Docker

Agenda

1. Introduction
2. Docker Files
3. Docker Build
4. Docker Container
5. Docker Hub
6. Docker Application

Introduction

1.1 What is Docker?

- An open-source software platform that debuted in 2013 with the purpose of creating, deploying, and managing virtual application containers on "common" operating system.

```
- Behold, The Whale.
```

Docker Files

2.1 What is a docker file?

- A text file with instructions for creating your docker image, which is a file with executable code. It is similar to a makefile for other programming languages and their compliers but is used for the file to work with docker.
- This in turn provides a form of automation for creating the docker image when creating for the first time and afterwards.
- A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

2.2 Creating a docker file

- Via VS code extension, … F1 while using the docker extension and selecting the type of file we want and its entry point (the main file it will execute)
- Via command line, touch DockerFile and vim Dockerfile to then edit it
- docker list to view created docker files

2.3 Editing the docker file

Docker Build

1. In VS Code, select Terminal > New Terminal.
2. In the terminal window or a Bash window, run this command.

docker run -d -p 80:80 docker/getting-started

If you want to remove it…

1. Run this command to get its container ID: docker ps
2. Then stop and remove the container:

docker stop docker rm

1. Refresh your browser. The contained built a moment ago is gone.

Docker Container

Docker Hub

5.1 docker push [OPTIONS] NAME[:TAG]

- docker push behaves much like Github push does. It will push any changes made up to the image it is said to push to.
- for the [OPTIONS], you have -a, -q, and --disable-content-trust
- -a pushes all tagged images up to either your local repo or your cloud repo.
- NAME is the name of your image. [:TAG] is whatever tag you want to give it, like a version number.

5.2 docker push

- docker push MYIMAGE:VERSION // docker push myimage:1.0
- If you push myimage:1.0, make a change, and then make a new one, myimage:1.1, you will be able to grab both versions, and run the differences in them. Remember, each push is a unique, read-only image. Keeping track of Version History is helpful, much like Github Commits