Intro to Docker workshop - worksheet 2

This document will guide you through the creation of docker images via Dockerfiles. By the end of this worksheet, you should have the ability to easily build your own docker images.

Follow the instructions and think about the questions made at each step. If you need help, feel free to call for help from the instructors.

- So, as far as you know, you're able to build your own images by pulling from a base image and copying over files and installing the programs you need. That's pretty cool, but you can STILL be easier.

  Let's say you have a cool script that gives you up-to-date information on bus arrivals. You want to have a container that you deploy and immediately start getting weather info. Go ahead and create a **weather** directory. Inside, create a file called **Dockerfile**. Type in **FROM ubuntu:latest**, save it and run **docker build . -t weather**.

  Look at your images. You'll see a nice little weather image. If you run it, you'll see that it's nothing more than an ubuntu image with another name, but we'll work on that.

- We have an ubuntu container, but our awesome bus arrival app needs python to run, so we should install python on our ubuntu container, right?

  No. It's a lot smarter to download an image that already has what we need, so go ahead and replace your **ubuntu image** with **python:latest**.

  Then we can start by adding the source code of our app to the image. Type **RUN mkdir /code** to create a new folder and **ADD bus.py /code/** to get our script inside the folder. **Note that each command needs to be in its own line**.

  You can re-build the image and run, and you'll see that your code now lives comfortably within the **code** directory in our container!

- Now we need to install the script's dependencies. This is more of the same:
  ADD requirements.txt /code/
  RUN pip install -r requirements.txt

- So now we can run our bus container and see our app living inside the container and we can run it by hand. But what if we could make run at startup so that we don't have to get stuck inside the container?

Just add **ENTRYPOINT [“python”, “/code/bus.py”]** ! Now you can run the image passing the code of the stop you want to monitor and just watch it. Isn’t it wonderful?

- **FINAL EXERCISE:** Go and find out the difference between the ADD and COPY commands and check out the other possible commands. If you’re feeling brave, try to get an **expressJS API** up and running in a docker container.