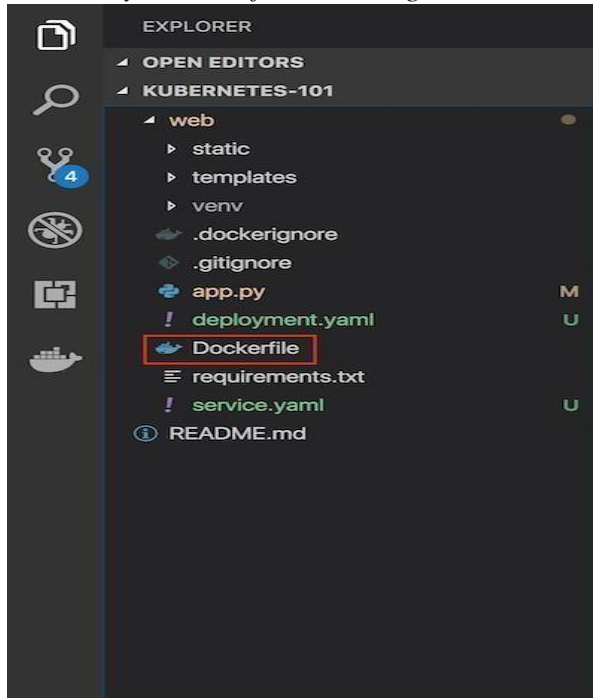


Containerize The App

DATE	20/11/2022
Team ID	PNT2022TMID46361
Project Name	Project-Inventory Management for Retailers

Containerize your Flask application:

- In your project directory, create a file named "Dockerfile." *Suggestion: Name your file exactly "Dockerfile," nothing else.*



A "Dockerfile" is used to indicate to Docker a base image, the Docker settings you need, and a list of commands you would like to have executed to prepare and start your new container. In the file, paste this code:

```
• LABEL maintainer="Kunal Malhotra, kunal.malhotra1@ibm.com"
```

```
FROM python:2.7
```

- RUN apt-get update
- RUN mkdir /app
- WORKDIR /app
- COPY ./app
- RUN pip install -r requirements.txt
- EXPOSE 5000
- ENTRYPOINT ["python"]
- CMD ["app.py"]

Explanation and breakdown of the above Dockerfile code

1. The first part of the above code is:

```
FROMpython:2.7
```

2. Because this Flask application uses Python 2.7, we want an environment that supports it and already has it installed. Fortunately, DockerHub has an official image that's installed on top of Ubuntu. In one line, we will have a base Ubuntu image with Python 2.7, virtualenv, and pip. There are tons of images on DockerHub, but if you would like to start off with a fresh Ubuntu image and build on top of it, you could do that.

3. Let's look at the next part of the code:

```
4. LABELmaintainer="KunalMalhotra,kunal.malhotra1@ibm.com"  
5. RUNapt-getupdate
```

6. Note the maintainer and update the Ubuntu package index. The command is `RUN`, which is a function that runs the command after it.

```
7. RUN mkdir/app  
8. WORKDIR/app  
9. COPY ./app
```

10. Now it's time to add the Flask application to the image. For simplicity, copy the application under the `/app` directory on our Docker Image.

`WORKDIR` is essentially a `cd` in bash, and `COPY` copies a certain directory to the provided directory in an image. `ADD` is another command that does the same thing as `COPY`, but it also allows you to add a repository from a URL. Thus, if you want to clone your git repository instead of copying it from your local repository (for staging and production purposes), you can use that. `COPY`, however, should be used most of the time unless you have a URL.

11. Now that we have our repository copied to the image, we will install all of our dependencies, which is defined in the **requirements.txt** part of the code.

```
RUNpipinstall--no-cache-dir-rrequirements.txt
```

12. We want to expose the port(5000) the Flask application runs on, so we use **EXPOSE**.

```
EXPOSE5000
```

13. **ENTRYPOINT** specifies the entry point of your application.
14. **ENTRYPOINT** ["python"]
15. **CMD** ["app.py"]

Build an image from the Dockerfile

Open the terminal and type this command to build an image from your Dockerfile: `docker build -t <image_name>:<tag>` .(note the period to indicate we're in our apps top level directory). For example: `docker build -t app: latest` .

```
kunal@mbp:~/kunal/malhotra$ docker build -t app:latest .
Sending build context to Docker daemon 348.2kB
Step 1/8 : FROM python:2.7
--> 6c76e39e7cfe
Step 2/8 : LABEL maintainer="Kunal Malhotra, kunal.malhotra@ibm.com"
--> d8b57d41591c
Step 3/8 : RUN apt-get update
--> Using cache
--> 6262a134e40e
Step 4/8 : COPY ./app
--> f877237999f
Step 5/8 : WORKDIR /app
Removing intermediate container f9018b93d2fe
--> 0accef20b3d
Step 6/8 : RUN pip install -r requirements.txt
--> Running in 8153040b00b7
Collecting click==6.7 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/34/c1/8806f99713db993c5366c362b2f908f18269f8d792aff1abfd700775a77/click-6.7-py2.py3-none-any.whl (71kB)
Collecting Flask==1.0.2 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/7f/67/08578774ed4536d3242b14d0cb4696386634607a7824e0997202cd0ebdb4b7/flask-1.0.2-py2.py3-none-any.whl (91kB)
Collecting itsdangerous==0.24 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/d2/b4/6d0bcbb945c00f6d608d8975131ab3f25b22f2bcef1dab221165194b2d4/itsdangerous-0.24.tar.gz (46kB)
Collecting Jinja2==2.10 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/7f/f6/ae64bacdfc95f27a016a7bed8e866763ba4d277a78ca76f32659220a731/jinja2-2.10-py2.py3-none-any.whl (126kB)
Collecting MarkupSafe==1.0 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/20/cf/12c3e56473e52375aa29c4764e70d1bf3efad6682bef8d8a0e4fe335243/MarkupSafe-1.0.tar.gz
Collecting Werkzeug==0.14.1 (from -r requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/20/cf/12c3e56473e52375aa29c4764e70d1bf3efad6682bef8d8a0e4fe335243/Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
Building wheels for collected packages: itsdangerous, MarkupSafe
Running setup.py bdist_wheel for itsdangerous: started
Running setup.py bdist_wheel for itsdangerous: finished with status 'done'
Stored in directory: /root/.cache/pip/wheels/2c/4a/61/5599631c1554768c6290b08c02c72d7317910374ca602ff1e5
Running setup.py bdist_wheel for MarkupSafe: started
Running setup.py bdist_wheel for MarkupSafe: finished with status 'done'
Stored in directory: /root/.cache/pip/wheels/33/56/20/eb49a5c612ffef1c50632146b16596f9e4676768661e4e6
Successfully built itsdangerous MarkupSafe
Installing collected packages: click, itsdangerous, MarkupSafe, Jinja2, Werkzeug, Flask
Successfully installed Flask-1.0.2 Jinja2-2.10 MarkupSafe-1.0 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24
Removing intermediate container 8153040b00b7
--> 66a036a07be
Step 7/8 : ENTRYPOINT [ "python" ]
--> Running in bd1c8315e1
Removing intermediate container bd1c8315e1
--> 73cfc38ac1c
Step 8/8 : CMD [ "app.py" ]
--> Running in a784d430dd6f
Removing intermediate container a784d430dd6f
--> d8b0a83763a5
Successfully built d8b0a83763a5
Successfully tagged app:latest
kunal@mbp:~/kunal/malhotra$
```

Run your container locally and test

After you build your image successfully, type: `docker run -d -p 5000:5000 app`
This command will create a container that contains all the application code and dependencies from the image and runs it locally

