

Challenge-1:

Please find below the detailed steps to host a static html page on the nginx web server:

1. Go to the following link and download the latest version of Docker desktop for your OS:
<https://www.docker.com/products/docker-desktop/>
2. Follow the instructions in the docker installer and complete the installation with the default configurations provided.
3. Run the docker desktop application and verify that it is installed correctly by opening a command prompt and typing the following command:

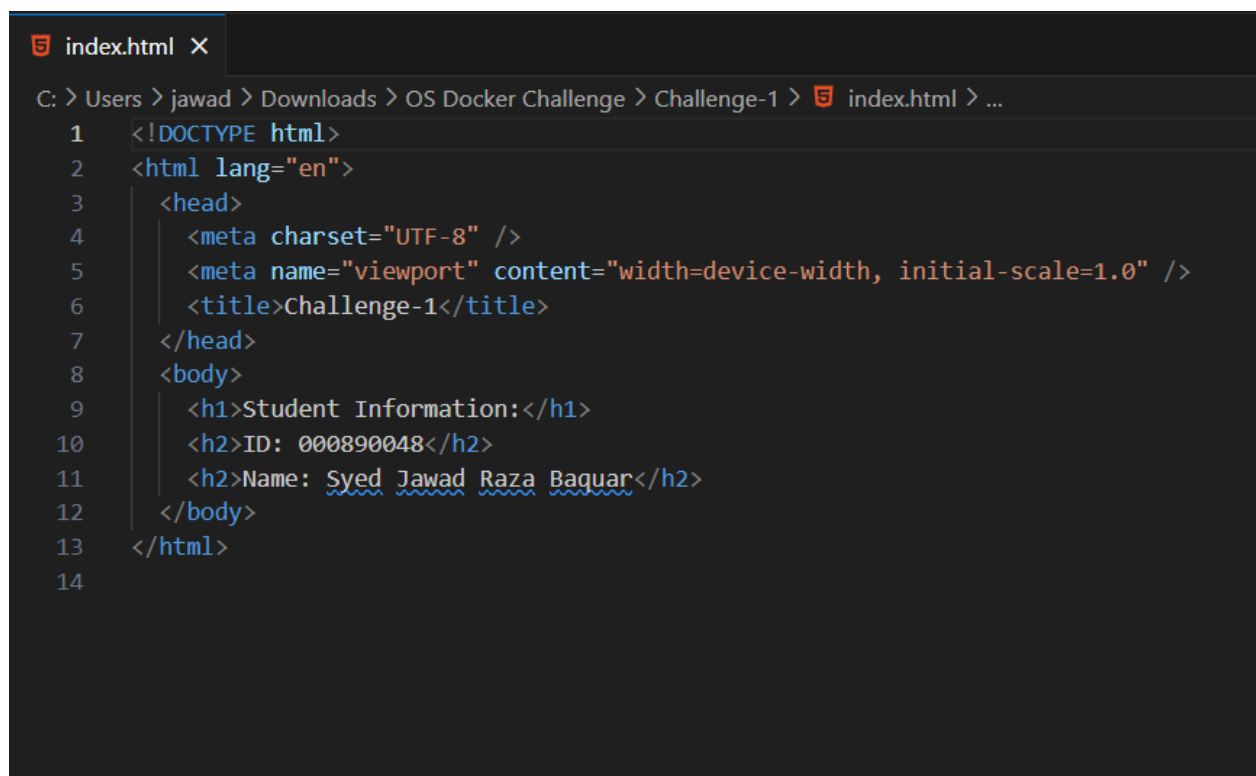
`docker --version`

```
C:\Users\jawad>docker --version
Docker version 25.0.3, build 4debf41

C:\Users\jawad>|
```

Seeing the docker version as output verifies that our installation was successful.

4. Create a file named 'index.html' with the following content:

A screenshot of a code editor window with a dark theme. The title bar shows 'index.html' with a close button. The breadcrumb path is 'C: > Users > jawad > Downloads > OS Docker Challenge > Challenge-1 > index.html > ...'. The code is an HTML document with line numbers 1 through 14 on the left. The content includes a DOCTYPE declaration, HTML and head tags with meta information for charset, viewport, and title, and a body with three h2 tags for student information.

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Challenge-1</title>
7   </head>
8   <body>
9     <h1>Student Information:</h1>
10    <h2>ID: 000890048</h2>
11    <h2>Name: Syed Jawad Raza Baquar</h2>
12  </body>
13 </html>
14
```

This is a simple html file that display the student's information such as ID and name.

5. Create a 'Dockerfile' in the same directory as your 'index.html' file with the following content:

```
C: > Users > jawad > Downloads > OS Docker Challenge > Challenge-1 > Dockerfile > FROM
1 FROM nginx:stable-alpine
2
3 COPY index.html /usr/share/nginx/html/index.html
4
5 EXPOSE 80
```

The 'Dockerfile' contains the instructions that are used to build layers for the docker image that will later be hosted in the container.

'FROM' command tells docker to look for the nginx web version in the Docker hub repository. We are using the stable-apline version of the nginx server.

'COPY' command copies our 'index.html' file into the nginx server's directory which the web server uses to store and host static content like html files.

'EXPOSE' command documents the fact that we will host the image inside the container on port 80. We will have to map this port with a port on the host machine that the nginx server will use to listen to any requests.

6. Navigate to the directory that has your 'index.html' and 'Dockerfile' and run the following command to build the docker image:
docker build -t challenge-1 .

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-1>docker build -t challenge-1 .
[+] Building 2.8s (8/8) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile             0.0s
=> => transferring dockerfile: 126B                             0.0s
=> [internal] load metadata for docker.io/library/nginx:stable-alpin 1.4s
=> [auth] library/nginx:pull token for registry-1.docker.io     0.0s
=> [internal] load .dockerignore                                 0.0s
=> => transferring context: 2B                                     0.0s
=> [1/2] FROM docker.io/library/nginx:stable-alpine@sha256:8f62e8ffc 1.2s
=> => resolve docker.io/library/nginx:stable-alpine@sha256:8f62e8ffc 0.0s
=> => sha256:8f62e8ffc22a112ab3aeb56f56b9ea3e2561248 8.70kB / 8.70kB 0.0s
=> => sha256:3c854c8cbf469fda815b8f6183308c07cfa2fbb 3.38MB / 3.38MB 0.2s
=> => sha256:b407bcc8063852cf7b980fa6d83d6caa2c17b2fa4c1 625B / 625B 0.3s
=> => sha256:e7ece7ddc2e44b2af69a1aa282047794bb31287 2.31kB / 2.31kB 0.0s
=> => sha256:249f59e1dec7f7eacbeba4bb9215b8000e4bd 11.42kB / 11.42kB 0.0s
=> => sha256:de5d475193dd13b444c2e58fc772d8a3297e370 1.80MB / 1.80MB 0.3s
=> => extracting sha256:3c854c8cbf469fda815b8f6183308c07cfa2fbb57038 0.1s
=> => sha256:da33b1ad0ac4b49641e40469216939f6488c1d8116b 954B / 954B 0.3s
=> => sha256:a0fbd691d7c1a07fbddeb8b338578f4e199a49e2491e 768B / 768B 0.4s
=> => extracting sha256:de5d475193dd13b444c2e58fc772d8a3297e370eb90e 0.2s
=> => sha256:5e845cc16269ce694cab9e13e1bdc03d82091 11.67MB / 11.67MB 0.8s
=> => sha256:16eaaaf5f1c0db05389b2c1c2d90db8c9154289 1.40kB / 1.40kB 0.5s
=> => extracting sha256:b407bcc8063852cf7b980fa6d83d6caa2c17b2fa4c10 0.0s
=> => extracting sha256:da33b1ad0ac4b49641e40469216939f6488c1d8116b2 0.0s
=> => extracting sha256:a0fbd691d7c1a07fbddeb8b338578f4e199a49e2491ef 0.0s
=> => extracting sha256:16eaaaf5f1c0db05389b2c1c2d90db8c9154289520fa 0.0s
=> => extracting sha256:5e845cc16269ce694cab9e13e1bdc03d82091f380f1b 0.3s
=> [internal] load build context                                0.0s
=> => transferring context: 371B                                     0.0s
=> [2/2] COPY index.html /usr/share/nginx/html/index.html     0.1s
=> => exporting to image                                             0.0s
=> => exporting layers                                              0.0s
=> => writing image sha256:73f44be238d3adca3ffad516267731a2ac924a77c 0.0s
=> => naming to docker.io/library/challenge-1                      0.0s

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

'docker build' is used to build docker images using the instructions provided in the 'Dockerfile'.

'-t challenge-1' tells docker that we want to tag or name our image as challenge-1.

‘.’ specifies the build context. It tells docker to look for the ‘Dockerfile’ in the current directory.

7. Run the following command to run the docker container:
docker run -d -p 8080:80 challenge-1

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

C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-1>docker run -d -p 80
80:80 challenge-1
115cff7580a9783b05b0173b38422ded440add7bb4e9f19a69e1ae5344842e31
C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-1>
```

‘docker run’ command is used to create a docker container and start the container using the provided docker image.

‘-d’ tells docker to run the container in detached mode or in the background. Which means the container keeps running even after we close the command prompt.

‘-p 8080:80’ flag tells docker that the docker container will be published on port 8080 on the host machine which will be mapped to port 80 on the container. Any request on port 8080 will be forwarded to port 80 on the container.

‘challenge-1’ is the name of the image from which the container will be created and run.

8. Navigate to ‘http://localhost:8080/’ in a web browser to see the output shown below:



The screenshot shows a web browser window with the address bar set to 'localhost:8080'. The page content is as follows:

Student Information:

ID: 000890048

Name: Syed Jawad Raza Baquar

9. Run the command ‘docker ps -a’ to list all the services or docker containers we have created so far. See which containers are still in running status. Note down their Container ID. Then run the command ‘docker stop <Container ID>’ to stop that container. Verify that no container are currently running by using the command ‘docker container ls’

```
C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-1>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
115cff7580a9   challenge-1    "/docker-entrypoint..." 6 minutes ago  Up 6 minutes  0.0.0.0:8080->80/tcp      nifty_feynman
b5f21897c163   hello-world    "/hello"                  3 hours ago   Exited (0) 3 hours ago                                xenodochial_brown

C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-1>docker stop 115cff7580a9
115cff7580a9

C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-1>docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-1>
```

We perform the above step to free up any ports that we will need to use in the future.

Challenge-2:

Please find below the step by step instructions to host an api server using node and using nginx as a web server to act as reverse proxy and redirect all the requests it receives to the api-server:

1. Create a simple node server application with the following code:

```
js app.js > ...
1  const express = require("express");
2
3  const app = express();
4
5  const books = [
6    { id: 1, title: "To Kill a Mockingbird", author: "Harper Lee" },
7    { id: 2, title: "The Great Gatsby", author: "F. Scott Fitzgerald" },
8    {
9      id: 3,
10     title: "One Hundred Years of Solitude",
11     author: "Gabriel Garcia Marquez",
12   },
13 ];
14
15 app.get("/api/books", (req, res) => {
16   res.json(books);
17 });
18
19 app.get("/api/books/:id", (req, res) => {
20   const book = books.find((b) => b.id === parseInt(req.params.id));
21   if (!book) {
22     return res
23       .status(404)
24       .json({ message: "The book with the given ID was not found." });
25   }
26   res.json(book);
27 });
28
29 const PORT = process.env.PORT || 3000;
30 app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
31
```

2. In the same directory as our node server, create a 'Dockerfile.api-server' file with the following code:

```

Dockerfile.api-server > ...
1  FROM node:lts-alpine AS api-server
2
3  WORKDIR /app
4
5  COPY . .
6
7  RUN npm install express
8
9  EXPOSE 3000
10
11 CMD ["node", "app.js"]

```

This 'Dockerfile.api-server' will be used to build the image for our api-server and run it in a container.

We use the 'node:lts-alpine' version of the node server to host our APIs.

We set our working directory as /app so that all further commands and actions will be performed in this directory in the container.

We copy all the files in our directory to use while building the image.

Then we install any dependencies needed to run our application.

We document the port that will be used by this image in the container.

Then we provide 'node app.js' command to run our API server once the image has been built.

3. Create a 'nginx.conf' file in the same directory with the following code:

```

nginx.conf
1  events {}
2  http {
3      server {
4          listen 8080;
5          location / {
6              proxy_pass http://api-server:3000/;
7          }
8      }
9  }

```

We are basically configuring our nginx web server to work as reverse proxy. It will listen to network traffic on port 8080 and redirect all the requests it receives to our node api-server running on port 3000.

4. Create a 'docker-compose.yml' file in the same directory with the following code:

```

🐳 docker-compose.yml
1  version: "3.8"
2
3  services:
4    api-server:
5      build:
6        context: .
7        dockerfile: Dockerfile.api-server
8      ports:
9        - "3000:3000"
10
11     reverse-proxy:
12       image: nginx:stable-alpine
13       ports:
14         - "8080:8080"
15       volumes:
16         - ./nginx.conf:/etc/nginx/nginx.conf:ro
17

```

We use docker compose to create and run two services, our api server and our web server as reverse proxy together at the same time. Docker compose can be used to create multiple containers as services and combine them together as one service or container and host it. We use the 'Dockerfile.api-server' file that we created to build and run our api-server service.

We use the 'nginx:stable-alpine' version of the nginx server to create the image for our web server. It will listen on port 8080.

We use volume mount feature to use our 'nginx.conf' file in the current directory as the configuration file for our nginx server and provide the server with read-only permission to our configuration file.

5. Open command prompt and navigate to the directory that has all our files. Run the following command to use docker compose to build and run our services:
 docker-compose up -d

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-2>docker-compose up -d
[+] Running 2/2
  ✓ Container challenge-2-reverse-proxy-1 Started 0.6s
  ✓ Container challenge-2-api-server-1 Started 0.4s

C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-2>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
7988d0ae4e03   nginx:stable-alpine   "/docker-entrypoint..." 15 seconds ago Up 13 seconds 80/tcp, 0.0.0.0:8080->8080/tcp   challenge-2-reverse-proxy-1
5d79522d7ab7   challenge-2-api-server  "docker-entrypoint.s..." 15 seconds ago Up 14 seconds 0.0.0.0:3000->3000/tcp         challenge-2-api-server-1

C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-2>
```

6. We can see that both our services are running. Use the command 'docker ps' to verify that both our containers are indeed up and running.
7. Open a web browser. When we navigate to 'http://localhost:8080/api/books' we see that all the books in our list are displayed.

```
syedjawadrazabaqar/osassign: x localhost:8080/api/books
localhost:8080/api/books
[{"id":1,"title":"To Kill a Mockingbird","author":"Harper Lee"}, {"id":2,"title":"The Great Gatsby","author":"F. Scott Fitzgerald"}, {"id":3,"title":"One Hundred Years of Solitude","author":"Gabriel Garcia Marquez"}]
```

8. When we navigate to 'http://localhost:8080/api/books/<Book ID>' we can see that the book with the provided Book ID is displayed.



9. Run the command 'docker-compose down' to stop all the services and free up the ports for future use.

```
C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-2>docker-compose down
[+] Running 3/3
  ✓Container challenge-2-reverse-proxy-1 Removed      0.5s
  ✓Container challenge-2-api-server-1    Removed     10.3s
  ✓Network challenge-2_default           Removed      0.2s

C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-2>docker-compose ps
NAME                IMAGE              COMMAND                  SERVICE    CREATED   STATUS    PORTS
C:\Users\jawad\Downloads\OS Docker Challenge\Challenge-2>
```


References:

<https://www.freecodecamp.org/news/the-docker-handbook/>