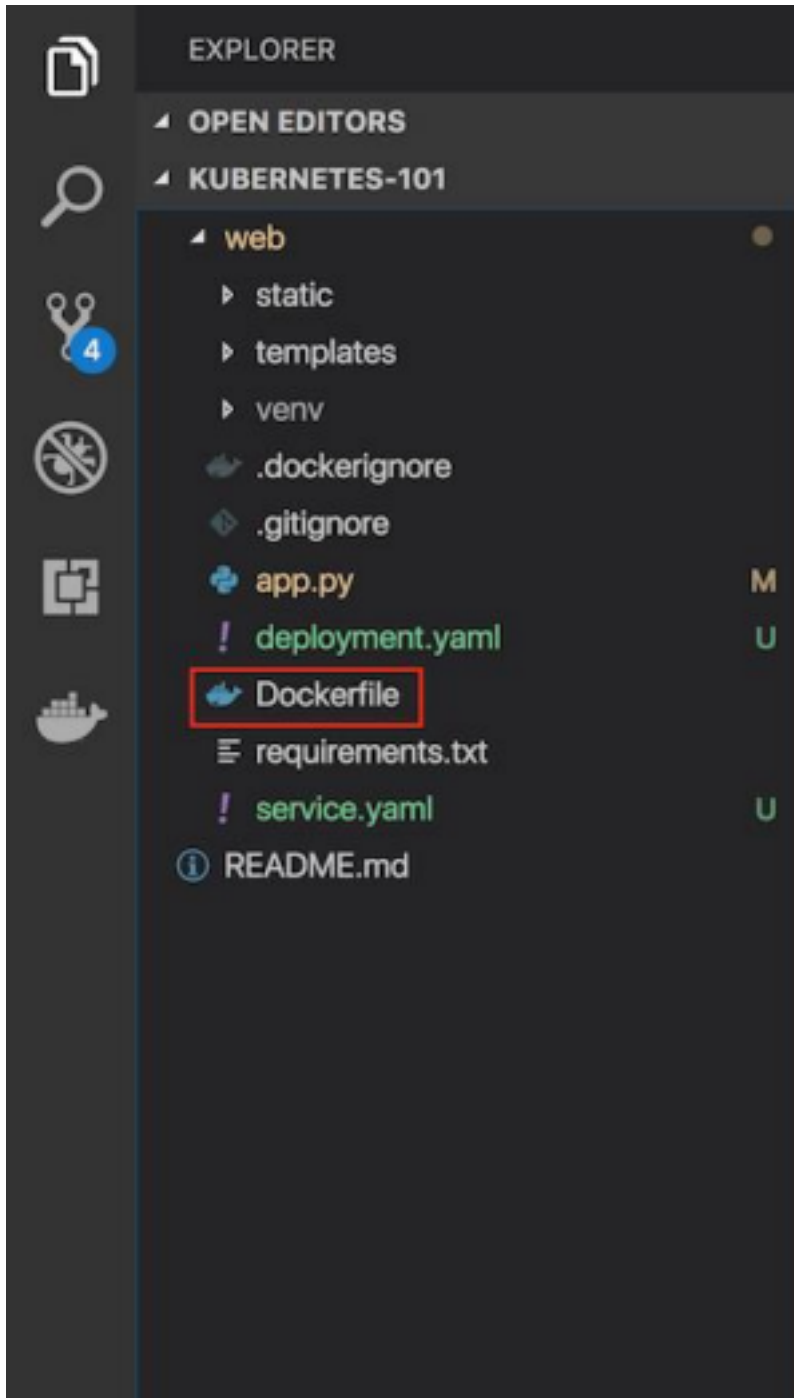


## DEPLOYMENT OF APP IN IBM CLOUD

### CONTAINERIZE THE APP

- 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="Kunal Malhotra, kunal.malhotra1@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 first part of the code above is:
2. FROM python:2.7

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="Kunal Malhotra, kunal.malhotra1@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, which is defined in the requirements.txt part of the code.
12. RUN pip install --no-cache-dir -r requirements.txt

Show more

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

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 .`

```

user@img-ubuntu-kernel-hypervisor:~$ cd /opt/teitak
Sending build context to Docker daemon 348.2kB
Step 1/8 : FROM python:2.7
--> 65b36b070f6
Step 2/8 : LABEL maintainer="Kuntal Acharya, kuntal.acharya@gmail.com"
--> Using cache
--> a8607d1189c1
Step 3/8 : RUN apt-get update
--> Using cache
--> 65b36b070f6
Step 4/8 : CMD ["/app"]
--> 86297117699f
Step 5/8 : WORKDIR /app
Removing intermediate container f90c8b66276
--> 86c8e17b0d1
Step 6/8 : RUN pip install -r requirements.txt
--> Running in 61130a0a0c7
Collecting click==6.7 (from -r requirements.txt [line 12])
  Downloading https://files.pythonhosted.org/packages/34/c1/8b60997346d995c3566362c9798b1820b45d573e11fab030675e771c1dc-6.7-py2.py3-none-any.whl (79kB)
Collecting Flask==1.0.2 (from -r requirements.txt [line 21])
  Downloading https://files.pythonhosted.org/packages/7f/61/9857f77ead43563d34b146c34833632400f9324cd07261c3b2b467f3a3e-1.0.2-py2.py3-none-any.whl (95kB)
Collecting Flask-Login==0.2.9 (from -r requirements.txt [line 31])
  Downloading https://files.pythonhosted.org/packages/34/46/46b0c4b5d0f9a00889b9711ca17f5a2123c733a022105246d3a97/flask_login-0.2.9.tar.gz (40kB)
Collecting Flask-SocketIO==2.1.8 (from -r requirements.txt [line 41])
  Downloading https://files.pythonhosted.org/packages/79/94/4680a3cf0912742126e3d4887630e427749c9f32830323a71272e3a2-2.10-py2.py3-none-any.whl (23kB)
Collecting MarkupSafe==0.23 (from -r requirements.txt [line 50])
  Downloading https://files.pythonhosted.org/packages/5d/4e/3d4714b73d6746768632a67381e7f4d32546995d64f0d41123/MarkupSafe-1.0.tar.gz
Collecting Werkzeug==0.14.1 (from -r requirements.txt [line 60])
  Downloading https://files.pythonhosted.org/packages/0b/64/12cd06473c2575a02c704679a3d73ef0662be3f3d8a044c325153/Werkzeug-0.14.1-py2.py3-none-any.whl (530kB)
Building wheels for collected packages: Flask-SocketIO, Werkzeug, MarkupSafe
Running setup.py bdist_wheel for Flask-SocketIO: started
Running setup.py bdist_wheel for Werkzeug: finished with status 'done'
Stored in directory: /root/.cache/pip/wheels/52/46/92/799963c235476b320866a82c7327c7918074a0007f345
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/53/56/28/46d4b5d17774e15d31146307676d47676861e446
Successfully built Flask-SocketIO Werkzeug MarkupSafe
Installing collected packages: click, Flask-SocketIO, Werkzeug, Flask
Successfully installed Flask-1.0.2 Werkzeug-0.14.1 click-6.7 Flask-SocketIO-2.10
Removing intermediate container 81538060805
--> 66d3b30070c
Step 7/8 : ENTRYPOINT ["python"]
--> Running in 5b113812d1
Removing intermediate container b4b38312e1
--> 749e33463c1
Step 8/8 : CMD ["/app.py"]
--> Running in a70403060f
Removing intermediate container a79403060f
--> a8611702d1
Successfully built a8611702d1
Successfully tagged opt/teitak
user@img-ubuntu-kernel-hypervisor:~$

```

## 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.

[illegible]

