Lab: Building Images

In this lab, you will use the Docker tool to build a custom image from a small Python Flask application.

## Get the code

1. From this lab directory, make sure you have the Python script “app.py” available.
2. Try running the script with “python app.py” This will probably fail if you don’t have Python installed, or the correct Python dependencies.

## Create a Dockerfile for the code

1. Create a new Dockerfile based on the examples from our materials, or from the Docker documentation.
2. Select the “bookworm” base image for Python 3.11.
3. Ensure the app.py script gets copied into your image.
4. Ensure the Python library **Flask** is installed in your container by including a RUN command to run “python -m pip install Flask”
5. Ensure the port 8888 is open in the image.
6. Set the follow default ENV variables:  
   ENVIRONMENT=dev  
   NAME=student
7. Ensure the CMD is set to run the app.py script using Python

## Build the container

1. Using the “docker build” command, build your image locally and tag it with something well-known, like “flask-app”
2. Inspect your built image to make sure the settings look correct as you would expect from your Dockerfile.

## Test the image

1. Run a container from your image - the container should start, output some text, and wait for web requests on port 8888.
2. Make requests to the container’s IP address and observe the output in your terminal. You can press Ctrl-C to stop Flask and exit the process, which will stop your container.

## “Deploy” the image to a staging environment

1. Run another container from the same image (you may need to give it a different name). Make sure you override the environment variables using the --env/-e option to “docker run” and provide YOUR name as the NAME variable, and “stage” as the ENVIRONMENT variable. You can also run the container in “detached” mode if you like to put it in the background.
2. Using curl or your browser, make requests to the container IP address. Note that the environment variables have changed the output of the script, even though you have not changed the code or the image.

## Experiment with the Image

1. Modify your Dockerfile to see if you can optimize it. Try changing the base image to see if you can get the custom image smaller.
2. Try changing the order of the commands in the Dockerfile. Edit the code (you can just add a comment with a #, for example) and observe the output of “docker build” to see what effect it has on the caching and build time.
3. Try pushing your newly created image up to DockerHub under your account.