Week 12 - Big-Scale Analytics

Building a Docker image

Docker images are templates that contain all the files, dependencies, etc required to run a container. The main step in building an image is creating a "`dockerfile`". Docker can build images automatically by reading the instructions from a "`dockerfile`". A "`dockerfile`" is a text document that contains all the commands a user could call on the command line to assemble an image. Using "`docker build`" users can create an automated build that executes several command-line instructions in succession.

When dockerizing Python applications, another important text file to create is the "`requirements.txt`" file. It contains all the dependencies (i.e, Python libraries) that are required by the application and should be installed.

In what follows, we will give you a walk-through of dockerizing a Python Flask application.

Dockerizing a Flask application

Before starting, you need to create a new directory that will contain your docker files. Create a new directory with a name of your choice and copy the Flask app files from week 9 to this directory. We are going to dockerize the Flask app from week 9.

Important note: make sure that inside the application.py script, for $app.run()$ method you have set the host to be 0.0.0.0 (example below)

```
if __name__ == '__main__':
    app.run(debug = True, host='0.0.0.0')
```

This is a necessity, as 0.0.0.0 is a wildcard IP address that will match any possible incoming port on the host machine.

The next step is to create a dockerfile. Create a new file named **dockerfile** (note that this file **should not have any extension**) and edit it with an editor of your choice. Here's a view of how this file should look like in our case:

```
FROM ubuntu:16.04

MAINTAINER Your Name "your_email"

RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev

COPY ./requirements.txt /app/requirements.txt

WORKDIR /app

RUN pip3 install --upgrade pip
RUN pip3 install -r requirements.txt

COPY . /app
EXPOSE 5000

ENTRYPOINT [ "python3" ]

CMD [ "application.py" ]
```

Let's go over this dockerfile line by line:
1. We base our image on ubuntu version 16.04. This means that when building the image, docker will pull the image of this version of ubuntu from Docker Hub.
2. MAINTAINER sets the author field of the image. This will be useful if you want to push your image to Docker Hub later.
3. The command coming after RUN will be executed in a shell. This means that Python3 and pip3 will be installed on ubuntu (the parent image)
4. Copy the "requirements.txt" file from the local machine to "/app" directory in the ubuntu image.
5. Set the working directory in ubuntu to be "/app". This is equivalent to when you do "cd /app" in the terminal of your machine.
6. Update pip3
7. Install the libraries inside "requirement.txt" file.
8. Copy all the files in the current directory of your machine (the directory in which dockerfile is) to the "/app" directory in ubuntu. This will copy the Python scripts to "/app" directory.
9. The EXPOSE instructions informs Docker that the container listens on the specified network ports at runtime. The port 5000 is the one used by Flask
10. ENTRYPOINT configures the container to run as an executable.Here we want to execute a Python3 script.
11. Finally, the last line will execute the "application.py" script

Note that the last two line will internally run `python3 application.py` inside the docker container.

It only remains to create the "requirements.txt" file. We need the following libraries for our application. Note that the versions are specified after the "==" sign.

```
Flask==0.10.1
joblib==0.14.1
numpy==1.18.4
scipy==1.4.1
scikit-learn==0.22.2.post1
pandas==0.24.2
```

Now you are ready to build your image. You can do it by simply running the following in the terminal:

```
docker build -t flask-churn-app:latest .
```

What comes before ":" is the name of the image and what comes after it specifies that this is the latest version of this image. The "." Means that we want to build the docker file in the current directory. After the build completed, you can run the container:

```
docker run -it —d -p 5000:5000 flask-churn-app
```

The "-it" option means that the container runs in interactive mode, "-d" means the container will run in background, and "-p" means that container's port 5000 is published to host's port 5000.

Now you container is running and you can open the Flask web app from your browser.

Recommended tutorial to watch: watch this tutorial video which explains more in details some of the above-mentioned concepts.