Dockerize a Django app.

How to dockerize a Django app

This is all about how to package and distribute an API build in Django and Django REST Framework using docker containers.

Docker is an open platform that performs Operating System-level virtualization also known as containerization.

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings. Available for both Linux and Windows-based apps, containerized software will always run the same, regardless of the environment. Containers isolate software from its surroundings, for example, differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure — Docker documentation.

In order to dockerize the Django app first,

Open the project where the Django app created in a text editor(or any editor). Then create a docker file.
Docker file is a list of instructions that docker will follow while creating our image.

Docker has two stages one is a build stage and another is run stage. On the build stage, it builds the container image. It looks the docker file and creates an image and runs all of the commands in the docker file.

Add a file named `Dockerfile` without any extension.
We are using a python-based image. `FROM python:3`. Here we are using official python as a parent image. Next thing is to create an environment `ENV PYTHONUNBUFFERED 1.` Here the environment variable is set which instructs the Docker not to buffer the output from buffer from the standard output buffer, but simply send it straight to the terminal. We want to create a directory is called for the source code. The folder will be created in the container.
`RUN mkdir /code`.

Next, we have to create a working directory `WORKDIR /code`.

Now we want to copy all the files for our project to our container. As of this point in time, the container does not contain any source code. We have to copy the code from the local machine to the container using the COPY command `COPY . /code/` means all files from the local to the code directory named code in the container.

we have to install all the required libraries for our python project by running pip install on the requirements.txt file which would have been copied with the rest of the things in the COPY command above

Docker Compose

Compose is a tool for defining and running multi-container Docker applications. — Docker documentation

So for a simple docker file, this is enough. Next, we have to compose the docker-compose config. Which defines all the necessary services for our app. In a distributed application, different pieces of the app are called “services”. For example, if you imagine a video sharing site, it probably includes a service for storing application data in a database, a service for video transcoding in the background after a user uploads something, a service for the front-end, and so on.

In the docker-compose, we have to specify which version of the docker-compose syntax we are using. We are using docker-compose version 3.

So add `version:'3'` .Now we can define the services.
```` ``` ````

```
version:'3'
services:
    web:
      build: .
      command: python <path to migrate>/manage.py runserver 0.0.0.0:8000 #
path of the manage file
      volumes:
        - .:/code  #maps the volume from the local volume to the docker.
when a change is made in the source code it will simultaneously change the
code in the docker container.
      ports:
        - "8000:8000" # we are creating a port from our container to our
local machine. Here in the example when we running our development server
```
````

```
port 8000 it will map that to our host machine. When we have access to port
8000 to the host machine will fold our connection to the docker. container.
```

Next is set up our database for our for the project." setting up" the database is done by docker-compose ..one of the services would be the database and when you run docker-compose up it creates a service/container that is running database on it.

what manage.py migrate does is create all the tables and data inside the created database.

More technically, your Django project will have a lot of migrations that are created in this folder called "migrations" ... when you run migrate above, Django/python runs this set of commands on your SQL database to make everything ready for your app to run.
So add a migration command to migrate our project. So run the command in the terminal `docker-compose run web python <path to migrate>/manage.py migrate`
you can also manually build the container using the command in the terminal `docker-compose build` you can see that its running everything that we defined. Now the image is created it `docker-compose up` (but it's best to rebuild if you have changed because docker will not rebuild if it has a cached version already saved on your disk)