This is a three-part series. This article will focus more on creating and containerization while in part two I will set up the proxy and finally in part three, we will be deploying our FastApi app to the cloud, by incorporating the deployment of the app into a GitHub Actions pipeline.

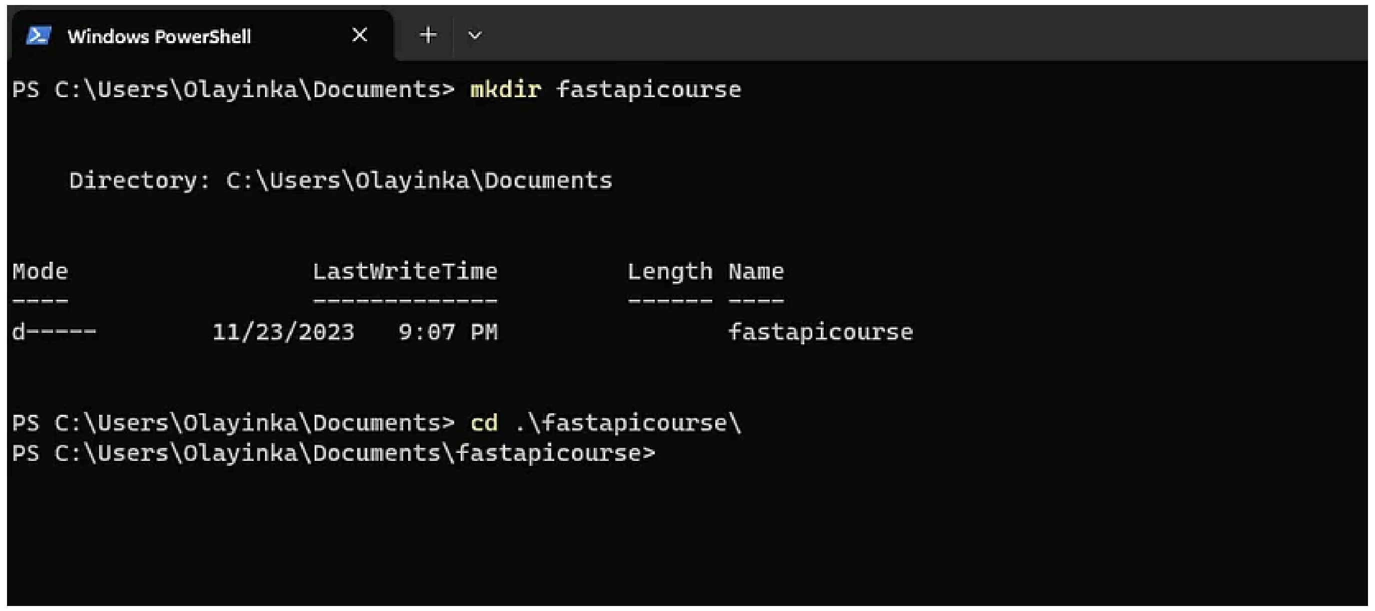
Background:

I recently had a somewhat hectic experience containerizing and deploying a FastApi application and this was partly due to the lack of sufficient resources which would have greatly improved my experience. While this isn't necessarily a replica of the project I worked on, I believe that this tutorial will cover the same core concepts.

Set Up and Installation

To get started with the tutorial, we need to create a project directory

```
mkdir fastapicourse
```



Windows PowerShell

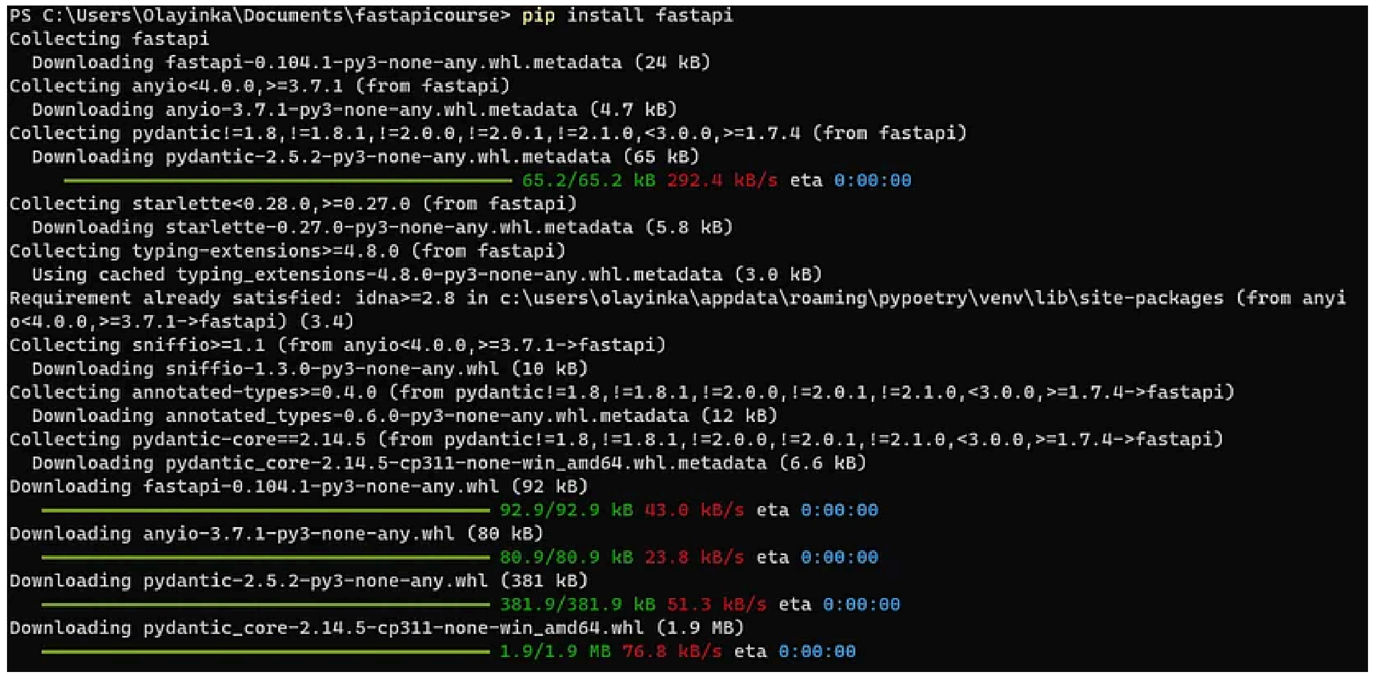
```
PS C:\Users\Olayinka\Documents> mkdir fastapicourse


    Directory: C:\Users\Olayinka\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/23/2023   9:07 PM                fastapicourse


PS C:\Users\Olayinka\Documents> cd .\fastapicourse\
PS C:\Users\Olayinka\Documents\fastapicourse>
```

Change directories into the newly created one:

```
cd fastapicourse
```

Following the default FastApi installation guide, make sure you have Python installed and added to your *Path* before running this command.

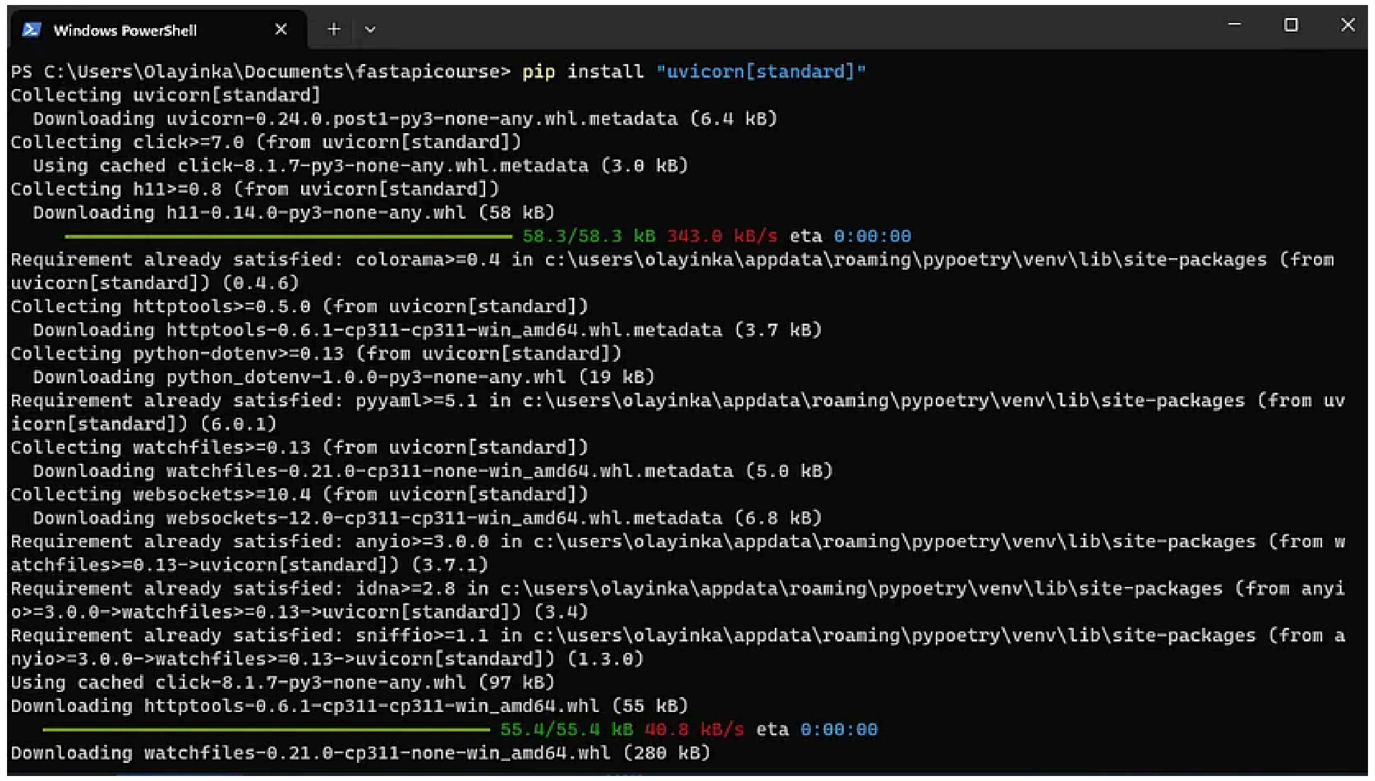
```
pip install fastapi
```



```
PS C:\Users\Olayinka\Documents\fastapicourse> pip install fastapi
Collecting fastapi
  Downloading fastapi-0.104.1-py3-none-any.whl.metadata (24 kB)
Collecting anyio<4.0.0,>=3.7.1 (from fastapi)
  Downloading anyio-3.7.1-py3-none-any.whl.metadata (4.7 kB)
Collecting pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0,>=1.7.4 (from fastapi)
  Downloading pydantic-2.5.2-py3-none-any.whl.metadata (65 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.2/65.2 kB 292.4 kB/s eta 0:00:00
Collecting starlette<0.28.0,>=0.27.0 (from fastapi)
  Downloading starlette-0.27.0-py3-none-any.whl.metadata (5.8 kB)
Collecting typing-extensions>=4.8.0 (from fastapi)
  Using cached typing_extensions-4.8.0-py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: idna>=2.8 in c:\users\olayinka\appdata\roaming\pypoetry\venv\lib\site-packages (from anyi
o<4.0.0,>=3.7.1->fastapi) (3.4)
Collecting sniffio>=1.1 (from anyio<4.0.0,>=3.7.1->fastapi)
  Downloading sniffio-1.3.0-py3-none-any.whl (10 kB)
Collecting annotated-types>=0.4.0 (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0,>=1.7.4->fastapi)
  Downloading annotated_types-0.6.0-py3-none-any.whl.metadata (12 kB)
Collecting pydantic-core==2.14.5 (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0,>=1.7.4->fastapi)
  Downloading pydantic_core-2.14.5-cp311-none-win_amd64.whl.metadata (6.6 kB)
Downloading fastapi-0.104.1-py3-none-any.whl (92 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.9/92.9 kB 43.0 kB/s eta 0:00:00
Downloading anyio-3.7.1-py3-none-any.whl (80 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.9/80.9 kB 23.8 kB/s eta 0:00:00
Downloading pydantic-2.5.2-py3-none-any.whl (381 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 381.9/381.9 kB 51.3 kB/s eta 0:00:00
Downloading pydantic_core-2.14.5-cp311-none-win_amd64.whl (1.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 76.8 kB/s eta 0:00:00
```

We will also need an ASGI server like *uvicorn*. **Why do we need uvicorn?**

```
pip install "uvicorn[standard]"
```



```
PS C:\Users\Olayinka\Documents\fastapicourse> pip install "uvicorn[standard]"
Collecting uvicorn[standard]
  Downloading uvicorn-0.24.0.post1-py3-none-any.whl.metadata (6.4 kB)
Collecting click>=7.0 (from uvicorn[standard])
  Using cached click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting h11>=0.8 (from uvicorn[standard])
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 343.0 kB/s eta 0:00:00
Requirement already satisfied: colorama>=0.4 in c:\users\olayinka\appdata\roaming\pypoetry\venv\lib\site-packages (from
uvicorn[standard]) (0.4.6)
Collecting httptools>=0.5.0 (from uvicorn[standard])
  Downloading httptools-0.6.1-cp311-cp311-win_amd64.whl.metadata (3.7 kB)
Collecting python-dotenv>=0.13 (from uvicorn[standard])
  Downloading python_dotenv-1.0.0-py3-none-any.whl (19 kB)
Requirement already satisfied: pyyaml>=5.1 in c:\users\olayinka\appdata\roaming\pypoetry\venv\lib\site-packages (from uv
icorn[standard]) (6.0.1)
Collecting watchfiles>=0.13 (from uvicorn[standard])
  Downloading watchfiles-0.21.0-cp311-none-win_amd64.whl.metadata (5.0 kB)
Collecting websockets>=10.4 (from uvicorn[standard])
  Downloading websockets-12.0-cp311-cp311-win_amd64.whl.metadata (6.8 kB)
Requirement already satisfied: anyio>=3.0.0 in c:\users\olayinka\appdata\roaming\pypoetry\venv\lib\site-packages (from w
atchfiles>=0.13->uvicorn[standard]) (3.7.1)
Requirement already satisfied: idna>=2.8 in c:\users\olayinka\appdata\roaming\pypoetry\venv\lib\site-packages (from anyi
o>=3.0.0->watchfiles>=0.13->uvicorn[standard]) (3.4)
Requirement already satisfied: sniffio>=1.1 in c:\users\olayinka\appdata\roaming\pypoetry\venv\lib\site-packages (from a
nyio>=3.0.0->watchfiles>=0.13->uvicorn[standard]) (1.3.0)
Using cached click-8.1.7-py3-none-any.whl (97 kB)
Downloading httptools-0.6.1-cp311-cp311-win_amd64.whl (55 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.4/55.4 kB 40.8 kB/s eta 0:00:00
Downloading watchfiles-0.21.0-cp311-none-win_amd64.whl (280 kB)
```

You can then open up the directory using your favorite code editor. I will be using Visual Studio Code for the rest of the tutorial.

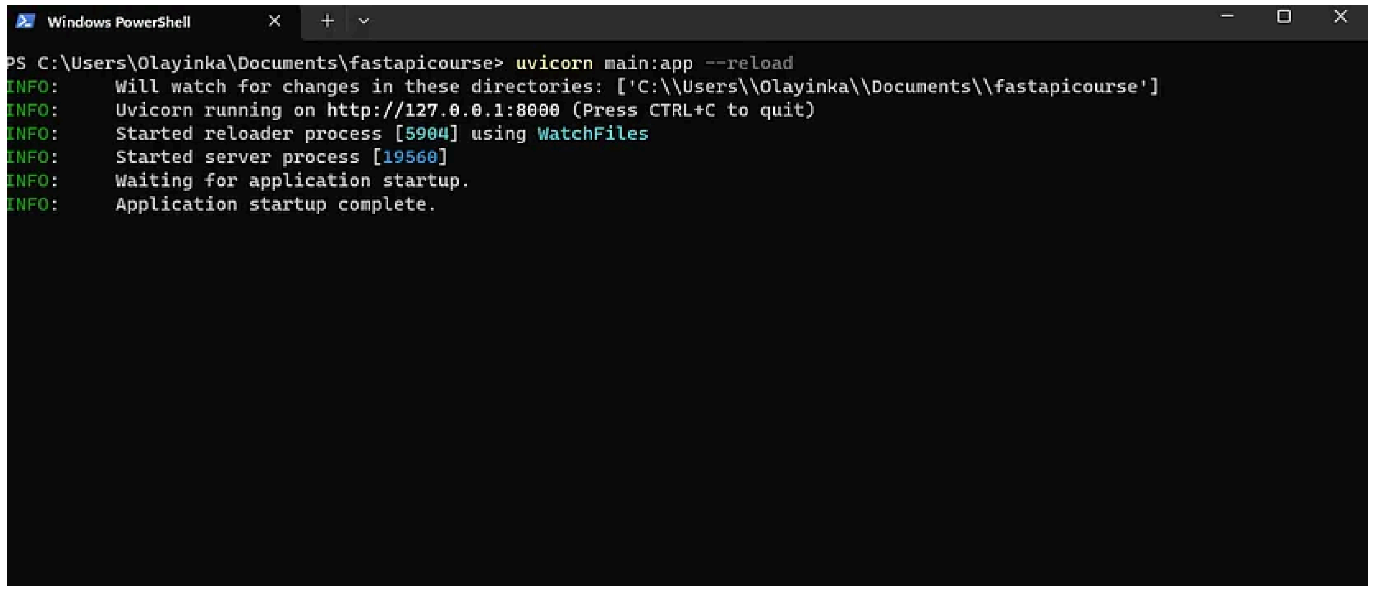
Create a file called *main.py* and populate it with the code below. In essence, it's a simple API with greetings at the root and an endpoint for handling items with associated IDs and optional queries.

```python
from typing import Union

from fastapi import FastAPI

app = FastAPI()


@app.get("/")
def read_root():
    return {"Hello": "World"}


@app.get("/items/{item_id}")
def read_item(item_id: int, q: Union[str, None] = None):
    return {"item_id": item_id, "q": q}
```

Save the file and run the following command in your terminal:

```
uvicorn main:app --reload
```

If you’ve been following along, you should see output that indicates the app started up successfully:



```
PS C:\Users\Olayinka\Documents\fastapicourse> uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\Users\\Olayinka\\Documents\\fastapicourse']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [5904] using WatchFiles
INFO:     Started server process [19560]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Create a file named requirements.txt in the project directory and add the following lines:

```
fastapi
uvicorn
```

The requirements.txt will be used to install the dependencies for the project when we set up the container.

Containerization

After making sure you've set up docker and have it running, you may proceed with this part of the tutorial.

Create a dockerfile in the root directory of the *fastapicourse* directory and add the following config to it.

```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

FROM python:3.11-slim: This line tells Docker to start building the image from the official Python 3.11-slim image. This image is a slimmed-down version of the official Python image, which means that it contains only the essential packages needed to run Python.

WORKDIR /app: This line sets the working directory for the container to /app. This means that all subsequent commands will be executed relative to this directory.

COPY requirements.txt .: This line copies the file requirements.txt from the local machine to the working directory of the container. The requirements.txt file contains a list of dependencies we specified earlier that the application needs to run.

RUN pip install -r requirements.txt: This line runs the pip command to install the Python packages listed in the requirements.txt file. This will install

all of the dependencies that the application needs to run.

COPY . .: This line copies all of the files from the local machine to the working directory of the container. This means that the application's source code and any other files that the application needs will be available in the container.

EXPOSE 8000: This line exposes port 8000 of the container. This means that the container will be able to listen for traffic on this port.

CMD ["uvicorn", "main: app", " — host", "0.0.0.0", " — port", "8000"]: This line sets the command that will be run when the container is started. The command will run the Uvicorn ASGI server, which will serve the application on port 8000.

Build the Docker image:

Create a docker-compose.yml file in the project directory and add the following:

```
version: "3"

services:
  app:
    build: .
    ports:
      - "8000:8000"
```

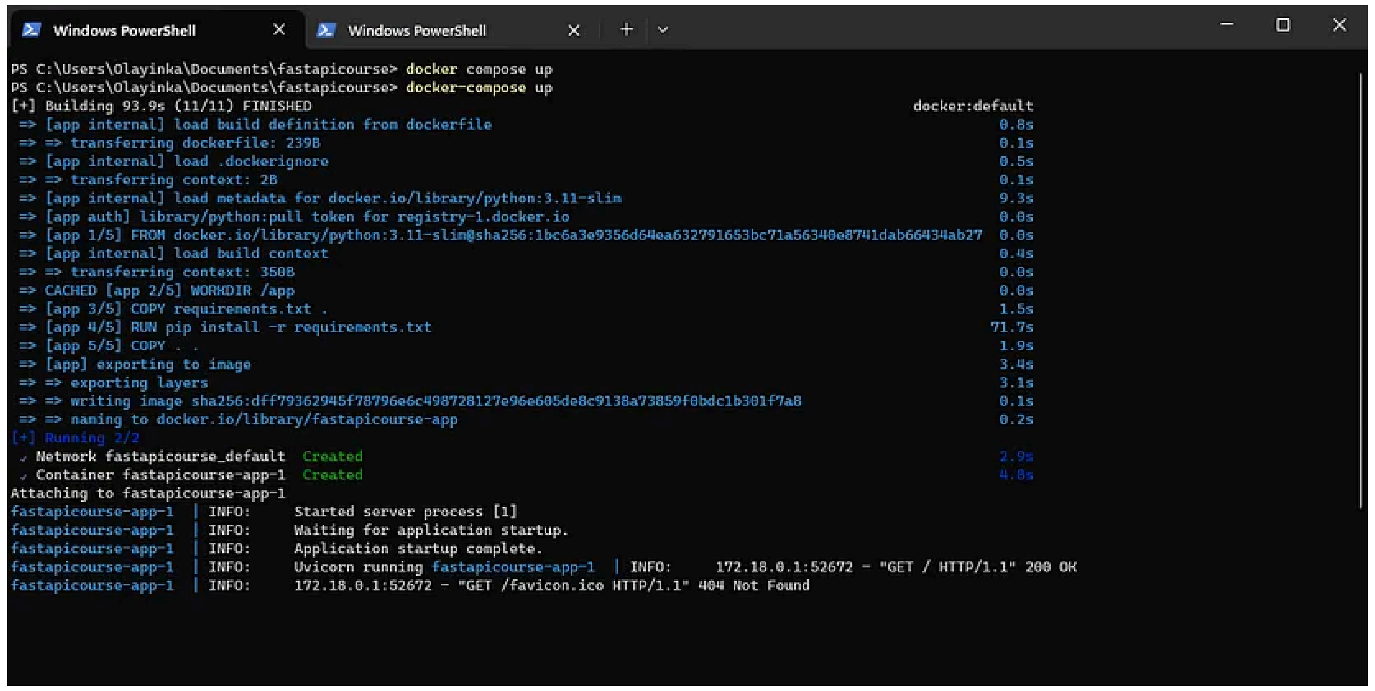
The docker-compose specifies a service named app and by running this command in your terminal:

```
docker-compose up
```

or run:

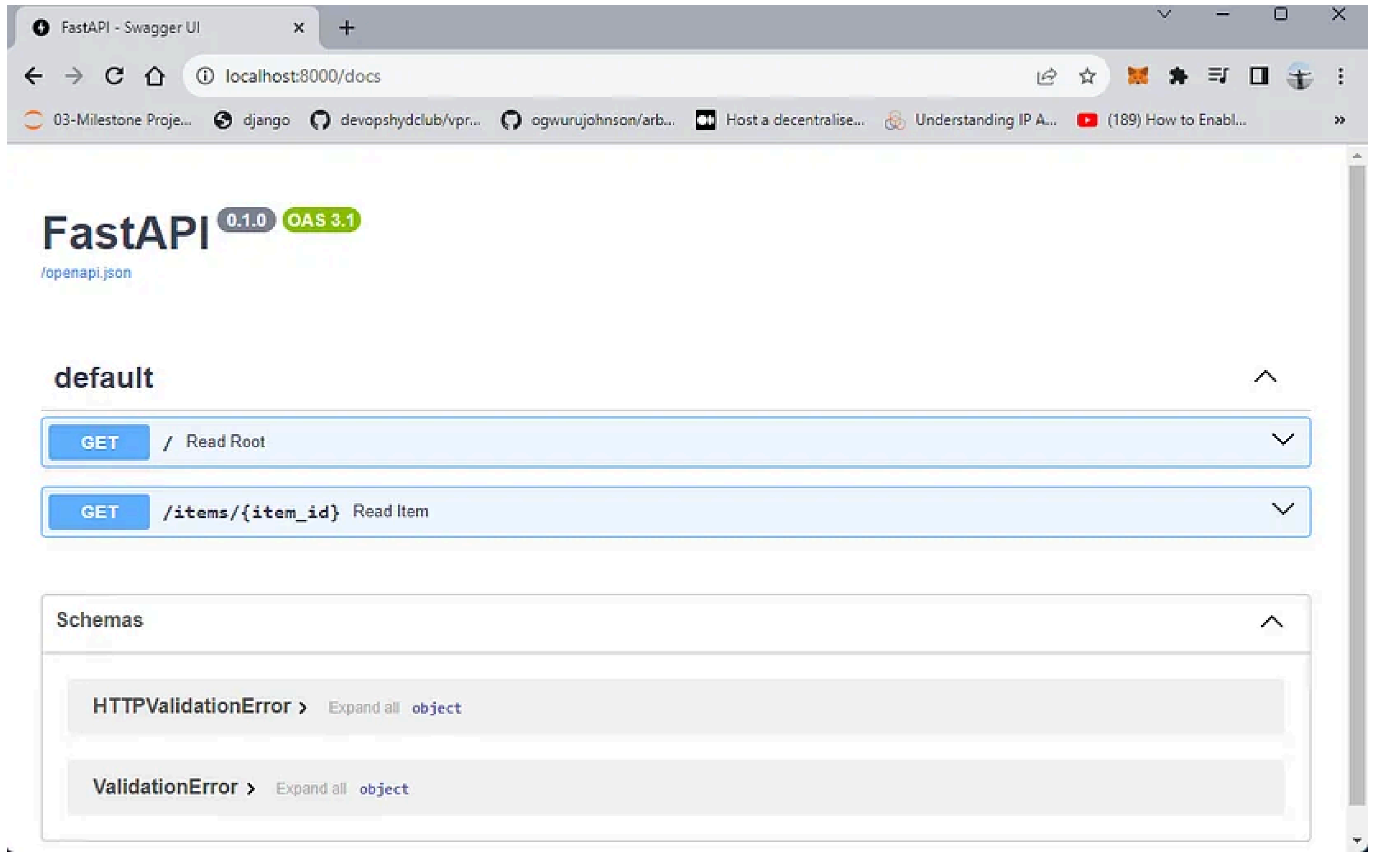
```
docker-compose up -d
```

to run the app in detached mode, the “app” container will be created and the container will run on the exposed port in the dockerfile. You should get a result like this in your terminal:



```
PS C:\Users\Olayinka\Documents\fastapicourse> docker compose up
PS C:\Users\Olayinka\Documents\fastapicourse> docker-compose up
[+] Building 93.9s (11/11) FINISHED                                                                    docker:default
 => [app internal] load build definition from dockerfile                                                         0.8s
 => => transferring dockerfile: 239B                                                                             0.1s
 => [app internal] load .dockerignore                                                                            0.5s
 => => transferring context: 2B                                                                                  0.1s
 => [app internal] load metadata for docker.io/library/python:3.11-slim                                          9.3s
 => [app auth] library/python:pull token for registry-1.docker.io                                                0.0s
 => [app 1/5] FROM docker.io/library/python:3.11-slim@sha256:1bc6a3e9356d64ea632791653bc71a56340e8741dab66434ab27  0.0s
 => [app internal] load build context                                                                            0.4s
 => => transferring context: 350B                                                                                0.0s
 => CACHED [app 2/5] WORKDIR /app                                                                                0.0s
 => [app 3/5] COPY requirements.txt .                                                                            1.5s
 => [app 4/5] RUN pip install -r requirements.txt                                                               71.7s
 => [app 5/5] COPY . .                                                                                           1.9s
 => [app] exporting to image                                                                                     3.4s
 => => exporting layers                                                                                          3.1s
 => => writing image sha256:dff79362945f78796e6c498728127e96e605de8c9138a73859f0bdc1b301f7a8                     0.1s
 => => naming to docker.io/library/fastapicourse-app                                                             0.2s
[+] Running 2/2
 ✓ Network fastapicourse_default  Created                                                                        2.9s
 ✓ Container fastapicourse-app-1  Created                                                                        4.8s
Attaching to fastapicourse-app-1
fastapicourse-app-1  | INFO:     Started server process [1]
fastapicourse-app-1  | INFO:     Waiting for application startup.
fastapicourse-app-1  | INFO:     Application startup complete.
fastapicourse-app-1  | INFO:     Uvicorn running fastapicourse-app-1  | INFO:     172.18.0.1:52672 - "GET / HTTP/1.1" 200 OK
fastapicourse-app-1  | INFO:     172.18.0.1:52672 - "GET /favicon.ico HTTP/1.1" 404 Not Found
```

If you get the FastAPI app running successfully, you can now visit http://localhost/docs to view the default page for FastApi


FastAPI - Swagger UI
localhost:8000/docs
FastAPI 0.1.0 OAS 3.1
/openapi.json
default
GET / Read Root
GET /items/{item_id} Read Item
Schemas
HTTPValidationError › Expand all object
ValidationError › Expand all object

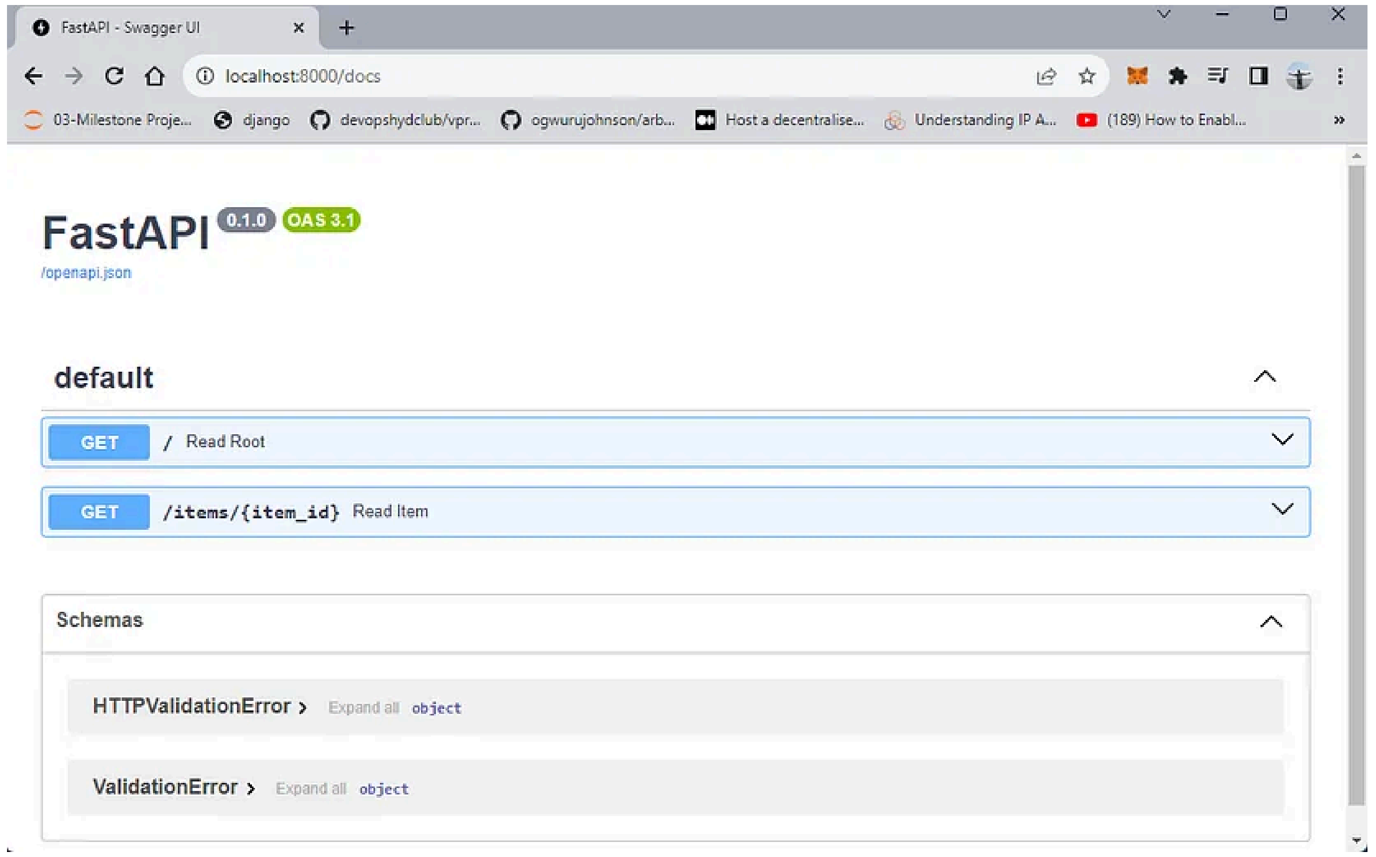
Congrats! You have successfully containerized a FastApi application. Please read the follow-up article where I set up the proxy.

Containerization Docker Compose Docker Fastapi Python

This is the second part of a three-part series. If you still haven't read part one, please check it out here as I will be building up on what was done in the previous tutorial.

Recap:

From the previous tutorial, we were able to get our Fast API app containerized


FastAPI - Swagger UI
localhost:8000/docs
FastAPI 0.1.0 OAS 3.1
/openapi.json
default
GET / Read Root
GET /items/{item_id} Read Item
Schemas
HTTPValidationError › Expand all object
ValidationError › Expand all object

You could also clone the GitHub repository to follow along.

Setting up config for Nginx

We have to provide the configuration for our proxy so it would be able to communicate with our API when they've both been containerized.

Create a new directory in the project directory, you may name it whatever you wish but I'm choosing to go with *nginx*. Now create another file in this directory called *nginx.conf* as this file will hold our configuration for nginx.

Now add the following lines to the nginx.conf file:

```
server {
    listen 80;
    server_name http://127.0.0.1;

    client_max_body_size 32m;

    location ^~ /.well-known/acme-challenge {
        default_type text/plain;
        root /var/www/letsencrypt;
    }

    location / {
        proxy_pass http://app:8000;
        return 301 https://$host$request_uri;
    }
}

# server {
#     listen 443 default_server ssl;
#     server_name example.com;
#     server_tokens off;

#     ssl_certificate     /etc/letsencrypt/live/example.com/fullchain.pem;
#     ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
#     ssl_dhparam         /etc/letsencrypt/dhparams/dhparams.pem;

#     client_max_body_size 32m;

#     location / {
#         proxy_pass http://app:8000;
#         proxy_set_header Host $http_host;
#         proxy_set_header X-Real-IP $remote_addr;
#         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
#         proxy_set_header X-Forwarded-Proto $scheme;
#     }
# }
```

The server_name http://127.0.0.1; directive specifies the server name clients should use to reach this server. In this case, the server name is 127.0.0.1, which is the loopback address for my local machine.

This config will tell nginx to proxy requests to our *app* container that we specified in the docker-compose, **notice this where I specified the *proxy_pass* parameter**. This configuration also includes commented-out boilerplate for enabling SSL which will definitely be useful when we might eventually need to add a domain name.

Updating docker-compose.yml

We need to update our docker-compose to include a container for Nginx.

```
version: "3"

services:
  app:
    build: .
    ports:
      - "8000:8000"
    restart: unless-stopped
    logging:
      driver: "json-file"
      options:
        max-file: "1"
        max-size: "100k"


    nginx:
      restart: unless-stopped
      image: nginx
      container_name: nginx
      ports:
        - "80:80"
        # - "443:443"
      # environment:
      #   - CERTBOT_EMAIL=youremail@gmail.com
      volumes:
        - ./nginx:/etc/nginx/user_conf.d:ro
        # - letsencrypt:/etc/letsencrypt
      depends_on:
        - app
      logging:
        driver: "json-file"
        options:
          max-size: "100k"
          max-file: "1"
```

This updated configuration file defines a multi-service Docker application consisting of two services: our *app* which we configured earlier and the newly added *nginx*.

Nginx configuration explained:

restart: unless-stopped: This tells Docker to restart the container if it crashes or is stopped manually.

image: nginx: This specifies that the default Nginx image from Dockerhub should be used for the container.

container_name: nginx*:* This assigns the name 'nginx' to the container.

ports:: This defines the ports that the container will expose.

- ***"80:80"***: This exposes the Nginx container's port 80 on the host machine as port 80.

volumes: This defines volumes that will be mounted into the container.

- ***./nginx:/etc/nginx/user_conf.d:ro***: This mounts the **nginx** directory from the current directory into the **/etc/nginx/user_conf.d** directory inside the container. The ***:ro*** flag indicates that the volume should be read-only.

depends_on:: This specifies that the **nginx** container depends on the **app** container. This means that the **nginx** container will not start until the **app** container is up and running.

logging:: This defines the logging configuration for the container.

driver: "json-file": This specifies that JSON-formatted logs should be written to files.

options:: This defines options for the JSON file logging driver.

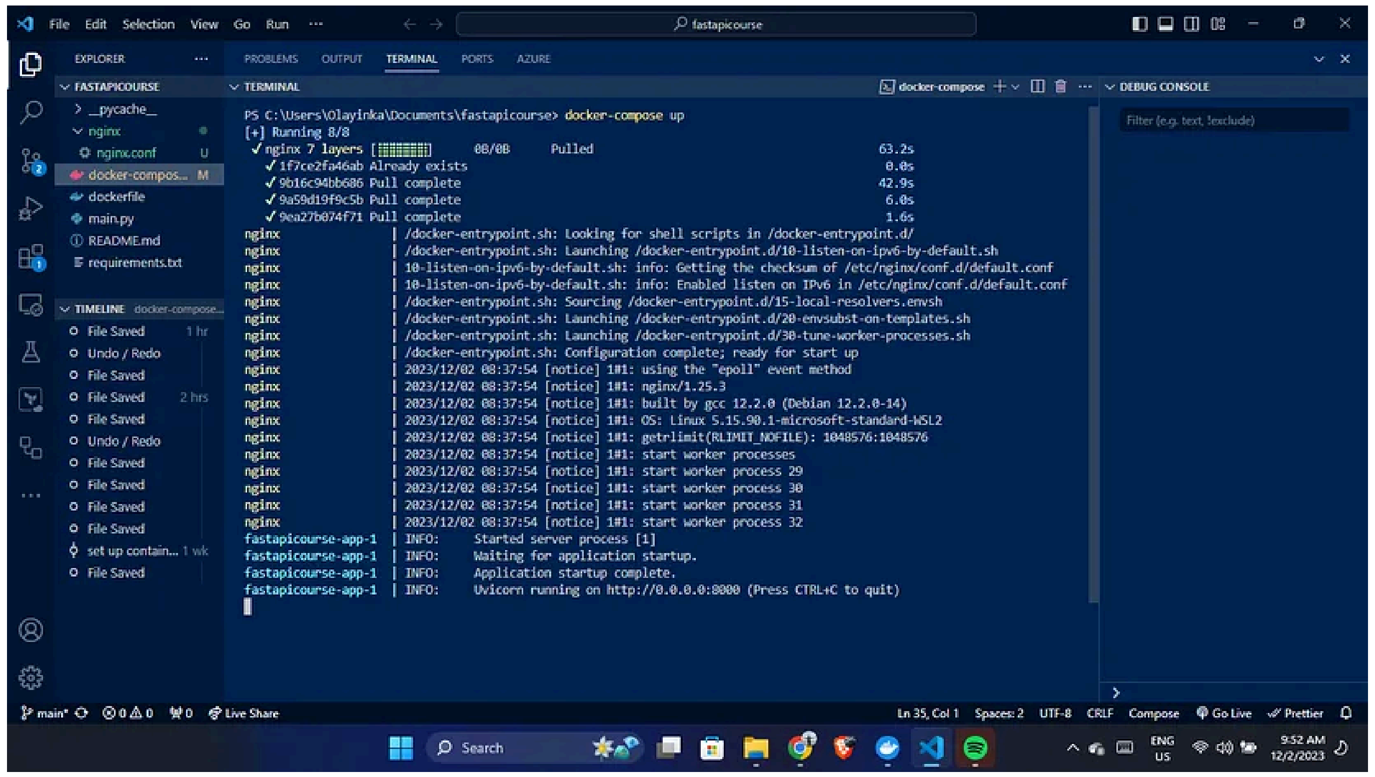
max-size: "100k": This limits the size of each log file to 100 kilobytes.

***max-file: "1"**:* This limits the number of log files to 1.

SSL-related configurations are present but commented out.

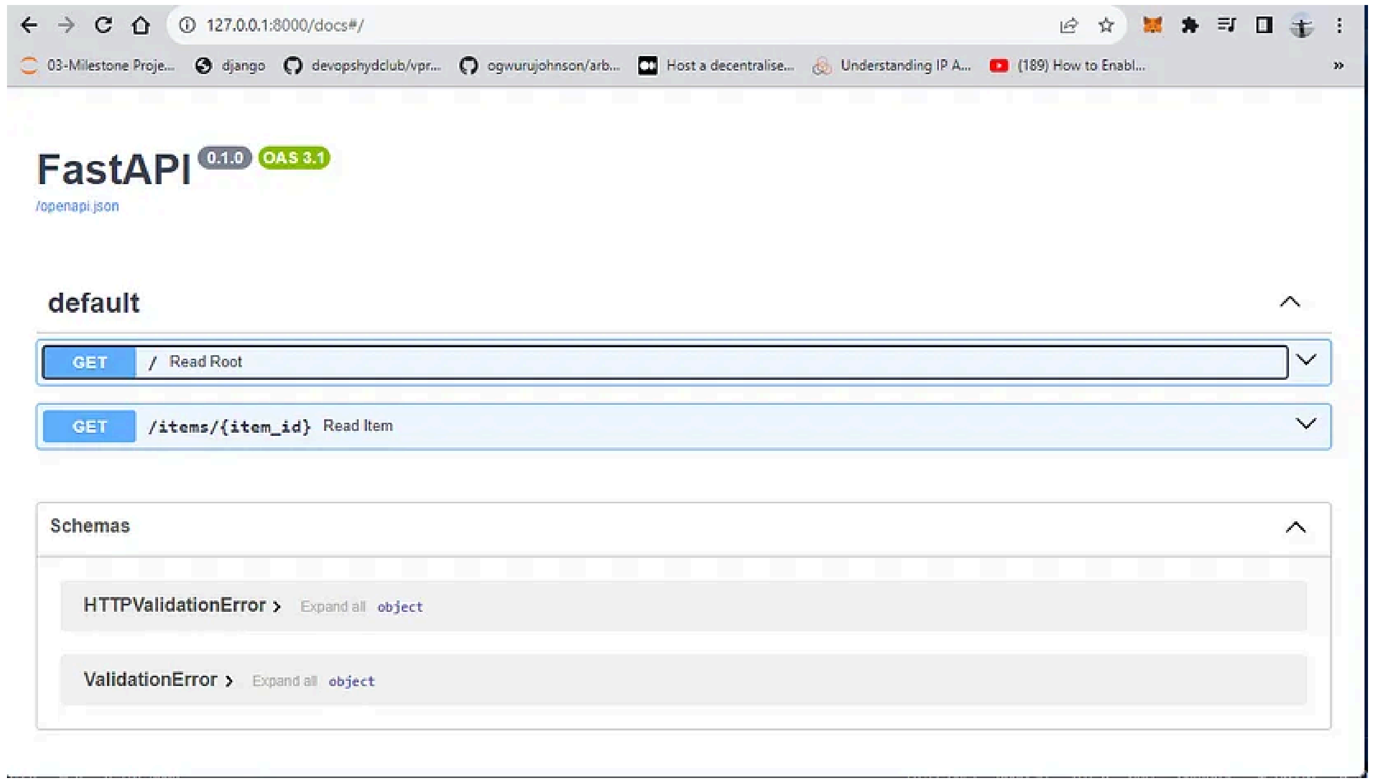
Now that we have our services containerized, we can then

```
docker-compose up
```



```
PS C:\Users\Olayinka\Documents\fastapicourse> docker-compose up
[+] Running 8/8
 ✓ nginx 7 layers [⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                                  63.2s
   ✓ 1f7ce2fa46ab Already exists                                                     0.0s
   ✓ 9b16c94bb686 Pull complete                                                     42.9s
   ✓ 9a59d19f9c5b Pull complete                                                      6.0s
   ✓ 9ea27b074f71 Pull complete                                                      1.6s
nginx                | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx                | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx                | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginx                | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
nginx                | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
nginx                | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx                | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx                | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx                | 2023/12/02 08:37:54 [notice] 1#1: using the "epoll" event method
nginx                | 2023/12/02 08:37:54 [notice] 1#1: nginx/1.25.3
nginx                | 2023/12/02 08:37:54 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
nginx                | 2023/12/02 08:37:54 [notice] 1#1: OS: Linux 5.15.90.1-microsoft-standard-WSL2
nginx                | 2023/12/02 08:37:54 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
nginx                | 2023/12/02 08:37:54 [notice] 1#1: start worker processes
nginx                | 2023/12/02 08:37:54 [notice] 1#1: start worker process 29
nginx                | 2023/12/02 08:37:54 [notice] 1#1: start worker process 30
nginx                | 2023/12/02 08:37:54 [notice] 1#1: start worker process 31
nginx                | 2023/12/02 08:37:54 [notice] 1#1: start worker process 32
fastapicourse-app-1  | INFO:     Started server process [1]
fastapicourse-app-1  | INFO:     Waiting for application startup.
fastapicourse-app-1  | INFO:     Application startup complete.
fastapicourse-app-1  | INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

While the containers are up and running, if you've been following along, you should now be able to visit http://127.0.0.1:8000/docs to view the default page for FastApi


127.0.0.1:8000/docs#/
FastAPI 0.1.0 OAS 3.1
/openapi.json
default
GET / Read Root
GET /items/{item_id} Read Item
Schemas
HTTPValidationError Expand all object
ValidationError Expand all object

Congratulations! 😊 You've successfully containerized your application and it's now accessible through a proxy as well. Please look out for the third and final part of this series where I will be deploying this app to the web.

Docker Nginx Proxy Python