# Create ana customize image and upload on docker hub make it public. Using your Customize image build a container.

## Answer

## Steps

1. **Create New Directory**

```
PS C:\Users\ > mkdir myapp


    Directory: C:\Users\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/24/2024  10:34 AM                myapp
```

2. **Create a file called index.html**

```
PS C:\Users\ > cd myapp
PS C:\Users\ \myapp> echo "hello wolrd!"> index.html
PS C:\Users\ \myapp> ls


    Directory: C:\Users\ \myapp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/24/2024  10:34 AM             30 index.html
```

3. **Create file name Dockerfile**
   **A Dockerfile is a text file with instructions to build a Docker Image**

   - When we run a Dockerfile Docker image is created
   - When we run the docker image, containers are created

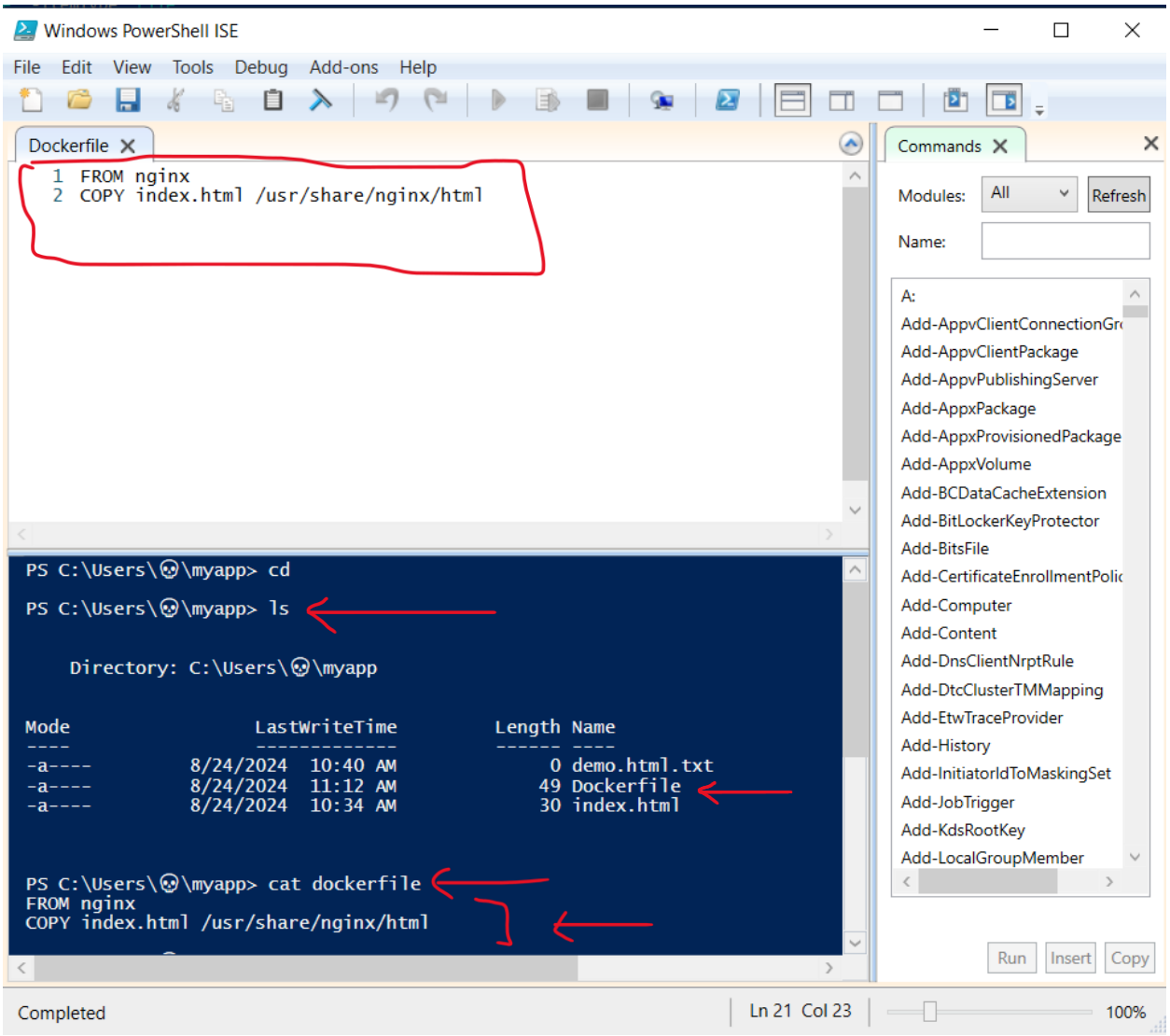
```
PS C:\Users\ \myapp>
PS C:\Users\ \myapp> New-Item -Name "Dockerfile" -ItemType "File"


    Directory: C:\Users\ \myapp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/24/2024  11:08 AM              0 Dockerfile


PS C:\Users\ \myapp> ls


    Directory: C:\Users\ \myapp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/24/2024  10:40 AM              0 demo.html.txt
-a----         8/24/2024  11:08 AM              0 Dockerfile
-a----         8/24/2024  10:34 AM             30 index.html
```

4. **Open the "*Dockerfile*" file in a text editor and add the following lines:**

```
FROM nginx
COPY index.html /usr/share/nginx/html
```

**This Dockerfile defines a new Docker image that**

- Uses the official ***nginx*** images as a base
- Then copy the index.html file to appropriate location in the image




5. **Start and Build Docker image from "*dockerfile*"**

   `docker build -t myapp .`

   **This command builds a new docker image with the tag "*myapp*" using the Dockerfile in the current directory**

```
PS C:\Users\ \myapp> docker build -t myapp .
[+] Building 122.2s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 86B
 => [internal] load metadata for docker.io/library/nginx:latest
 => [auth] library/nginx:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 67B
 => [1/2] FROM docker.io/library/nginx:latest@sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577add
 => => resolve docker.io/library/nginx:latest@sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577add
 => => sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c 7.49kB / 7.49kB
 => => sha256:e4fff0779e6ddd22366469f08626c3ab1884b5cbe1719b26da238c95f247b305 29.13MB / 29.13MB
 => => sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577add 10.27kB / 10.27kB
 => => sha256:5f0574409b3add89581b96c68afe9e9c7b284651c3a974b6e8bac46bf95e6b7f 2.29kB / 2.29kB
 => => sha256:2a0cb278fd9f7737ef5ddc52b4198821dd02e87ed204f74d7e491016b96ebe7f 41.88MB / 41.88MB
 => => sha256:7045d6c32ae2d3dc002f33beb0c1cdd7f69b2663a9720117ac9b82ec28865e30 627B / 627B
 => => sha256:03de31afb03573e0fa679d6777ba3267c2b8ec087cbc0efa46524c1de08f43ec 956B / 956B
 => => sha256:0f17be8dcff2e2c27ee6a33c1bacc582e71f76f855c2d69d510f2a93df897303 394B / 394B
 => => sha256:14b7e5e8f3946da0f9120dab3b0e05ef24a5ca874ba484327db8b3308a92b532 1.21kB / 1.21kB
 => => sha256:23fa5a7b99a685258885918c468ded042b95b5a7c56cee758a689f4f7e5971e0 1.40kB / 1.40kB
 => => extracting sha256:e4fff0779e6ddd22366469f08626c3ab1884b5cbe1719b26da238c95f247b305
 => => extracting sha256:2a0cb278fd9f7737ef5ddc52b4198821dd02e87ed204f74d7e491016b96ebe7f
 => => extracting sha256:7045d6c32ae2d3dc002f33beb0c1cdd7f69b2663a9720117ac9b82ec28865e30
 => => extracting sha256:03de31afb03573e0fa679d6777ba3267c2b8ec087cbc0efa46524c1de08f43ec
 => => extracting sha256:0f17be8dcff2e2c27ee6a33c1bacc582e71f76f855c2d69d510f2a93df897303
 => => extracting sha256:14b7e5e8f3946da0f9120dab3b0e05ef24a5ca874ba484327db8b3308a92b532
 => => extracting sha256:23fa5a7b99a685258885918c468ded042b95b5a7c56cee758a689f4f7e5971e0
 => [2/2] COPY index.html /usr/share/nginx/html
 => exporting to image
 => => exporting layers
 => => writing image sha256:54fb87523b02d798e92375f023dd23f225f564aa2150d505998fe03d2ec3ac0d
 => => naming to docker.io/library/myapp

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ht5qfz9muig5aiv5o77txtvis

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ \myapp>
```

**Successfully Built Image**

```
PS C:\Users\ \myapp> docker images myapp
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
myapp        latest    54fb87523b02   9 minutes ago    188MB
```

## 6. Run Docker Container From the image

```
docker run -p 8080:80 myapp
```

**This tells Docker to run the myapp container and map port 8080 on your local machine to port 80 inside the container**

```
PS C:\Users\ \myapp> docker run -p 8080:80 myapp
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/08/24 06:25:38 [notice] 1#1: using the "epoll" event method
2024/08/24 06:25:38 [notice] 1#1: nginx/1.27.1
2024/08/24 06:25:38 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/08/24 06:25:38 [notice] 1#1: OS: Linux 5.15.153.1-microsoft-standard-WSL2
2024/08/24 06:25:38 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/08/24 06:25:38 [notice] 1#1: start worker processes
2024/08/24 06:25:38 [notice] 1#1: start worker process 29
2024/08/24 06:25:38 [notice] 1#1: start worker process 30
2024/08/24 06:25:38 [notice] 1#1: start worker process 31
2024/08/24 06:25:38 [notice] 1#1: start worker process 32
2024/08/24 06:26:33 [notice] 1#1: signal 2 (SIGINT) received, exiting
2024/08/24 06:26:33 [notice] 29#29: exiting
2024/08/24 06:26:33 [notice] 31#31: exiting
2024/08/24 06:26:33 [notice] 30#30: exiting
2024/08/24 06:26:33 [notice] 29#29: exit
2024/08/24 06:26:33 [notice] 30#30: exit
2024/08/24 06:26:33 [notice] 31#31: exit
2024/08/24 06:26:33 [notice] 32#32: exiting
2024/08/24 06:26:33 [notice] 32#32: exit
2024/08/24 06:26:33 [notice] 1#1: signal 14 (SIGALRM) received
2024/08/24 06:26:33 [notice] 1#1: signal 17 (SIGCHLD) received from 30
2024/08/24 06:26:33 [notice] 1#1: worker process 30 exited with code 0
2024/08/24 06:26:33 [notice] 1#1: signal 29 (SIGIO) received
2024/08/24 06:26:33 [notice] 1#1: signal 17 (SIGCHLD) received from 31
2024/08/24 06:26:33 [notice] 1#1: worker process 31 exited with code 0
2024/08/24 06:26:33 [notice] 1#1: signal 29 (SIGIO) received
2024/08/24 06:26:33 [notice] 1#1: signal 17 (SIGCHLD) received from 29
2024/08/24 06:26:33 [notice] 1#1: worker process 29 exited with code 0
2024/08/24 06:26:33 [notice] 1#1: signal 29 (SIGIO) received
2024/08/24 06:26:33 [notice] 1#1: signal 17 (SIGCHLD) received from 32
2024/08/24 06:26:33 [notice] 1#1: worker process 32 exited with code 0
2024/08/24 06:26:33 [notice] 1#1: exit
```

**This container is Stop by pressing  Ctl+C to run this in detach mode**

```
docker run – d -p 8080:80 myapp
```

```
PS C:\Users\ \myapp>  docker run -d -p 8080:80 myapp
08dc7dc4c43a20c4a067080fd5ea321482c449630d159c446476096dff15030b
```

**Now Next Step is to Check if the Container is Runnning**

```
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS       PORTS                   NAMES
0a8dc317460a   myapp     "/docker-entrypoint.…"   2 hours ago   Up 2 hours   0.0.0.0:8080->80/tcp    ecstatic_lewin
```

**To check look at the status**

7. **Creates a new tag**

```
docker tag myapp safdarnagrish/myapp
```

The command `**docker tag myapp safdarnagrish/myapp**` creates a new tag for the existing Docker image `myapp`, naming it `**safdarnagrish/myapp**`, which is necessary for pushing the image to your Docker Hub repository under your username `**safdarnagrish**`.

8. **Push the Image**

```
docker push safdarnagrish/myapp
```

The command docker push **safdarnagrish/myapp** uploads the Docker image tagged as safdarnagrish/myapp from your local machine to your Docker Hub repository under my account (**safdarnagrish**).

Screenshot Show this image is push to the repository

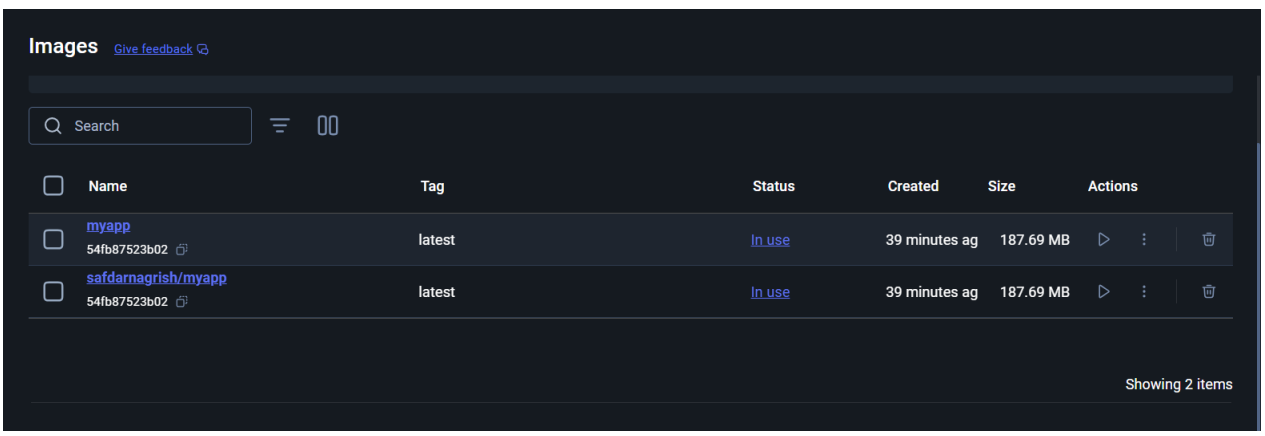
```
PS C:\Users\ \myapp> docker tag myapp safdarnagrish/myapp
PS C:\Users\ \myapp> docker push safdarnagrish/myapp
Using default tag: latest
The push refers to repository [docker.io/safdarnagrish/myapp]
01fee8f02148: Pushing [==================================================>]  4.096kB
5f0272c6e96d: Preparing
f4f00eaedec7: Preparing
55e54df86207: Preparing
ec1a2ca4ac87: Preparing
8b87c0c66524: Waiting
72db5db515fd: Waiting
9853575bc4f9: Waiting
```

Again Checked

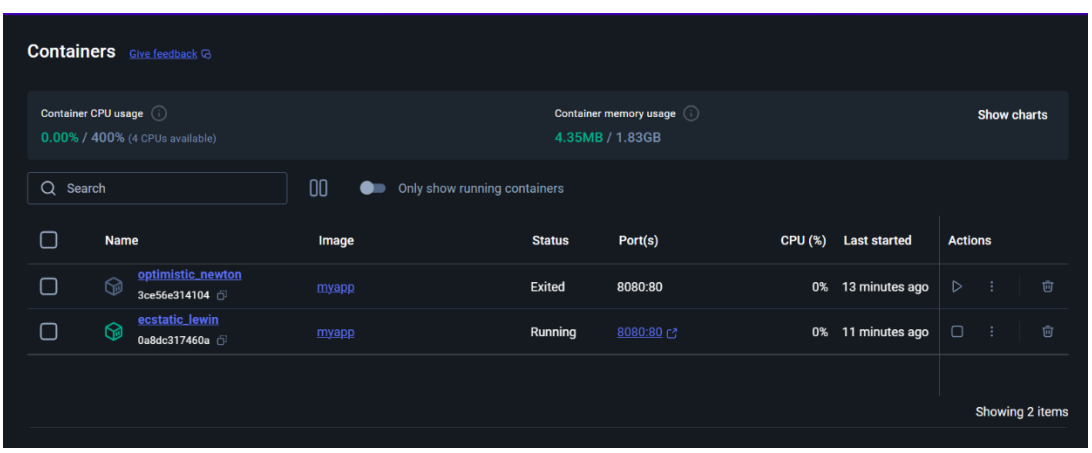
```
PS C:\Users\[redacted] \myapp> docker push safdarnagrish/myapp
Using default tag: latest
The push refers to repository [docker.io/safdarnagrish/myapp]
01fee8f02148: Layer already exists
5f0272c6e96d: Layer already exists
f4f00eaedec7: Layer already exists
55e54df86207: Layer already exists
ec1a2ca4ac87: Layer already exists
8b87c0c66524: Layer already exists
72db5db515fd: Layer already exists
9853575bc4f9: Layer already exists
latest: digest: sha256:13b6b04a212120907118bab4168b6875e460d88e8e58dfe156d9f17e786f5b60 size: 1985
PS C:\Users\[redacted] \myapp>
```

Now it is successfully pushed

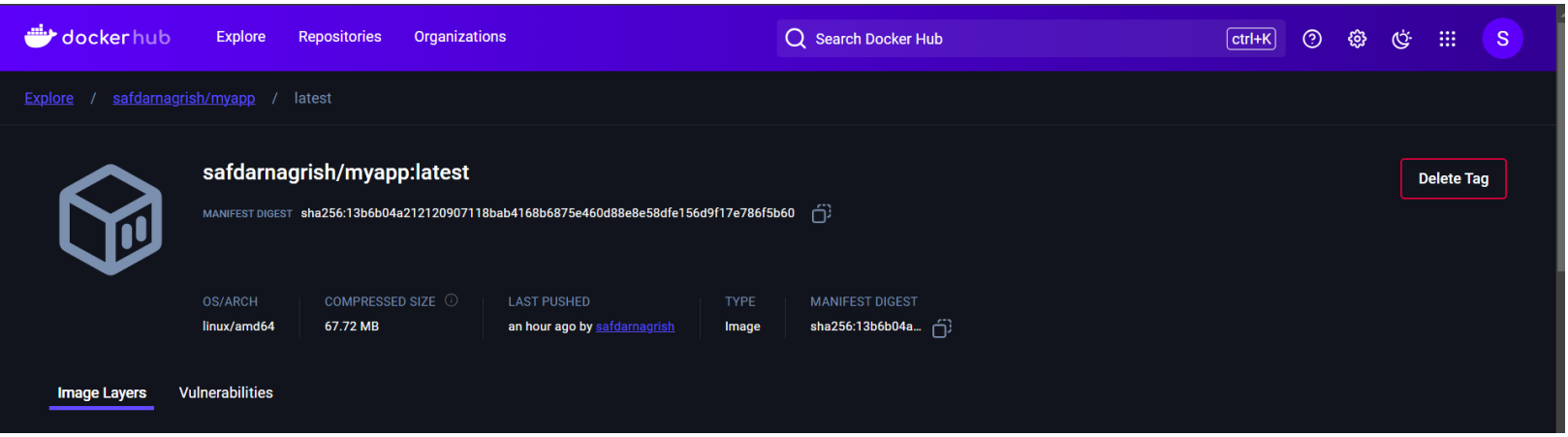
## 9. Check Images on Docker Desktop




## Check Containers



**10. Check Docker Hub**




**Here the is visible to public on Docker Hub**

# Thank You