Seif El-Din Mohamed Aboelhassan Hendawy
Lab 2

What is the difference between:
1- CMD & ENTRYPOINT

Feature	CMD	ENTRYPOINT
Definition	Defines the default command or parameters that will be executed when the container starts.	Specifies the executable that will run when the container starts, and it also can take additional parameters.
Usage	Typically used to provide default arguments for the main command or process that the container will run.	Often used to define the main command or application that the container will run.
Overriding	The command specified using CMD can be overridden by providing a command when running the container.	The command specified using ENTRYPOINT can be overridden by providing arguments when running the container.
Multiple Use	The last CMD instruction in the Dockerfile will be used if multiple CMD instructions are defined.	Only one ENTRYPOINT instruction can be defined in a Dockerfile. If multiple are defined, only the last one will be used.
Shell Form	If the CMD instruction is specified in shell form (e.g., CMD command param1), it will be executed using a shell.	If the ENTRYPOINT instruction is specified in shell form (e.g., ENTRYPOINT command param1), it will be executed using a shell.

JSON Form	If the CMD instruction is specified in JSON form (e.g., CMD ["command", "param1"]), it will be executed directly without a shell.	If the ENTRYPOINT instruction is specified in JSON form (e.g., ENTRYPOINT ["command", "param1"]), it will be executed directly without a shell.
Combining	CMD can be combined with ENTRYPOINT to provide default arguments that can be overridden.	ENTRYPOINT can be combined with CMD to provide default command or arguments that can be overridden.
Recommendation	Use CMD for providing default command arguments. It's common to combine CMD with ENTRYPOINT for a more flexible configuration.	Use ENTRYPOINT to define the primary executable of the container. Combine it with CMD to provide default arguments that can be overridden.

2- COPY & ADD

Feature	COPY	ADD
Usage	Used to copy files and directories from the host system to the container's filesystem.	Similar to COPY, it's used to copy files and directories from the host to the container. It can also fetch remote URLs and extract compressed archives.
Syntax	COPY <src> <dest>	ADD <src> <dest>
Src Path	Specifies the path to the file or directory on the host machine.	Specifies the path to the file or directory on the host machine.
Dest Path	Specifies the path where the file or directory should be copied within the container.	Specifies the path where the file or directory should be copied within the container.

URL Support	Doesn't support fetching remote URLs.	Supports fetching remote URLs and copying them to the container.
Compression	Doesn't automatically extract compressed archives.	Automatically extracts compressed archives (.tar, .tar.gz, .tgz, .tar.bz2, .tbz, .tar.xz, .txz, .tar.Z) after copying.
Ownership	Copies files with the host's ownership and permissions.	Copies files with the host's ownership and permissions.
Best Practices	Use COPY when you only need to copy files from the host to the container.	Use COPY for simple file copying and to maintain clarity in the Dockerfile. Reserve ADD for cases where URL fetching and automatic decompression are required.
Security	More explicit and less likely to introduce unintended behavior.	Can be less predictable due to the automatic URL fetching and archive extraction. Be cautious with external URLs.
Recommendation	Use COPY for most cases of file copying, especially when clarity and control are important.	Use ADD when you need the additional features such as URL fetching and automatic archive extraction. Be aware of potential security implications.

Problem 1:

Run the image Nginx

Add html static files to the container and make sure they are accessible

```
seif@alienware-17-r5:~$ docker run -d --name my-nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
648e0aadf75a: Pull complete
262696647b70: Pull complete
e66d0270d23f: Pull complete
55ac49bd649c: Pull complete
cbf42f5a00d2: Pull complete
8015f365966b: Pull complete
4cadff8bc2aa: Pull complete
Digest: sha256:67f9a4f10d147a6e04629340e6493c9703300ca23a2f7f3aa56fe615d75d31ca
Status: Downloaded newer image for nginx:latest
79db74e86f43ab8b0a507badf8c3a50b54d94c5e3646d486e989e0f4b025976e
seif@alienware-17-r5:~$ touch a.html
seif@alienware-17-r5:~$ docker cp a.html my-nginx:/usr/share/nginx/html
Successfully copied 1.54kB to my-nginx:/usr/share/nginx/html
```

```
seif@alienware-17-r5:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                  NAMES
79db74e86f43   nginx          "/docker-entrypoint.…"   14 minutes ago   Up 14 minutes   80/tcp                 my-nginx
b645a75ea498   mysql:latest   "docker-entrypoint.s…"   17 minutes ago   Up 17 minutes   3306/tcp, 33060/tcp    app-database
seif@alienware-17-r5:~$ docker stop 79db74e86f43
79db74e86f43
seif@alienware-17-r5:~$ docker rm 79db74e86f43
79db74e86f43
seif@alienware-17-r5:~$ docker run -d --name my-nginx-new -p 8080:80 nginx
553c1e31b2b2ef8f80fdde4f5e4ae2766213e8f4e977ff07e5e201d7da96e718
seif@alienware-17-r5:~$ curl http://localhost
curl: (7) Failed to connect to localhost port 80 after 0 ms: Connection refused
seif@alienware-17-r5:~$ curl http://localhost:8080/
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
```

Commit the container with image name IMAGE_NAME

```
seif@alienware-17-r5:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                   NAMES
553c1e31b2b2   nginx          "/docker-entrypoint.…"   7 minutes ago    Up 7 minutes    0.0.0.0:8080->80/tcp, :::8080->80/tcp   my-nginx-new
b645a75ea498   mysql:latest   "docker-entrypoint.s…"   25 minutes ago   Up 25 minutes   3306/tcp, 33060/tcp                     app-database
seif@alienware-17-r5:~$ docker commit nginx neww
Error response from daemon: No such container: nginx
seif@alienware-17-r5:~$ docker commit 553c1e31b2b2 neww
sha256:19bb8e75a0f96d40dfebd5385f5cc48c4b62bf05b34c6a54b951f5a0c006e8fe
seif@alienware-17-r5:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                   NAMES
553c1e31b2b2   nginx          "/docker-entrypoint.…"   8 minutes ago    Up 8 minutes    0.0.0.0:8080->80/tcp, :::8080->80/tcp   my-nginx-new
b645a75ea498   mysql:latest   "docker-entrypoint.s…"   26 minutes ago   Up 26 minutes   3306/tcp, 33060/tcp                     app-database
seif@alienware-17-r5:~$ docker commit 553c1e31b2b2 IMAGE_NAME
invalid reference format: repository name must be lowercase
seif@alienware-17-r5:~$ docker commit 553c1e31b2b2 nginx
sha256:5cb37b3417a1fa7b172f1e15a3e0fa06f1ca9ef7223eaf4866cf3d9ec2452ede
seif@alienware-17-r5:~$
```

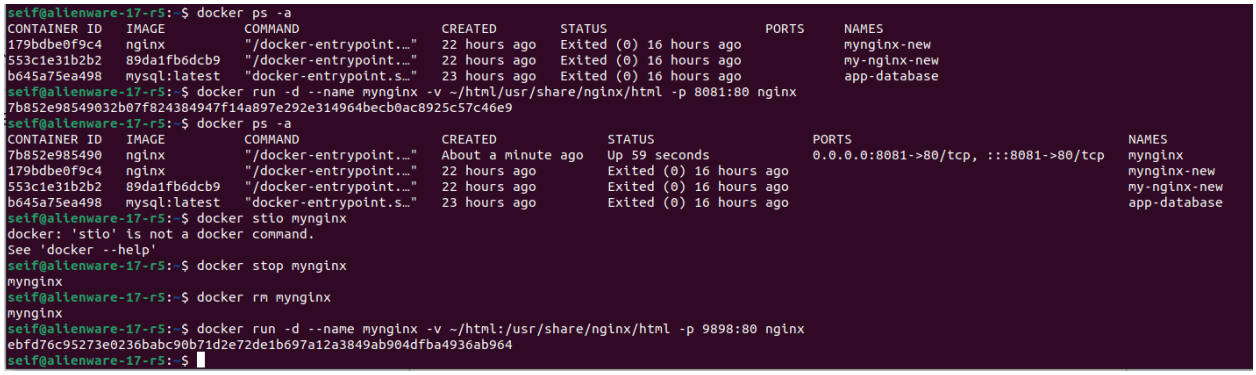
Problem 2:

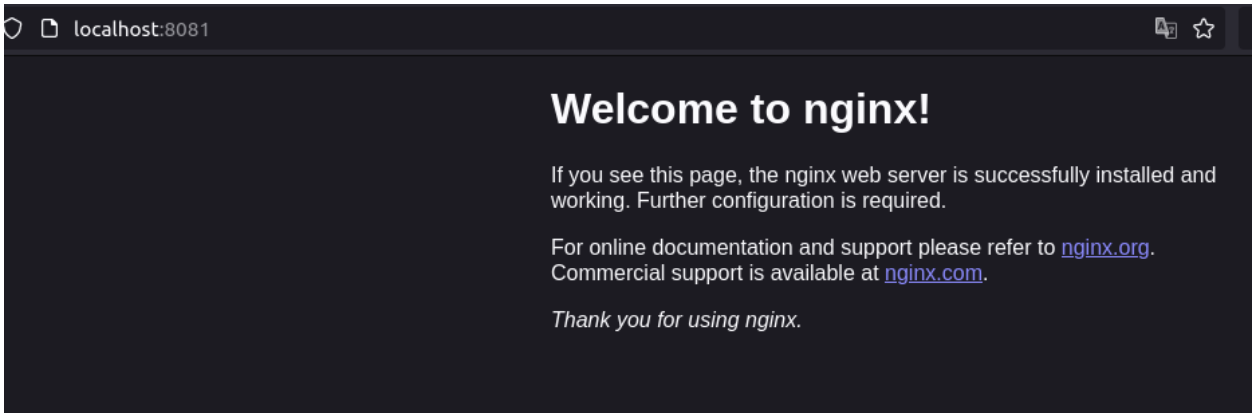
Run a container Nginx with name mynginx and attach a volume for containing static html file

Remove the container

Run a new container with the following:

- Attach the volume that was attached to the previous container
- Map port 80 to port 9898 on you host machine
- Access the html files from your browser





Problem 3:
Run a container nginx with name my-nginx and attach a and attach a volume 2 volumes to the container
Volume1 for containing static html file
Volume2 for containing nginx configuration

```
seif@alienware-17-r5:~$ docker run -d --name my-nginx -v ~/html:/usr/share/nginx/html  -v ~/nginx-config:/etc/nginx nginx
a0e419159aee10824b8f44ccc0c41425721a3867d43aa3896ac03eb8884bc084
```

```
seif@alienware-17-r5:~$ docker run -d -p 8888:80 nginx
18986da7a6136b4e7a673faf1c8cbf56b4f265362f4a6cffddd15594bfdf204a
seif@alienware-17-r5:~$ ip addr show | grep 'inet '
    inet 127.0.0.1/8 scope host lo
    inet 192.168.1.15/24 brd 192.168.1.255 scope global dynamic noprefixroute wlp69s0
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
seif@alienware-17-r5:~$ curl http://192.168.1.15:8888/
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
seif@alienware-17-r5:~$
```

Edit the html content

```
seif@alienware-17-r5:~$ docker exec -it my-nginx bash
root@46c1e75c8624:/# cd /usr/share/nginx/html
root@46c1e75c8624:/usr/share/nginx/html# ls
root@46c1e75c8624:/usr/share/nginx/html# nano index.html
bash: nano: command not found
root@46c1e75c8624:/usr/share/nginx/html# vi index.html
bash: vi: command not found
root@46c1e75c8624:/usr/share/nginx/html# vim index.html
bash: vim: command not found
root@46c1e75c8624:/usr/share/nginx/html# echo "Seif Hendawy DevOps ITI" > index.html
root@46c1e75c8624:/usr/share/nginx/html# cat index.html
Seif Hendawy DevOps ITI
root@46c1e75c8624:/usr/share/nginx/html# exit
exit
```

Remove the container

```
seif@alienware-17-r5:~$ docker stop my-nginx
my-nginx
seif@alienware-17-r5:~$ docker rm my-nginx
my-nginx
```

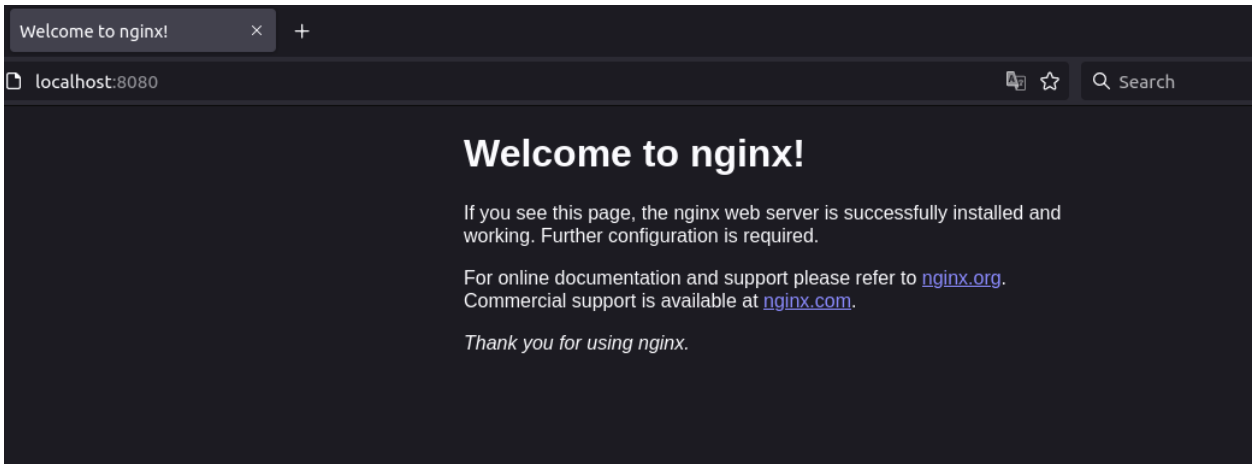
Run a new 2 containers with the following:
Attach the 2 volumes that was attached to the previous container in two different ways (volume mount – bind mount)
Volume Mount
Map port 80 to port 8080 on you host machine
docker run -d --name my-nginx-volc -v ~/html:/usr/share/nginx/html -v ~/nginx-config:/etc/nginx -p 8080:80 nginx

```
seif@alienware-17-r5:~$ docker run -d --name my-nginx-volc -v ~/html:/usr/share/nginx/html  -v ~/nginx-config:/etc/nginx -p 8080:80 nginx
86b5476560e52d34f15b1cc35ea72d5ade230ece36a2cd4f39616a88d37c2013
```

Bind Mount
docker run -d --name my-nginx-bv --mount type=bind,source=/home/seif/html,target=/usr/share/nginx/html --mount type=bind,source=/home/seif/nginx-config,target=/etc/nginx -p 8080:80 nginx

```
seif@alienware-17-r5:~$ docker run -d --name my-nginx-bv --mount type=bind,source=/home/seif/html,target=/usr/share/nginx/html --mount type=bind,source=/home/seif/nginx-config,target=/etc/nginx -p 8080:80
 nginx
90b209747fc7e95595470ad408ae6b8643fea0ef2a4a288947522b3cc0936f5c
seif@alienware-17-r5:~$
```

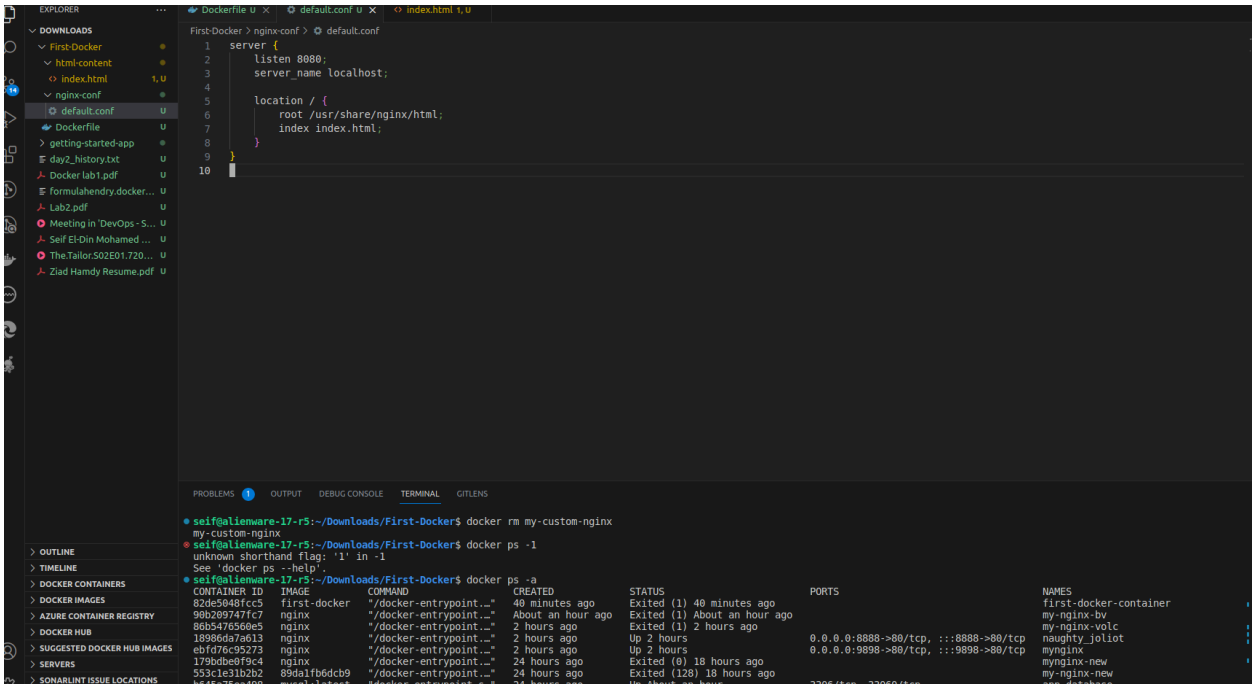
Access the html files from your browser



Problem 4:
Create a dockerfile for nginx image with different html content and different nginx conf that listen to port 8080 instead of port 80 on the container
Create container from the new \image

First-Docker > Dockerfile > EXPOSE

```
1  FROM nginx
2
3  COPY html-content /usr/share/nginx/html
4
5  COPY nginx-conf/default.conf /etc/nginx/conf.d/
6
7  EXPOSE 8080
```



Problem 5:
Create a reactjs simple app
Create a dockerfile to containerize the reactapp
Build the image and test it
(Bonus) create a dockerfile for the same app in smaller size using multi staging

Dockerfile 1

my-app > Dockerfile > ...

```
1   FROM node:14 as builder
2
3   WORKDIR /app
4
5   COPY package*.json ./
6
7   RUN npm install
8
9   COPY . .
10
11  RUN npm run build
12
13  FROM nginx:alpine
14
15  COPY --from=builder /app/build /usr/share/nginx/html
16
17  EXPOSE 80
18
19  CMD ["nginx", "-g", "daemon off;"]
20
```

```
seif@alienware-17-r5:~/Downloads/my-app$ docker build -t my-react-app .
[+] Building 207.3s (14/14) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 266B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/nginx:alpine
 => [internal] load metadata for docker.io/library/node:14
 => CACHED [stage-1 1/2] FROM docker.io/library/nginx:alpine@sha256:647c5c83418c19eef0cddc647b9899326e30815763
 => [builder 1/6] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2
 => [internal] load build context
 => => transferring context: 176.33MB
 => CACHED [builder 2/6] WORKDIR /app
 => [builder 3/6] COPY package*.json ./
 => [builder 4/6] RUN npm install
 => [builder 5/6] COPY . .
 => [builder 6/6] RUN npm run build
 => [stage-1 2/2] COPY --from=builder /app/build /usr/share/nginx/html
 => exporting to image
 => => exporting layers
 => => writing image sha256:873f3a4a91d1bf8993e812b828327912324da28ce6bf3f811b6a345ed4109f5f
 => => naming to docker.io/library/my-react-app
seif@alienware-17-r5:~/Downloads/my-app$ docker run -d -p 8080:80 --name react-app-container my-react-app
1378a3d8d10cde961218edda879a9c392cc166f8abe9180c7b4669e729d6b482
```