9~dfsg-1ubuntu3 [10.8 MB]
Get:5 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 nodejs-doc all 12.22.9~dfsg-1ubuntu3 [2,409 kB]
Get:6 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 nodejs amd64 12.22.9~dfsg-1ubuntu3 [122 kB]
Fetched 13.7 MB in 7s (2,075 kB/s)
Selecting previously unselected package javascript-common.
```



```
sharumss@sharumss-virtual-machine:~/my-app$ sudo apt install npm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu
  build-essential dpkg-dev fakeroot g++ g++-11 gcc
  gcc-11 git git-man gyp libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan6 libbinutils libc-dev-bin libc-devtools
```

Then, a react project was initialized using command ‘npx create-react-app react-app –use-npm’ inside my-app directory.

**Screenshot:**

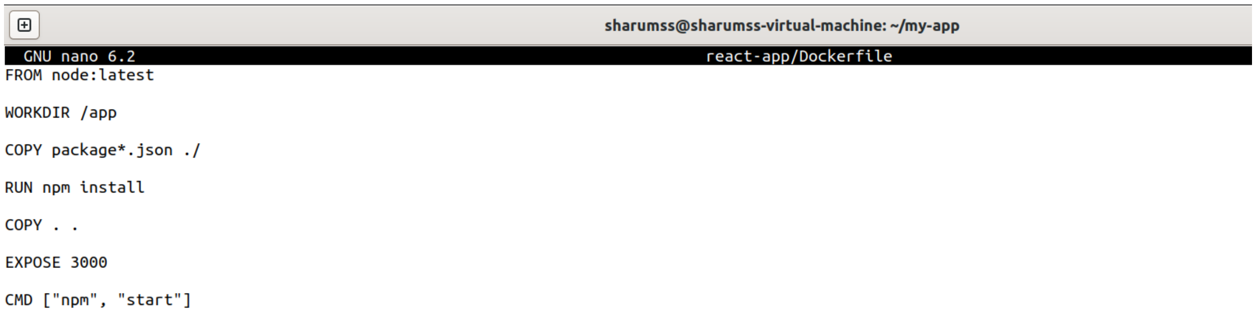


```
sharumss@sharumss-virtual-machine:~/my-app$ npx create-react-app react-app --use-npm

Creating a new React app in /home/sharumss/my-app/react-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

(                  ) ⠋ idealTree:mini-css-extract-plugin: timing idealTree:node_modules/mini-css-extract-plugin C
```

Following the project initialization, a Dockerfile was created inside of it outlining following configuration:

- Pulling the latest node image from Docker Hub.
- The working directory ‘app’ was initialized.
- Package.json and other dependencies to be copied inside the working directory.
- Then run command was used to install npm, followed by copying all files inside the working directory.
- Port 3000 was exposed and command npm start to be run to start the react application.

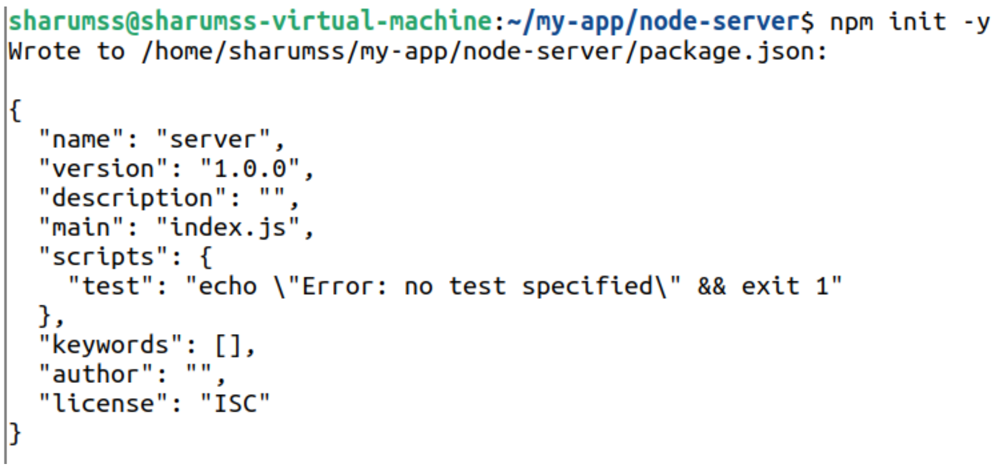
**Screemshot:**



```
sharumss@sharumss-virtual-machine: ~/my-app
  GNU nano 6.2                          react-app/Dockerfile
FROM node:latest

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

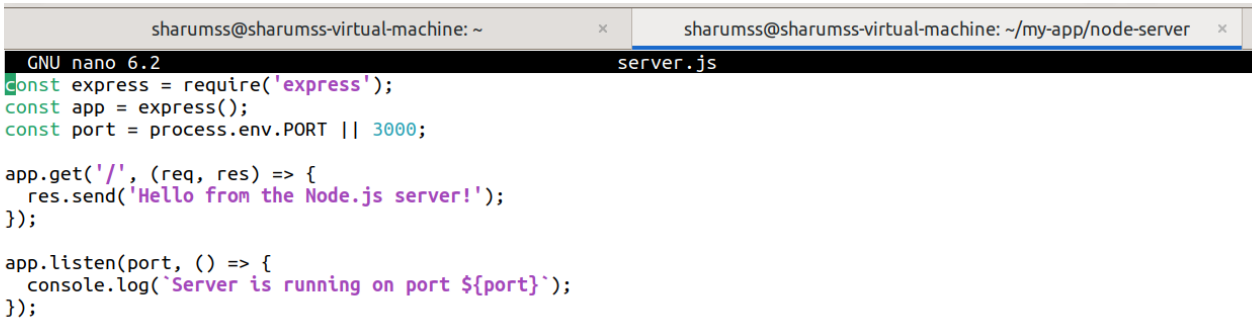
Then, a node server was initialized using the command 'npm init -y' inside node-server directory.

**Screenshot:**



```
sharumss@sharumss-virtual-machine:~/my-app/node-server$ npm init -y
Wrote to /home/sharumss/my-app/node-server/package.json:

{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

After initializing, server.js configuration was edited as follows:
**Screenshot:**


sharumss@sharumss-virtual-machine: ~
sharumss@sharumss-virtual-machine: ~/my-app/node-server

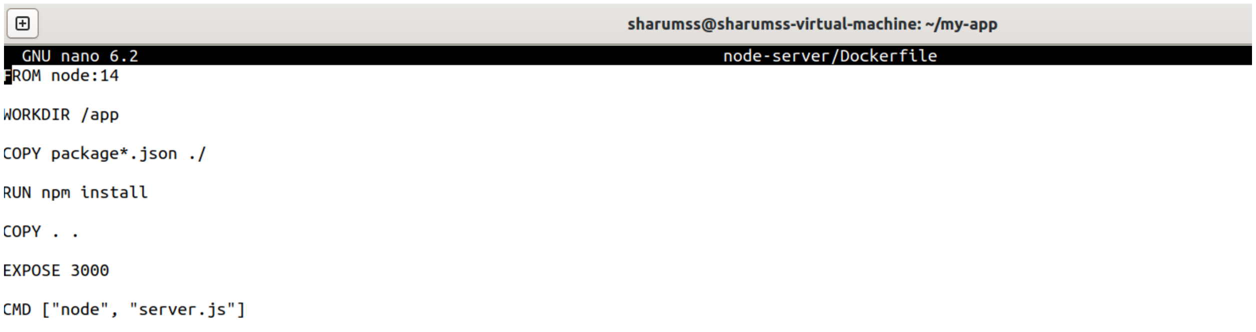
```
  GNU nano 6.2                                        server.js
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;

app.get('/', (req, res) => {
  res.send('Hello from the Node.js server!');
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

It basically displays 'Hello from the Node.js server' on accessing it.

Then a Dockerfile was created inside of node-server directory outlining following configuration:

- Start with the node:latest base image.
- Create a working directory at /app.
- Copy the package.json file to the working directory.
- Install the dependencies using npm install.
- Copy all other files from the current directory to the working directory.
- Expose port 3000.
- Run the npm start command to start the Node.js application.

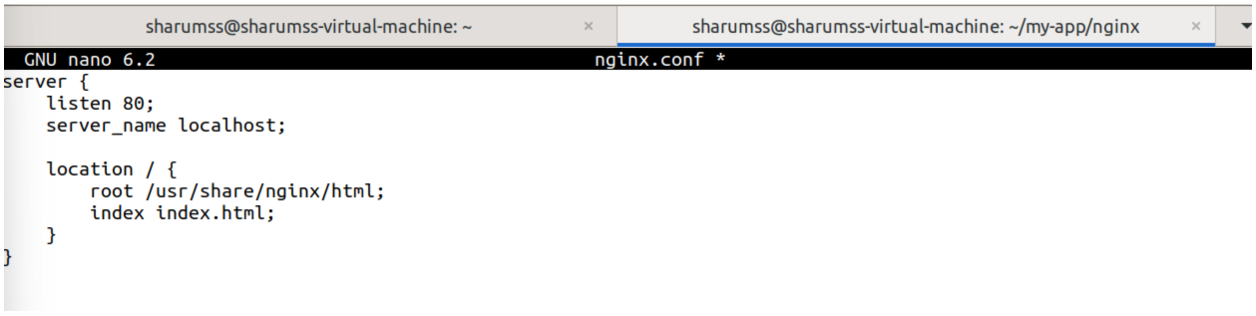
**Screenshot:**



```
sharumss@sharumss-virtual-machine: ~/my-app
  GNU nano 6.2                                    node-server/Dockerfile
FROM node:14

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node", "server.js"]
```

Then, a nginx configuration was created inside nginx directory inside my-app directory. The configuration basically displays default nginx page when accessed.

**Screenshot:**

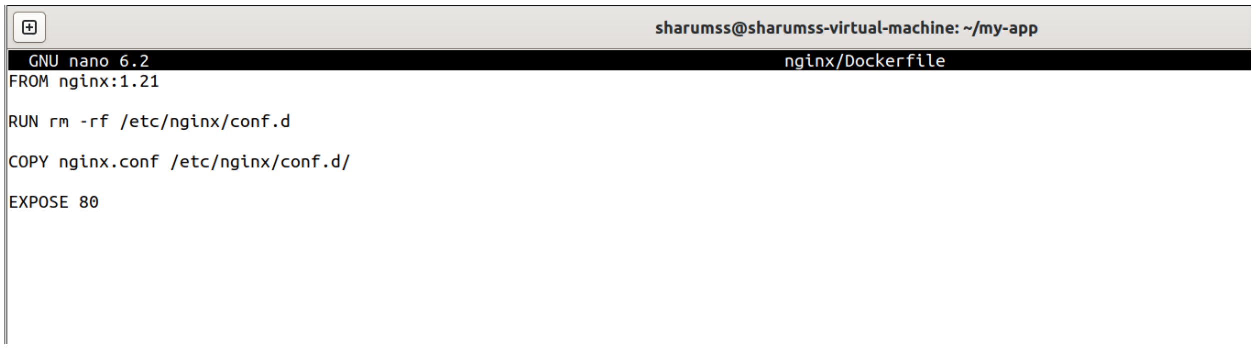


```
sharumss@sharumss-virtual-machine: ~        sharumss@sharumss-virtual-machine: ~/my-app/nginx
  GNU nano 6.2                                nginx.conf *
server {
    listen 80;
    server_name localhost;

    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}
```

Similarly, a Dockerfile was created inside of nginx directory outlining following configuration:

- Start with nginx:1.21 base image.
- Run command to delete the default conf.d and replace it with the configuration created above.
- Expose port 80.

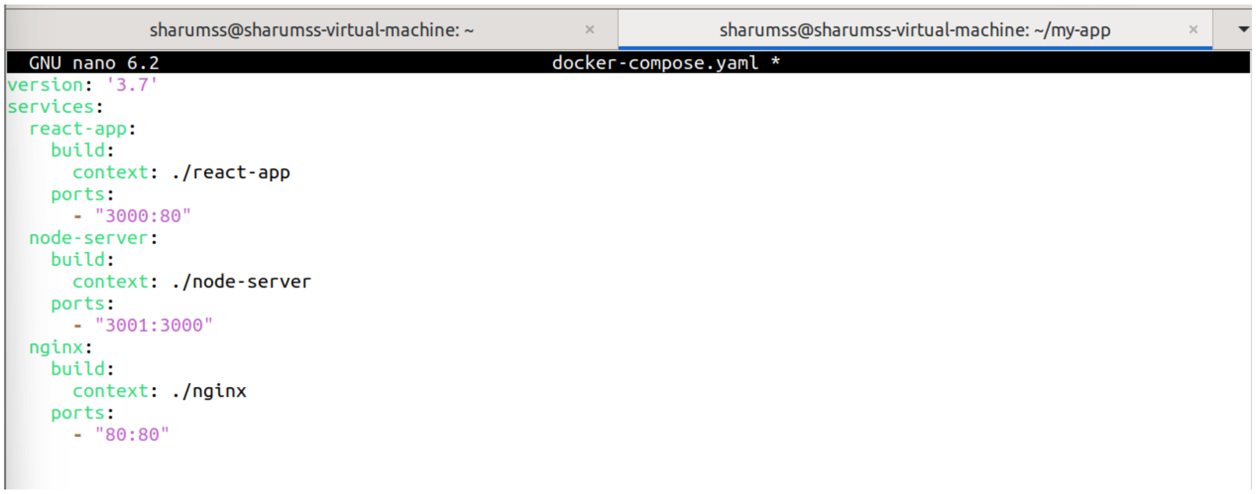
**Screenshot:**



```
sharumss@sharumss-virtual-machine: ~/my-app
 GNU nano 6.2                              nginx/Dockerfile
FROM nginx:1.21

RUN rm -rf /etc/nginx/conf.d

COPY nginx.conf /etc/nginx/conf.d/

EXPOSE 80
```

Finally, a docker compose file was created specifying all three services.

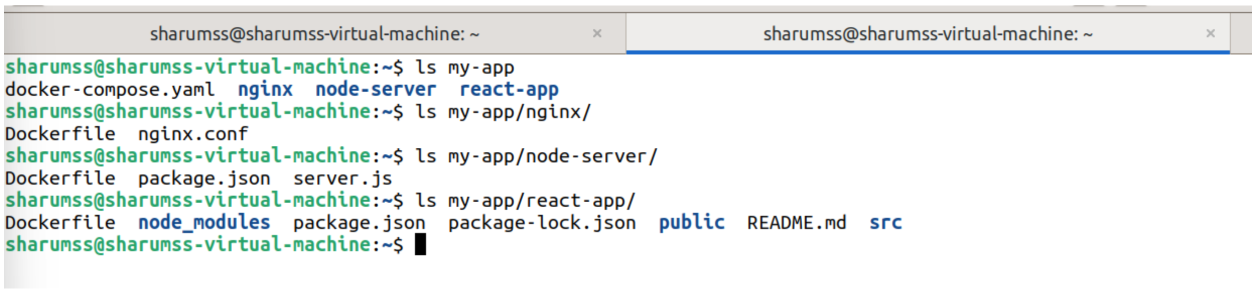
**Screenshot:**



```
sharumss@sharumss-virtual-machine: ~        sharumss@sharumss-virtual-machine: ~/my-app
 GNU nano 6.2                    docker-compose.yaml *
version: '3.7'
services:
  react-app:
    build:
      context: ./react-app
    ports:
      - "3000:80"
  node-server:
    build:
      context: ./node-server
    ports:
      - "3001:3000"
  nginx:
    build:
      context: ./nginx
    ports:
      - "80:80"
```

Below is the screenshot of all created files inside of my-app directory.

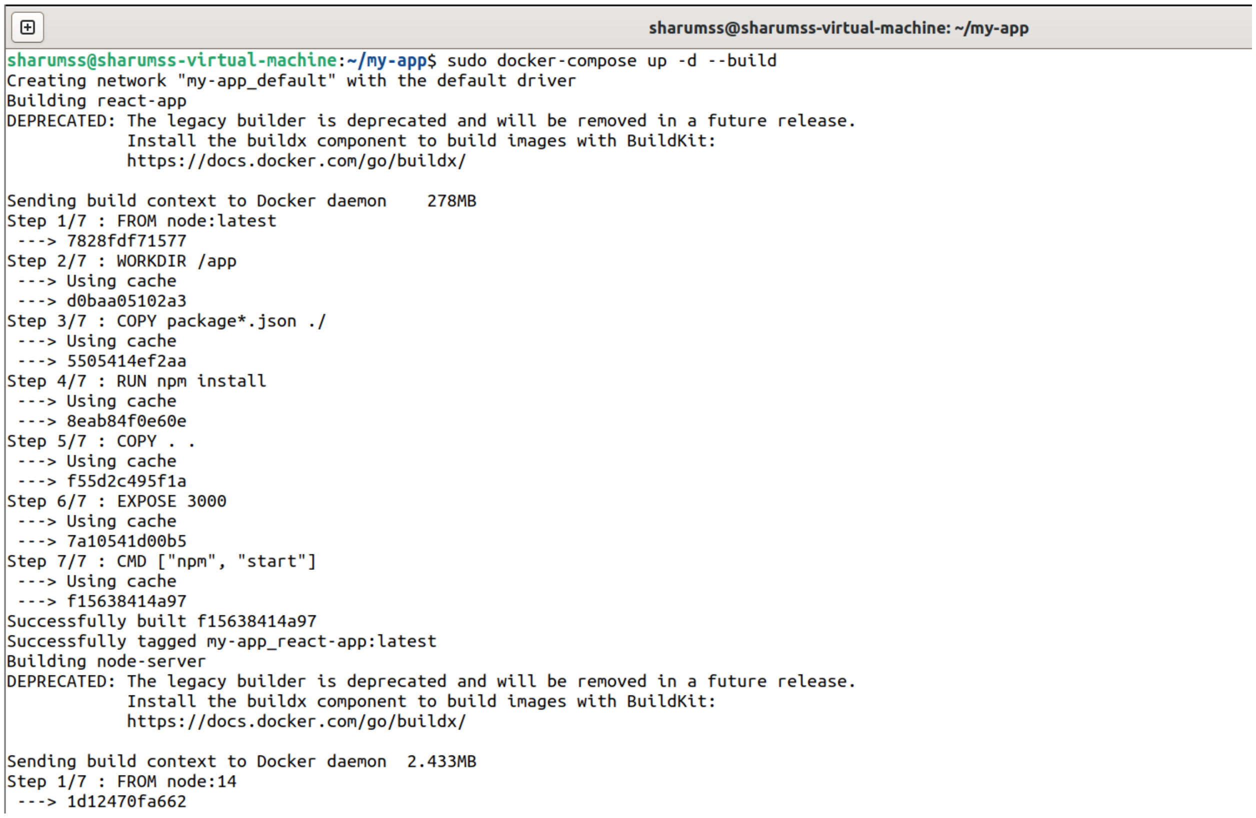
**Screenshot:**



```
sharumss@sharumss-virtual-machine: ~        sharumss@sharumss-virtual-machine: ~
sharumss@sharumss-virtual-machine:~$ ls my-app
docker-compose.yaml  nginx  node-server  react-app
sharumss@sharumss-virtual-machine:~$ ls my-app/nginx/
Dockerfile  nginx.conf
sharumss@sharumss-virtual-machine:~$ ls my-app/node-server/
Dockerfile  package.json  server.js
sharumss@sharumss-virtual-machine:~$ ls my-app/react-app/
Dockerfile  node_modules  package.json  package-lock.json  public  README.md  src
sharumss@sharumss-virtual-machine:~$ 
```

Then, the docker-file was build and the services defined in the compose file was started using 'docker-compose up -d --build'.

**Screenshot:**

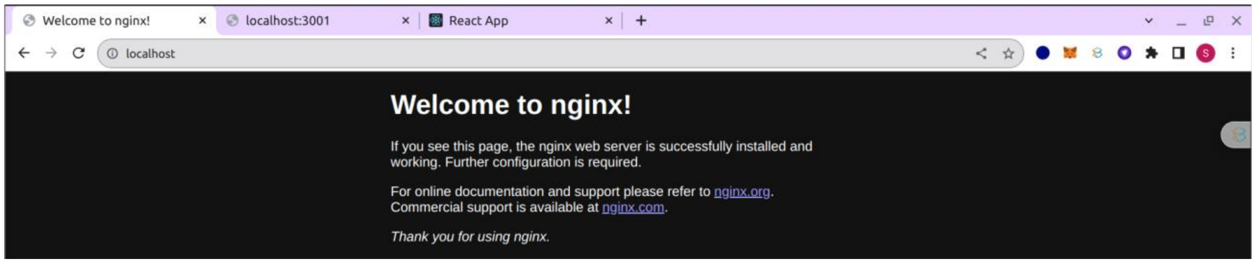


sharumss@sharumss-virtual-machine: ~/my-app

```
sharumss@sharumss-virtual-machine:~/my-app$ sudo docker-compose up -d --build
Creating network "my-app_default" with the default driver
Building react-app
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon    278MB
Step 1/7 : FROM node:latest
 ---> 7828fdf71577
Step 2/7 : WORKDIR /app
 ---> Using cache
 ---> d0baa05102a3
Step 3/7 : COPY package*.json ./
 ---> Using cache
 ---> 5505414ef2aa
Step 4/7 : RUN npm install
 ---> Using cache
 ---> 8eab84f0e60e
Step 5/7 : COPY . .
 ---> Using cache
 ---> f55d2c495f1a
Step 6/7 : EXPOSE 3000
 ---> Using cache
 ---> 7a10541d00b5
Step 7/7 : CMD ["npm", "start"]
 ---> Using cache
 ---> f15638414a97
Successfully built f15638414a97
Successfully tagged my-app_react-app:latest
Building node-server
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  2.433MB
Step 1/7 : FROM node:14
 ---> 1d12470fa662
```

After the containers were up and running, we could navigate all the pages respectively.

**Screenshots:**



Welcome to nginx! | localhost:3001 | React App

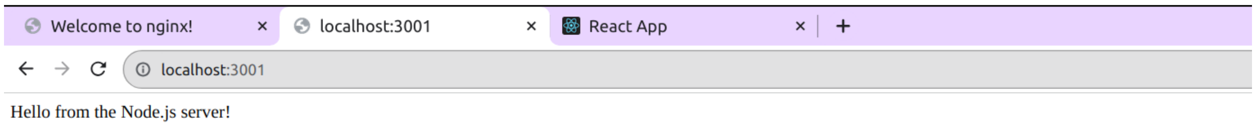
localhost

**Welcome to nginx!**

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

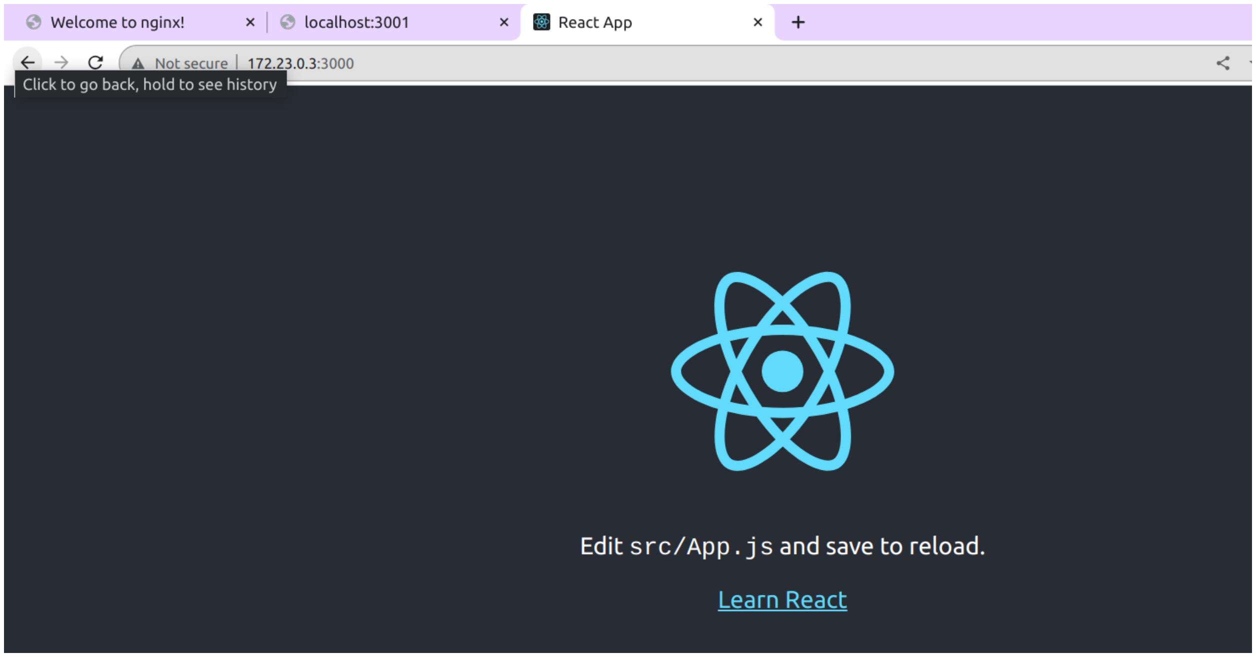
*Thank you for using nginx.*



Welcome to nginx! | localhost:3001 | React App

localhost:3001

```
Hello from the Node.js server!
```



Welcome to nginx! | localhost:3001 | React App

Not secure | 172.23.0.3:3000

Click to go back, hold to see history

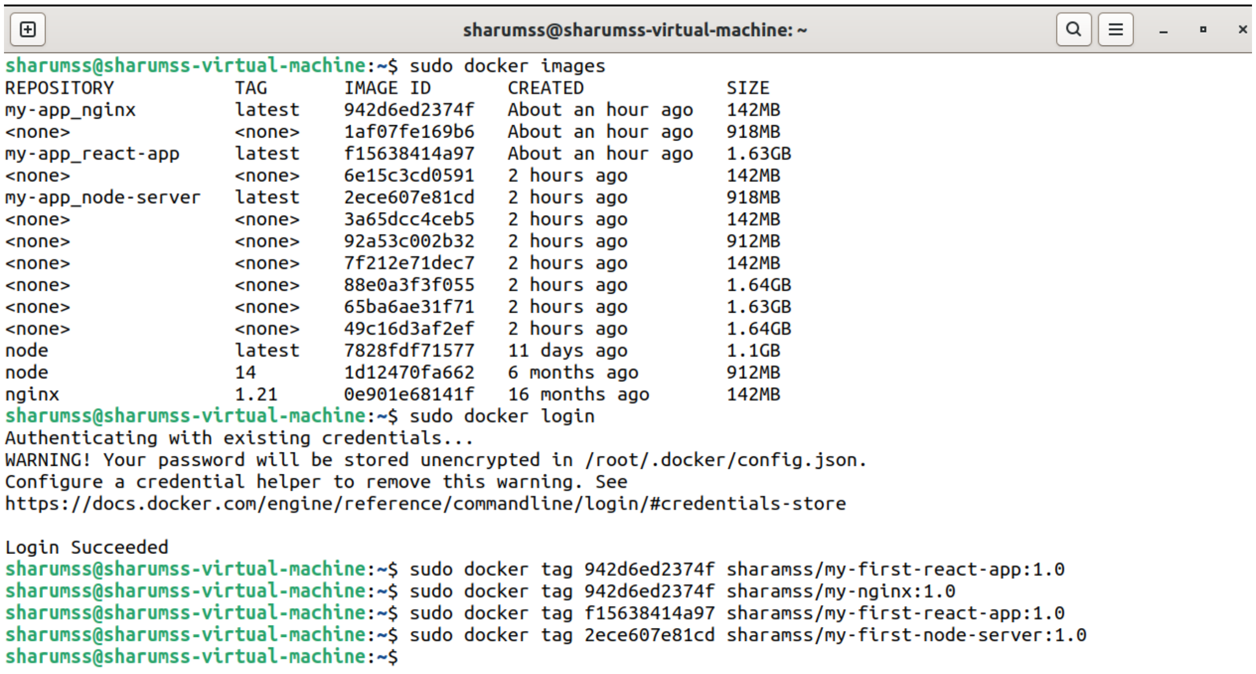
Edit `src/App.js` and save to reload.

Learn React

## Number 4: pushing the images to Docker Hub

The push the then created images, we logged in to Docker Hub and then the created images were tagged as per the nomenclature.

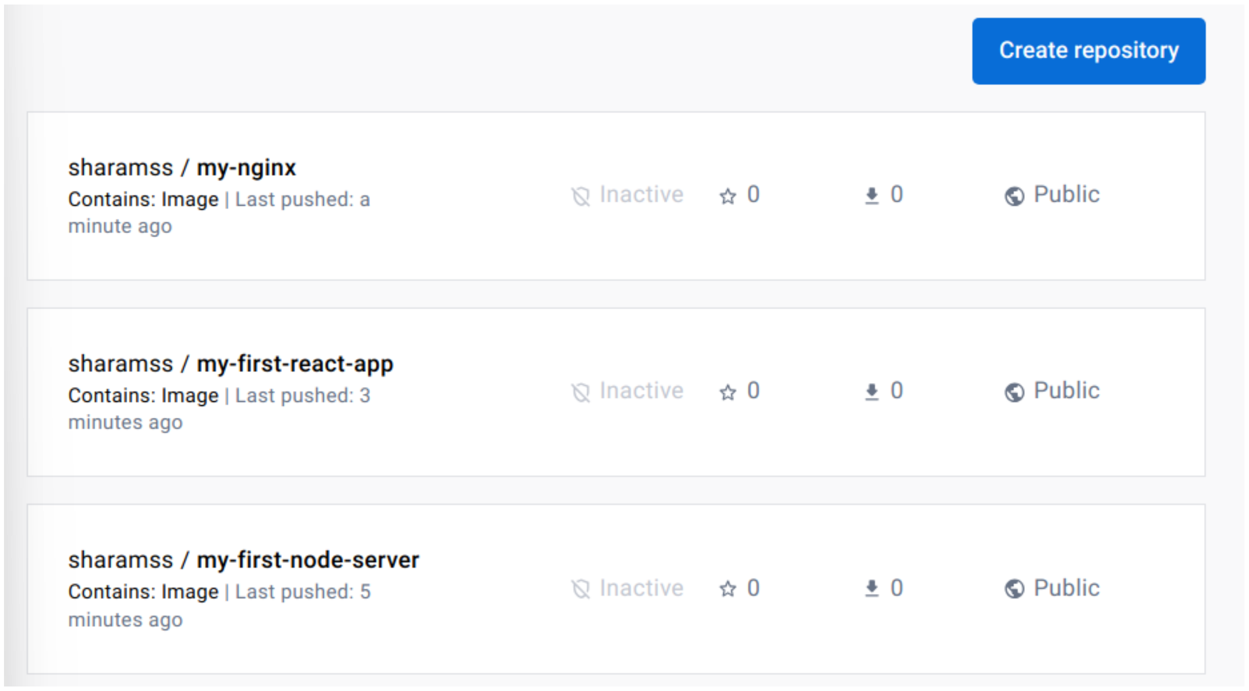
**Screenshot:**



```
sharumss@sharumss-virtual-machine:~$ sudo docker images
REPOSITORY             TAG       IMAGE ID       CREATED            SIZE
my-app_nginx           latest    942d6ed2374f   About an hour ago   142MB
<none>                 <none>    1af07fe169b6   About an hour ago   918MB
my-app_react-app       latest    f15638414a97   About an hour ago   1.63GB
<none>                 <none>    6e15c3cd0591   2 hours ago         142MB
my-app_node-server     latest    2ece607e81cd   2 hours ago         918MB
<none>                 <none>    3a65dcc4ceb5   2 hours ago         142MB
<none>                 <none>    92a53c002b32   2 hours ago         912MB
<none>                 <none>    7f212e71dec7   2 hours ago         142MB
<none>                 <none>    88e0a3f3f055   2 hours ago         1.64GB
<none>                 <none>    65ba6ae31f71   2 hours ago         1.63GB
<none>                 <none>    49c16d3af2ef   2 hours ago         1.64GB
node                   latest    7828fdf71577   11 days ago         1.1GB
node                   14        1d12470fa662   6 months ago        912MB
nginx                  1.21      0e901e68141f   16 months ago       142MB
sharumss@sharumss-virtual-machine:~$ sudo docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
sharumss@sharumss-virtual-machine:~$ sudo docker tag 942d6ed2374f sharamss/my-first-react-app:1.0
sharumss@sharumss-virtual-machine:~$ sudo docker tag 942d6ed2374f sharamss/my-nginx:1.0
sharumss@sharumss-virtual-machine:~$ sudo docker tag f15638414a97 sharamss/my-first-react-app:1.0
sharumss@sharumss-virtual-machine:~$ sudo docker tag 2ece607e81cd sharamss/my-first-node-server:1.0
sharumss@sharumss-virtual-machine:~$
```

Then the created images were pushed to the Docker Hub which is a remote repository for stpring the images using command ‘docker push:tag’ command.

**Screenshot:**



sharamss / **my-nginx**
Contains: Image | Last pushed: a minute ago
Inactive ☆ 0 ⬇ 0 Public

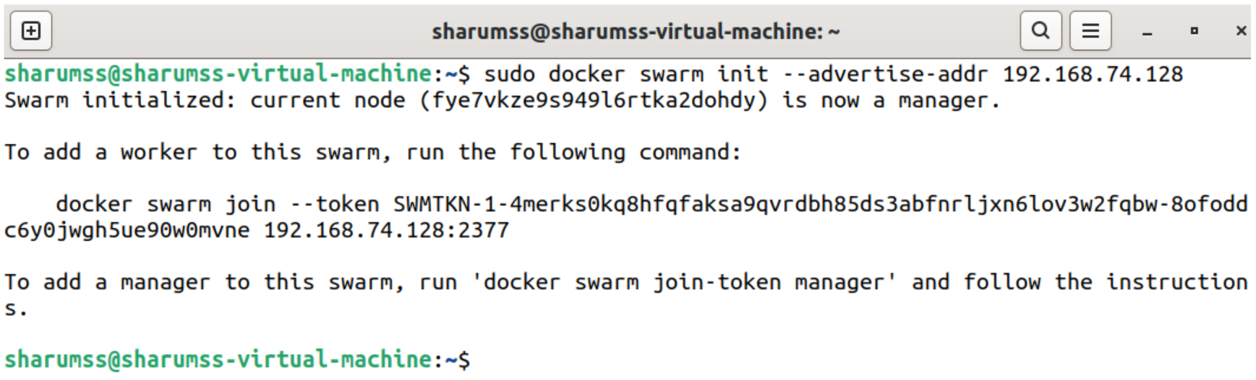
sharamss / **my-first-react-app**
Contains: Image | Last pushed: 3 minutes ago
Inactive ☆ 0 ⬇ 0 Public

sharamss / **my-first-node-server**
Contains: Image | Last pushed: 5 minutes ago
Inactive ☆ 0 ⬇ 0 Public

## Number 5: installing docker swarm and managing the clusters of containers.

A machine was initialized as a node manager and join token was generated.

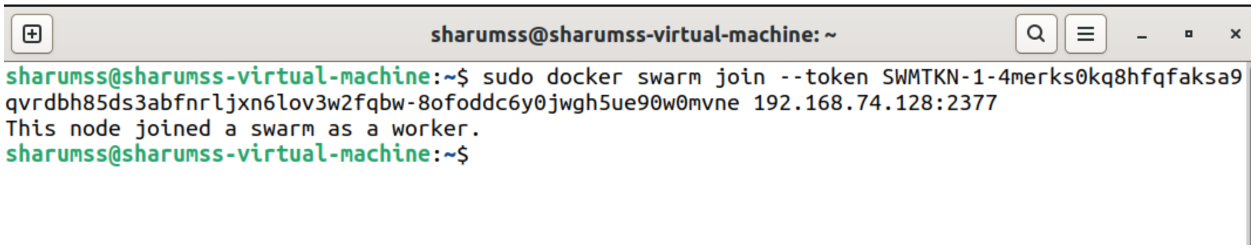
**Screenshot:**



```
sharumss@sharumss-virtual-machine:~$ sudo docker swarm init --advertise-addr 192.168.74.128
Swarm initialized: current node (fye7vkze9s949l6rtka2dohdy) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4merks0kq8hfqfaksa9qvrdbh85ds3abfnrljxn6lov3w2fqbw-8ofodd
c6y0jwgh5ue90w0mvne 192.168.74.128:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instruction
s.

sharumss@sharumss-virtual-machine:~$
```

Then on another machine, the generated token was inputted to make it join the node as a worker node.

**Screenshot:**

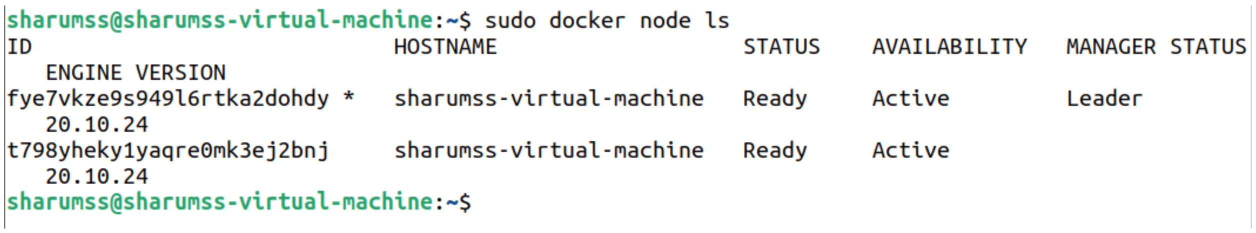


sharumss@sharumss-virtual-machine: ~

```
sharumss@sharumss-virtual-machine:~$ sudo docker swarm join --token SWMTKN-1-4merks0kq8hfqfaksa9
qvrdbh85ds3abfnrljxn6lov3w2fqbw-8ofoddc6y0jwgh5ue90w0mvne 192.168.74.128:2377
This node joined a swarm as a worker.
sharumss@sharumss-virtual-machine:~$
```

On the manager node, the joining of worker node in the swarm was verified.

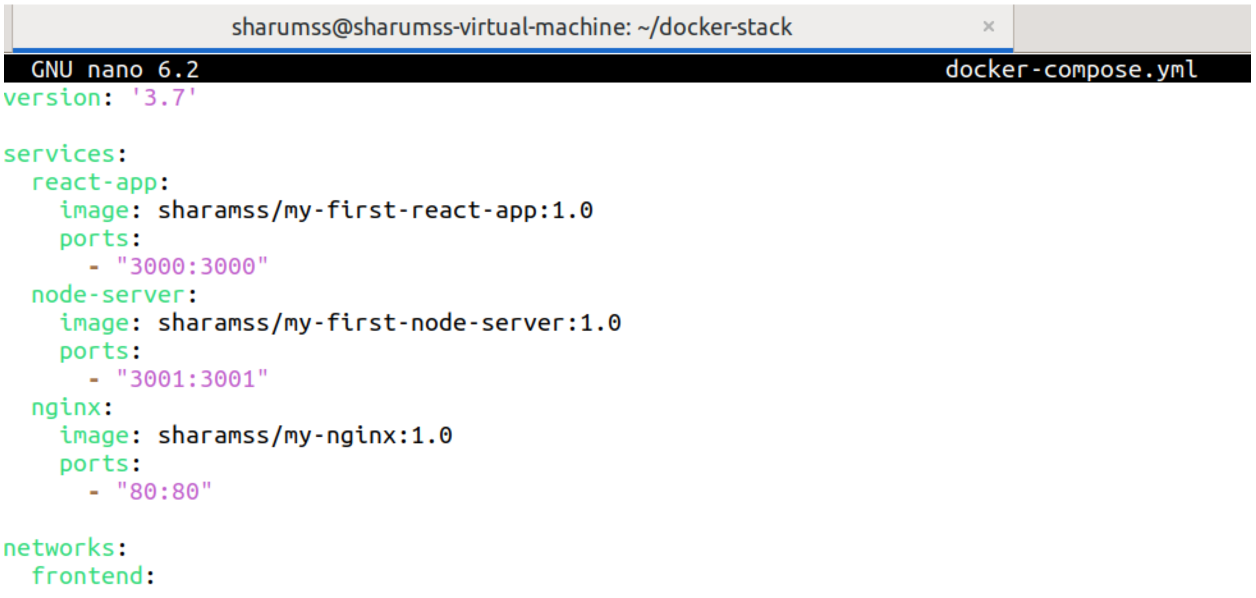
**Screenshot:**



```
sharumss@sharumss-virtual-machine:~$ sudo docker node ls
ID                            HOSTNAME                    STATUS    AVAILABILITY   MANAGER STATUS
   ENGINE VERSION
fye7vkze9s949l6rtka2dohdy *   sharumss-virtual-machine    Ready     Active         Leader
   20.10.24
t798yheky1yaqre0mk3ej2bnj     sharumss-virtual-machine    Ready     Active
   20.10.24
sharumss@sharumss-virtual-machine:~$
```

Then a docker stack file was created to deploy the created containers in the swarm.

**Screenshot:**

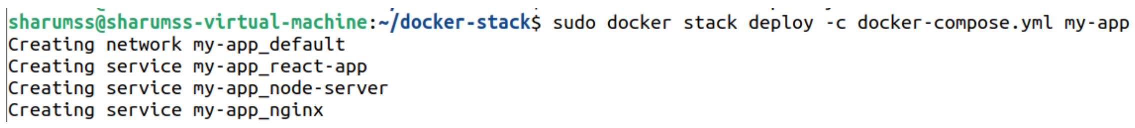


sharumss@sharumss-virtual-machine: ~/docker-stack

```
  GNU nano 6.2                                               docker-compose.yml
version: '3.7'

services:
  react-app:
    image: sharamss/my-first-react-app:1.0
    ports:
      - "3000:3000"
  node-server:
    image: sharamss/my-first-node-server:1.0
    ports:
      - "3001:3001"
  nginx:
    image: sharamss/my-nginx:1.0
    ports:
      - "80:80"

networks:
  frontend:
```

After creating the docker stack file, it was finally deployed in the swarm.

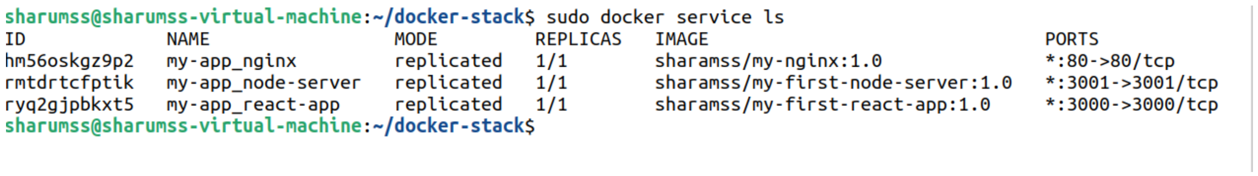
**Screenshot:**



```
sharumss@sharumss-virtual-machine:~/docker-stack$ sudo docker stack deploy -c docker-compose.yml my-app
Creating network my-app_default
Creating service my-app_react-app
Creating service my-app_node-server
Creating service my-app_nginx
```

We can then view the deployed services using 'docker service ls'.

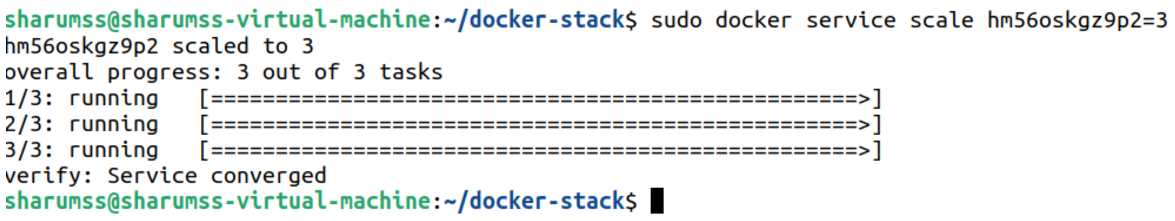
**Screenshot:**



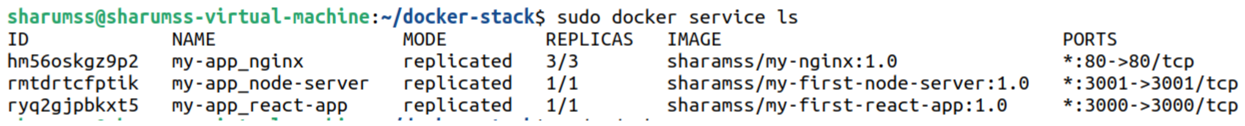
```
sharumss@sharumss-virtual-machine:~/docker-stack$ sudo docker service ls
ID             NAME                 MODE         REPLICAS   IMAGE                                PORTS
hm56oskgz9p2   my-app_nginx         replicated   1/1        sharamss/my-nginx:1.0                *:80->80/tcp
rmtdrtcfptik   my-app_node-server   replicated   1/1        sharamss/my-first-node-server:1.0    *:3001->3001/tcp
ryq2gjpbkxt5   my-app_react-app     replicated   1/1        sharamss/my-first-react-app:1.0      *:3000->3000/tcp
sharumss@sharumss-virtual-machine:~/docker-stack$
```

Then, the nginx service was scaled by creating 3 replicas.

**Screenshot:**



```
sharumss@sharumss-virtual-machine:~/docker-stack$ sudo docker service scale hm56oskgz9p2=3
hm56oskgz9p2 scaled to 3
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
sharumss@sharumss-virtual-machine:~/docker-stack$
```

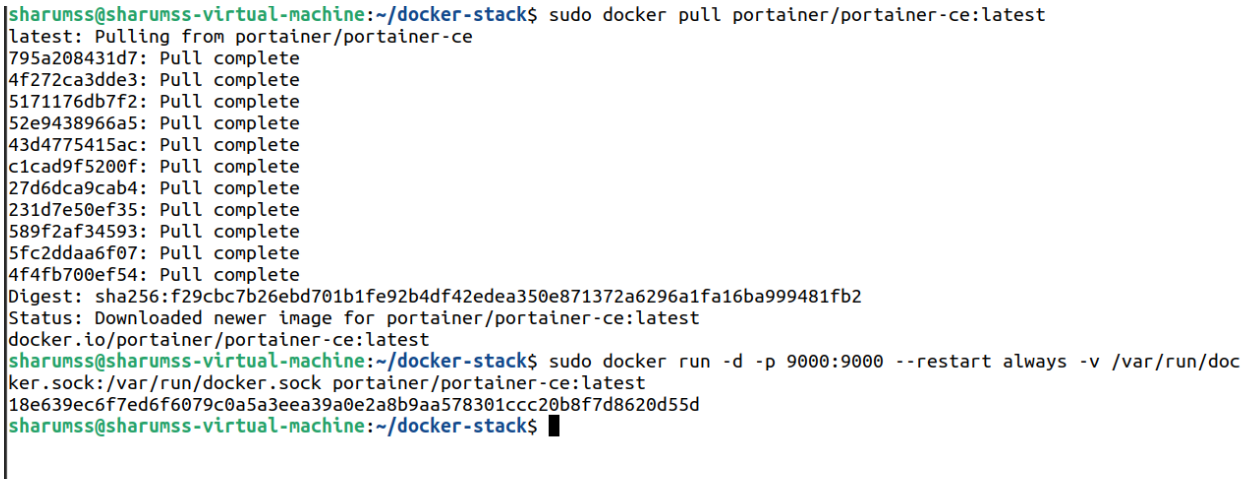


```
sharumss@sharumss-virtual-machine:~/docker-stack$ sudo docker service ls
ID             NAME                 MODE         REPLICAS   IMAGE                                PORTS
hm56oskgz9p2   my-app_nginx         replicated   3/3        sharamss/my-nginx:1.0                *:80->80/tcp
rmtdrtcfptik   my-app_node-server   replicated   1/1        sharamss/my-first-node-server:1.0    *:3001->3001/tcp
ryq2gjpbkxt5   my-app_react-app     replicated   1/1        sharamss/my-first-react-app:1.0      *:3000->3000/tcp
```

## Number 6: installing the poratiner and managing the docker from GUI

The image of docker portainer was pulled from the Docker Hub. And after pulling the image, the container was run exposing port 9000.

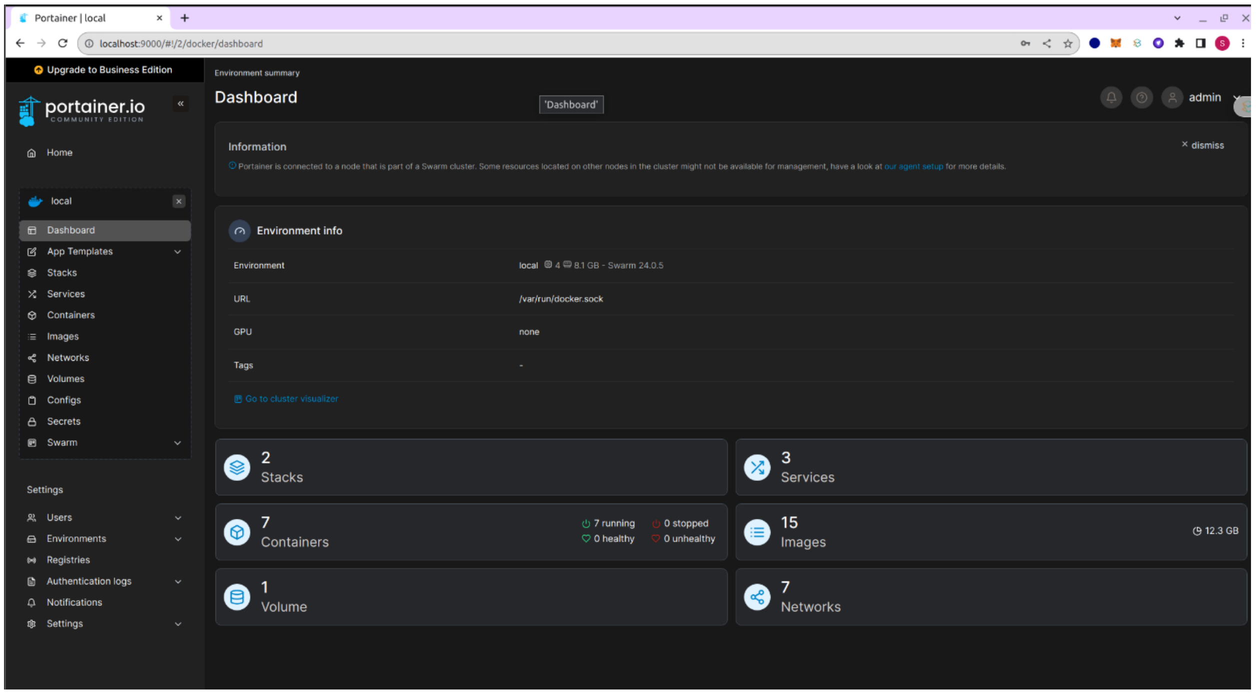
**Screenshot:**



```
sharumss@sharumss-virtual-machine:~/docker-stack$ sudo docker pull portainer/portainer-ce:latest
latest: Pulling from portainer/portainer-ce
795a208431d7: Pull complete
4f272ca3dde3: Pull complete
5171176db7f2: Pull complete
52e9438966a5: Pull complete
43d4775415ac: Pull complete
c1cad9f5200f: Pull complete
27d6dca9cab4: Pull complete
231d7e50ef35: Pull complete
589f2af34593: Pull complete
5fc2ddaa6f07: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:f29cbc7b26ebd701b1fe92b4df42edea350e871372a6296a1fa16ba999481fb2
Status: Downloaded newer image for portainer/portainer-ce:latest
docker.io/portainer/portainer-ce:latest
sharumss@sharumss-virtual-machine:~/docker-stack$ sudo docker run -d -p 9000:9000 --restart always -v /var/run/doc
ker.sock:/var/run/docker.sock portainer/portainer-ce:latest
18e639ec6f7ed6f6079c0a5a3eea39a0e2a8b9aa578301ccc20b8f7d8620d55d
sharumss@sharumss-virtual-machine:~/docker-stack$
```

On navigating, [http://localhost:9000](http://localhost:9000) on local machine, we could access the docker GUI and manage docker images, containers, volumes, networks, etc.

**Screenshot:**



Information

Portainer is connected to a node that is part of a Swarm cluster. Some resources located on other nodes in the cluster might not be available for management, have a look at our agent setup for more details.

Environment info

| | |
|---|---|
| Environment | local 4 8.1 GB - Swarm 24.0.5 |
| URL | /var/run/docker.sock |
| GPU | none |
| Tags | - |

Go to cluster visualizer

2 Stacks
3 Services
7 Containers — 7 running, 0 stopped, 0 healthy, 0 unhealthy
15 Images — 12.3 GB
1 Volume
7 Networks