

## Containerize the App

Date	07 November 2022
Team ID	PNT2022TMID36379
Project Name	Skills/Job Recommender Application

### 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="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" ]`

## 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 .`

```
kunali@kunal:~$ docker build -t app:latest .
Sending build context to Docker daemon 348.2kB
Step 1/8 : FROM python:2.7
--> 6c76b6e7cfe
Step 2/8 : LABEL maintainer="Kunal Khutria, kunali.kh@netai.com"
--> Using cache
--> 48c5941591c
Step 3/8 : RUN apt-get update
--> Using cache
--> 626a13444e
Step 4/8 : COPY . /app
--> f07777709d9
Step 5/8 : WORKDIR /app
Running intermediate container f04b9940fe
--> 86cd578c3e
Step 6/8 : RUN pip install -r requirements.txt
--> Running in 8153b4b8b7
Collecting click==6.7 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/34/c1/886f607348665c396c362c21966f18209f6a732aff1abfd700775e77/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/69/8637674e48536d342b144c34963468fa8324e09572d2c0b0b4b91f1e31/Flask-1.0.2-py2.py3-none-any.whl (33kB)
Collecting itsdangerous==0.24 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/d0/b4/6086b8071310d52729c2f73c6f1dd2211651946246/itsdangerous-0.24.tar.gz (46kB)
Collecting Jinja2==2.10 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/7f/ff/a0b4eeefc95f27f03a7e08a886763e4d277a78a7f12893228a7117e1a2-2.10-py2.py3-none-any.whl (128kB)
Collecting MarkupSafe==1.0 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/6d/6e/32f41d35b48f6768822a03700ef7e4825ee9206e4f1b0f417b/MarkupSafe-1.0.tar.gz
Collecting Werkzeug==0.14.1 (from -r requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/0b/42/2c3c54473e5375ac79c476447bd1873ef0062b9f680a4fe33243/Werkzeug-0.14.1-py2.py3-none-any.whl (325kB)
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/0c/46/61/559961c154768c52984680c726731751874c04607f1ab
  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/33/36/28/4e93c0d27f7a1c5e63274031899f96467070862e4e46
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 8153b4b8b7
--> 48c5941591c
Step 7/8 : ENTRYPOINT ["python"]
--> Running in b63c83815d
Running intermediate container b63c83815d
--> 75cfc33c1c
Step 8/8 : CMD ["app.py"]
--> Running in a784430b0ef
Running intermediate container a784430b0ef
--> 86cd578c3e
Successfully built app:latest
Successfully tagged app:latest
kunal@kunal:~$
```

## 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.

```
kunal@kunal:~$ docker run -d -p 5000:5000 app
3131f6d773e3d406a6522c0c363c4944b3112c0543e0d692d7
kunal@kunal:~$ docker ps
CONTAINER ID        IMAGE               COMMAND
3131f6d773e3d406a6522c0c363c4944b3112c0543e0d692d7      app                "python app.py"
kunal@kunal:~$
```

