

## Containerize the App

Date	07 November 2022
Team ID	PNT2022TMID37091
Project Name	News Tracker 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 .`

```

kubernetes@kali:~/kubernetes$ docker build -t app:latest .
Sending build context to Docker daemon 348.3kB
Step 1/8 : FROM python:2.7
--> 6c76395cfe
Step 2/8 : LABEL maintainer="Kunal Mishra, kunal.mishra@pmo.com"
--> Using cache
--> d85e941291c
Step 3/8 : RUN apt-get update
--> Using cache
--> 67c333664e
Step 4/8 : COPY ./app
--> 1677377669f
Step 5/8 : WORKDIR /app
Removing intermediate container f03d8f9a3fe
--> 86c6e7263c
Step 6/8 : RUN pip install -r requirements.txt
--> Running in 81290a0000
Collecting click==6.7 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/36/c1/880819953a8865c566162c90881269167630f1a6f708775a77f1c16c6-6.7-py2.py3-none-any.whl (71kB)
Collecting Flask==0.12.0 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/7f/e7/8857674e4536a342b14e4dc6934863660f332e0957802080eb4b71a1c1.0-py2.py3-none-any.whl (51kB)
Collecting Werkzeug==0.14 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/d6/b6/a0148ba845819c6684807511c1d729a2727cfe1a6221169184246/ Werkzeug-0.14.tar.gz (444k)
Collecting Jinja2==2.10 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/72/fd/7e7447eefcf5f2f7e101a7e618a6867830e4d27e701761359126f3113e1a2-2.10-py2.py3-none-any.whl (124kB)
Collecting MarkupSafe==1.0 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/60/30/f118358467a78882263780f7e7a48255a92046f1d6f4370a/MarkupSafe-1.0.tar.gz
Collecting Werkzeug==0.14.1 (from -r requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/2b/c4/723c36473e2375a29c476a76d1871ef166a287f80a46e35243/Werkzeug-0.14.1-py2.py3-none-any.whl (325kB)
Building wheels for collected packages: Werkzeug, MarkupSafe
  Running setup.py bdist_wheel for Werkzeug: started
  Running setup.py bdist_wheel for Werkzeug: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/2c/46/41/559b31c154766c2eb680724731781f7a0d4f11e
  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/23/56/70/4765c027f11c32246109919647670803c4e4e
Successfully built Werkzeug MarkupSafe
Installing collected packages: click, Werkzeug, MarkupSafe, Jinja2, Werkzeug, Flask
Successfully installed Flask-0.12.0 Jinja2-2.10 MarkupSafe-1.0 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24
Removing intermediate container 8150a64867
--> 6c76395cfe
Step 7/8 : ENTRYPOINT [ "python" ]
--> Running in bdc58815d
Removing intermediate container b03d8381cd
--> 75c6f38ac1c
Step 8/8 : CMD [ "app.py" ]
--> Running in a78490a8df
Removing intermediate container a78490a8df
--> 86c6e7263c
Successfully built app:latest
Successfully tagged myk8s-test
kubernetes@kali:~/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.

[illegible]

