

1) Start a new project called practice1. You're going to create an API. Put it inside a docker

Structure

```
|— app
|   |— main.py
|   |— requirements.txt
|— Dockerfile
```

Terminal:

1. **mkdir practice1**

cd practice1

Creating practice1 as the main directory helps keep all related files together.

2. **mkdir app** → creates a folder to contain our FastAPI app. The app folder acts like a container, making it easier to tell apart from Docker-related files.

3. **touch app/main.py** → This code sets up a basic FastAPI instance so that my app can respond to web requests.

```
from fastapi import FastAPI
app = FastAPI()
```

4. **Create a requirements.txt file inside the app.** Inside the .txt put

```
fastapi
uvicorn
```

The requirements.txt lists the Python libraries the app depends on. Here, I've added fastapi to run the app and uvicorn to serve it. This file is essential for Docker, as it enables us to install these dependencies inside the container.

5. **cd /SCAV2024/practice1**

touch Dockerfile

```
# Dockerfile
```

```
# Use the official Python image as a base
```

```
FROM python:3.10-slim
```

```
#https://katalon-inc.my.site.com/katalonhelpcenter/s/article/How-to-install-FFMPEG-on-Docker-Image
```

```
# Set the working directory in the container
```

```
WORKDIR /app
```

```
# Copies requirements.txt from your local app folder into the container.
```

```
COPY app/requirements.txt .
```

```
# Installs FastAPI, Uvicorn, and other dependencies listed in requirements.txt.
```

```
RUN pip install --no-cache-dir -r requirements.txt
```

```
# Copy the FastAPI app code to the working directory
```

```
COPY app /app
```

```
# Exposes port 80, which is used to access the application from outside the container.
```

```
EXPOSE 80
```

```
# Starts the FastAPI app with Uvicorn, listening on all network interfaces.
```

```
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
```

Okay now we have to test if it works. How do we do it?

We add a simple endpoint that will respond with a message when accessed in main.py:

```
@app.get("/")
```

```
async def root():
```

```
    return {"message": "Hello World!"}
```

Terminal:

1. docker build -t practice1app .

Here, **docker build** tells Docker to create an image, -t practice1app tags the image with the name practice1app, and the . specifies that Docker should use the current directory (where the Dockerfile is located) to build the image. Docker will follow the steps in the Dockerfile, installing dependencies and copying files into the container.

```
(base) mariaprospeznares@MacBook-Pro-de-Maria practice1 % docker build -t practice1app .
[+] Building 46.8s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 537B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.10-slim
=> [1/6] FROM docker.io/library/python:3.10-slim@sha256:18b42548f6642b997ca6772dfd6ec0e0ecbb1cc683bc1e3b27cfbf768d46d43af
=> => resolve docker.io/library/python:3.10-slim@sha256:18b42548f6642b997ca6772dfd6ec0e0ecbb1cc683bc1e3b27cfbf768d46d43af
=> => sha256:1b8e511c3a5b60fdecce05282a196f23a60d487a8838df4a711b5d83a7d58fc9 1.75kB / 1.75kB
=> => sha256:8df9adb5d39d2c82caf0d8ed092e4e2579d87f2a43b131df6a274bfff5af26b74 5.43kB / 5.43kB
=> => sha256:2d429b9e73a6cf90a5bb85105c8118b30a1b2deedeae3ea9587055ffcb80eb45 29.13MB / 29.13MB
=> => sha256:d447e55d51dbde79ba253e0c58f91c83c1bab832a8ff370b885d07c53b43f490 3.51MB / 3.51MB
=> => sha256:22370d9525db52f6dc1b9c58f06f52eac6ebdf2c3d701a01b12dd38835c7aa2b 15.65MB / 15.65MB
=> => sha256:18b42548f6642b997ca6772dfd6ec0e0ecbb1cc683bc1e3b27cfbf768d46d43af 9.13kB / 9.13kB
=> => sha256:5862c84e1a84d16dcc20b98e377838fb4defa167e4680ec68eef3667490024aa 249B / 249B
=> => extracting sha256:2d429b9e73a6cf90a5bb85105c8118b30a1b2deedeae3ea9587055ffcb80eb45
=> => extracting sha256:d447e55d51dbde79ba253e0c58f91c83c1bab832a8ff370b885d07c53b43f490
=> => extracting sha256:22370d9525db52f6dc1b9c58f06f52eac6ebdf2c3d701a01b12dd38835c7aa2b
=> => extracting sha256:5862c84e1a84d16dcc20b98e377838fb4defa167e4680ec68eef3667490024aa
=> [internal] load build context
=> => transferring context: 790B
=> [2/6] RUN apt-get update && apt-get install -y ffmpeg && rm -rf /var/lib/apt/lists/*
=> [3/6] WORKDIR /app
=> [4/6] COPY app/requirements.txt .
=> [5/6] RUN pip install --no-cache-dir -r requirements.txt
=> [6/6] COPY app /app
=> exporting to image
=> => exporting layers
=> => writing image sha256:048434634426813274458eae820feac98ef5cc4479dc7b2b88ba6254f7a9ecc
=> => naming to docker.io/library/practice1app
```

2. docker run -d -p 80:80 practice1app

Here, **docker run** starts a container from the image. The -d flag runs the container in detached mode, meaning it runs in the background, and -p 80:80 maps port 80 on your local machine to port 80 in the container, allowing you to access the FastAPI app at <http://localhost/>.

3. <http://localhost/>



2) Put ffmpeg inside a Docker

In the **Dockerfile** add:

**RUN apt-get update && **

**apt-get install -y ffmpeg && **

rm -rf /var/lib/apt/lists/*

Here, ffmpeg allows for processing multimedia files. These commands update the package list, install ffmpeg, and then clean up to keep the container size small.

Okay now we have to test if it works. How do we do it?

1. **docker run -it practicelapp /bin/bash**

Runs the Docker Container in Interactive Mode: To test ffmpeg directly.

2. **ffmpeg -version**

```
root@26f25a7dde1c:/app# ffmpeg -version
ffmpeg version 5.1.6-0+deb12u1 Copyright (c) 2000-2024 the FFmpeg developers
built with gcc 12 (Debian 12.2.0-14)
configuration: --prefix=/usr --extra-version=0+deb12u1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --enable-gpl --disable-stripping --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libsbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libglslang --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librabbitmq --enable-librist --enable-librubberband --enable-libshine --enable-lsnaappy --enable-libsoxr --enable-lspspeex --enable-lsrt --enable-lsssh --enable-lsvtav1 --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzimg --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-opengl --enable-openc1 --enable-opengl --enable-sdl2 --disable-sndio --enable-libjxl --enable-pocketsphinx --enable-librsvg --enable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libx264 --enable-libplacebo --enable-librav1e --enable-shared
libavutil 57. 28.100 / 57. 28.100
libavcodec 59. 37.100 / 59. 37.100
libavformat 59. 27.100 / 59. 27.100
libavdevice 59.  7.100 / 59.  7.100
libavfilter  8. 44.100 /  8. 44.100
libswscale  6.  7.100 /  6.  7.100
libswresample 4.  7.100 /  4.  7.100
libpostproc 56.  6.100 / 56.  6.100
root@26f25a7dde1c:/app#
```

Important Information we used to handle our docker container:

To check CONTAINER ID: **docker ps**

To stop container: **docker stop <container_id>**

Docker build and run: **docker build -t practice1app .**

docker run -d -p 80:80 practice1app

Interactive mode: **docker run -it practice1app /bin/bash**