Introduction to Fast API and Docker

Twitter: @HarunMbaabu

- What is Fast API?
- What is Docker?
- Why Fast API and Docker?
- Introduction to Docker Compose?

What is Fast API ?

FastAPI is a modern, fast with high-performance web framework for building APIs with Python.

It is easy to learn, fast to code, and production-ready. The most exciting feature of FastAPI is that it supports asynchronous code out of the box using the async/await Python keywords.

Testing FastAPI endpoints are really straightforward and can be done using TestClient provided by FastAPI. This makes Test Driven Development(TDD) very easy. We can easily deploy our apps via Dockers provided by FastAPI or via AWS Lambda.

What is Docker?

Docker is an open source platform for building, deploying, and managing containerized applications. It enables developers to package applications into containers, standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

Containers simplifys delivery of distributed applications, and have become increasingly popular as organizations shift to cloud-native development and hybrid multicloud environments.

So enough of talking let get our hands dirty and dockerize a simple Fast API application.

What is Docker?

A Docker image is a read-only, inert template that comes with instructions for deploying containers. In Docker, everything basically revolves around images.

A Docker container is a virtualized runtime environment that provides isolation capabilities for separating the execution of applications from the underpinning system. It’s an instance of a Docker image.

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

Hello World First API Application

First install and activate your virtual environment and install Fast API as shown below:

Installing virtual environment (windows/Linux/MacOS) :

```
pip3 install virtualenv
```

Create a virtual environment:

```
python3 -m venv luxenv
```

Activate your virtual environment:

```
source luxenv/bin/activate
```

Note: the following commands will only work on linux and MacOS, use this guide to learn how to install, create, and activate virtual environment on windows.

Installing Fast API

```
pip3 install fastapi
pip3 install uvicorn
```

server.py

```
from fastapi import FastAPI
app = FastAPI()
@app.get("/")
def read_root():
    return { "Lux app": "Welcome to Lux app" }
```

Creating a requirements.txt file:

```
>>> pip3 freeze > requirements.txt
```

In Python requirement.txt file is a type of file that usually stores information about all the libraries, modules, and packages in itself that are used while developing a particular project. It also stores all files and packages on which that project is dependent or requires to run. Typically this file "requirement.txt" is stored (or resides) in the root directory of your projects. Here another essential question arises why we need this type of file in our projects.

This requirements. txt file is used for specifying what python packages are required to run the project you are looking at. This is important because as you start developing your python applications, you will develop the application with specific versions of the packages in mind.

Dockerfile

```
FROM python:3.6
COPY .
COPY ./requirements.txt ./requirements.txt
WORKDIR .

EXPOSE 8000:8000
RUN pip install -r requirements.txt
CMD [ "uvicorn", "app:app", "--host", "0.0.0.0", "--reload"]
```

Building and running our first container

Build docker image called demo:

```
>>> docker build -t luxapp .
```

Run docker image called demo:

```
>>>docker run -p 8000:8000 -t -i luxapp
```

Now in your local browser visit http://127.0.0.1:0000 you should see the following result 👇🏻