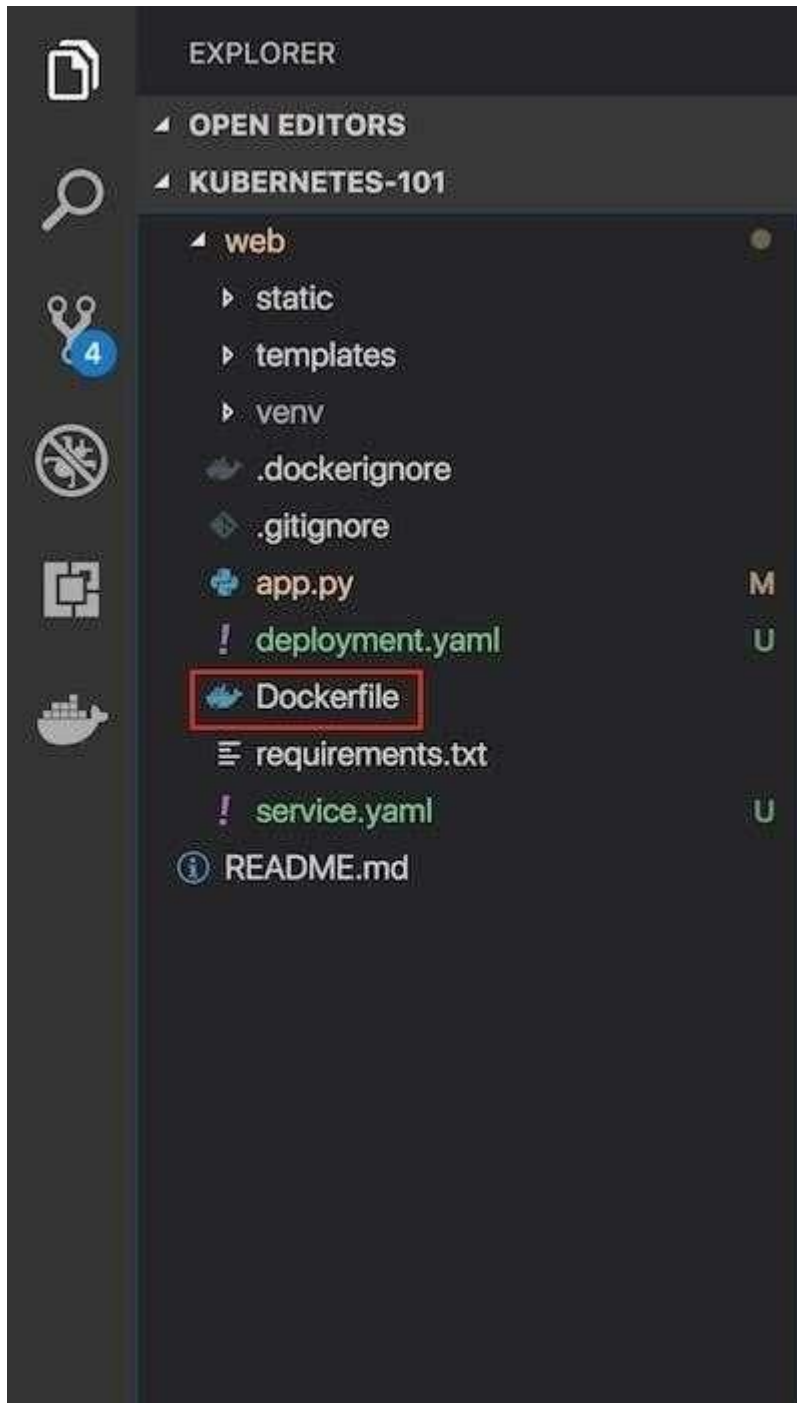


## CONTAINERIZE THE APP

<b>Team ID</b>	PNT2022TMID25063
<b>Project Name</b>	Skill / Job Recommender Application
<b>Date</b>	09 October 2022

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:

```
FROM python:2.7
LABEL maintainer="preetha,preetha@ibm.com"
```
- RUN apt-get update
- RUN mkdir /app WORKDIR /app COPY . /app
- RUN pip install -r requirements.txt

- EXPOSE 5000
- ENTRYPOINT [ "python" ]
- CMD [ "app.py" ]

Show more

## Explanation and breakdown of the above Dockerfile code

1.

The `FROM python:2.7`  
first

part of the code above is:

2.

Show more

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. LABEL maintainer="preetha-preetha@ibm.com"
5. RUN apt-get update

Show more

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`

Show more

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,

`RUN pip install --no-cache-dir -r requirements.txt`

which is defined in the `requirements.txt` part of the code.

12.

Show more

13.

We  
want `EXPOSE 5000`

to expose the port(5000) the Flask application runs on, so we use `EXPOSE`.

14.

Show more

15. `ENTRYPOINT` specifies the entrypoint of your application.

16. `ENTRYPOINT` [ "python" ]

17. `CMD` [ "app.py" ]

Show more

## 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 .`

```
humbly@humbly:~/kubernetes$ docker build -t app:latest .
Sending build context to Docker daemon 348.1kB
Step 1/8 : FROM python:2.7
----> 6c26e18a7cfe
Step 2/8 : LABEL maintainer="Kunal Mishra; kunal.mishra@fintec.com"
----> Using cache
----> 88c574d1291c
Step 3/8 : RUN apt-get update
----> Using cache
----> 6326d34e46e
Step 4/8 : COPY ./app
----> 10777270229f
Step 5/8 : WORKDIR /app
Removing intermediate container f90d8e9583fe
----> 80c6ef70c3d5
Step 6/8 : RUN pip install -r requirements.txt
----> Running in 8113b486007
Collecting click==6.7 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/9a/c1/1800/96715a666c330c362b2f900718209fba731af1ba7f00775a77c110c4-6.7-py2.py3-none-any.whl (71kB)
Collecting Flask==0.12 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/7f/e6/8857874b4536d343d14d6c0026188636820af820e95738208e6d4b010c1-0.12-py2.py3-none-any.whl (291kB)
Collecting itsdangerous==0.24 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/d0/b4/6086d0b45c08f1a08b68075111d0f725a2772ef61d0b221160104b294/itsdangerous-0.24.tar.gz (40kB)
Collecting Jinja2==2.10 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/7f/f6/96f6e0f095f27a0b7e2888067630e4d57e7b076332932204711761a2-2.10-py2.py3-none-any.whl (125kB)
Collecting MarkupSafe==1.1.1 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/49/3c/106355467a788820637001e77e48255e9295a64f1b0f4170b40e450fe-1.1.1-py2.py3-none-any.whl (125kB)
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/20/4a/45/55961c155476b0c2b0b0460272d7317816374a0d827f1e5
  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/13/56/20/d0e48c5d1277e1c56632156b1693019c647070862e4e6
Successfully built itsdangerous MarkupSafe
Installing collected packages: click, itsdangerous, MarkupSafe, Jinja2, Werkzeug, Flask
Successfully installed Flask-0.12 Jinja2-2.10 MarkupSafe-1.1 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24
Removing intermediate container 8113b486007
----> 6c26e18a7cfe
Step 7/8 : ENTRYPOINT ["python"]
----> Running in bdc3c8815d2
Removing intermediate container bdc3c8815d2
----> 7509c38a3c
Step 8/8 : CMD ["app.py"]
----> Running in c7944030a4p
Removing intermediate container a74449b48df
----> 8806d3783d5
Successfully built 8806d3783d5
Successfully tagged app:latest
humbly@humbly:~/kubernetes$
```

## Run your container locally and test

After you build your image succesfully, 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.

