Big Data, organisation and analysis

Building a REST API service part II

Steffen M. Noe, Spring 2024

Recap on building a REST API

Docker

- Docker has a 2-stage build system
- Step 1: Create/Build the Docker image

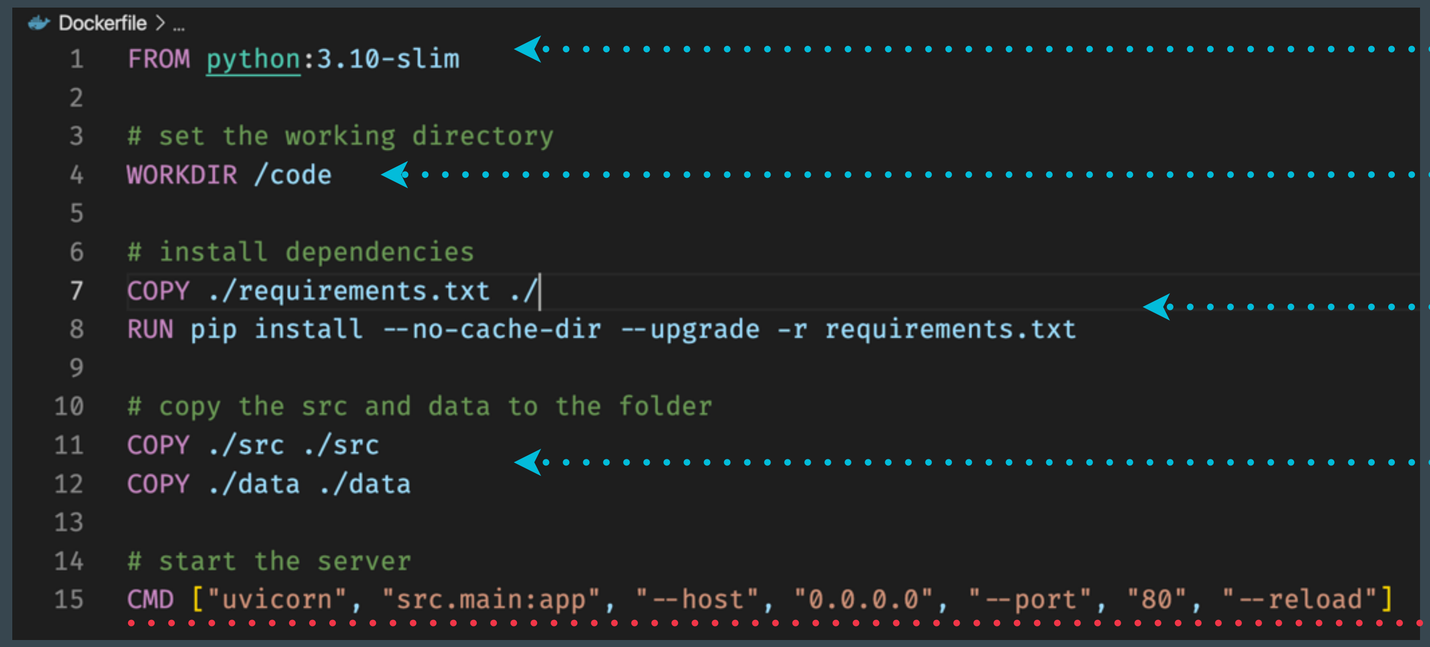
```
docker build -t <tag_image_name> <directory_with_Dockerfile>
```

- Step 2: Create a Docker container

```
docker run –name <container_name> -p <nn:mm> -d <tag_image_name>
```

Docker build

- The build process relies on the Dockerfile



Parent system from Dockerhub

Landing point inside the container

Copy into container and install dependencies

Copy python source code and necessary data into the container

Starting the app using uvicorn inside the container

Docker build

The image has its own filesystem

Dockerfile and source on local computer

```dockerfile
FROM python:3.10-slim

# set the working directory
WORKDIR /code

# install dependencies
COPY ./requirements.txt ./
RUN pip install --no-cache-dir --upgrade -r requirements.txt

# copy the src and data to the folder
COPY ./src ./src
COPY ./data ./data

# start the server
CMD ["uvicorn", "src.main:app", "--host", "0.0.0.0", "--port", "80", "--reload"]
```

```python
# Application that implements a REST API

from fastapi import FastAPI

import pandas as pd
import io

# import a file responder
from fastapi.responses import FileResponse

# import a streaming responder
from fastapi.responses import StreamingResponse

# Link the app with the REST API
app = FastAPI()

# default endpoint, returns a message
@app.get("/")
def read_root():
    return {"Hello": "World"}
```

requirements.txt

```
fastapi
pydantic
uvicorn
pandas
redis
debugpy
```

Container image

Files in the container:

```
/-+-root/
  .
  |
  +-code/+-src/-main.py
  |      +-data/
  |      +-
  .
  +-lib/…
```

Dockerhub hosts parent images

Docker run

Create (instantiate) the container from the image

Container image

Files in the container:

```
/-+-root/
  .
  |
  +-code/+-src/-main.py
  |      +-data/
  |      +-
  .
  +-lib/…
```

```
docker run …
```

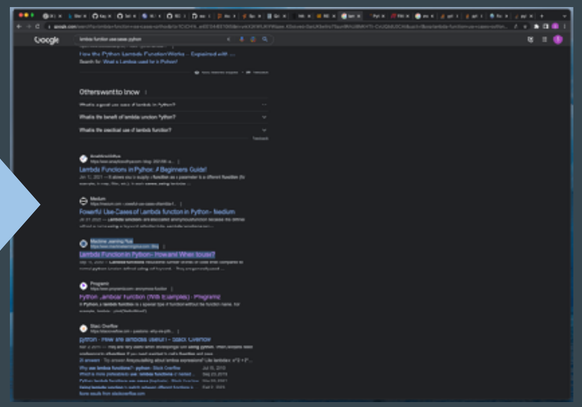
Container

serve REST API
on port 80

http://0.0.0.0/
http://0.0.0.0/data/
http://0.0.0.0/pandas/

this IP because we "simulate" a network on our own computer!

Webbrowser



Docker commands we need

Docker images

```
docker build -t <tag_image_name> <directory_with_Dockerfile>
```

```
docker images
```
show images

```
docker image rm
```
remove images

Docker containers

```
docker run –name <container_name> -p <nn:mm> -d <tag_image_name>
```

```
docker ps
```
show running containers

```
docker ls
```
show running containers

```
docker ls -a
```
show all containers (running and non-running)

```
docker stop <container_name>
```
stop containers

```
docker rm <container_name>
```
remove containers