**Running Apps with Containers (Docker)**

**Introduction to Containers and Docker**

* Software development platform to enable easy deployment through containers
* A container contains everything that we need to run our application code
* Docker is system agnostic, meaning it can run in any OS
* We can run many docker containers, where you may be able to run just a few virtual machines
* Docker communicates natively with the system kernel
* Docker containers can share files with other Docker containers
* We begin with a Docker file, which can be built into a Docker image, which can be ran as a Docker container
  + Docker file: Run commands such as downloading, installing and running your software
  + Docker image:
    - You can share with others
  + Docker container:
    - Runs the docker image
* Containers are more lightweight
* Reproducibility due to having the code for the docker file and the docker image
* Isolation, as we can easily replace containers

**Installing Docker**

* We installed Docker tools

**Creating Our API App Dockerfile**

* Show Dockerfile
  + It is used to create a Docker image
  + 1: We will be using Python 3.6.4 Docker image from Docker hub
    - It comes with Python ready to use
    - This is a Linux base image
      * Thus, in the subsequent commands we will use Linux syntax
  + 6: Where subsequent commands will be run
  + 12: We are taking everything from our local api directory and we are copying that over to the container
  + 23: Specifies the command that will be run when our container starts up
* Show ml\_api\run.sh
  + We are using Gunicorn to start our Flask application
* Show dockerignore file
  + This is gonna be used when we run our Docker build command

**Building and Running the Docker Container**

* First, we build the Docker image
* Command:
  + PIP\_EXTRA\_INDEX\_URL is referenced by ARG
    - $PIP\_EXTRA\_INDEX\_URL$ references our environment variable that we have set
  + -t ml\_api (name of the image) : latest (tag of the image)
  + “.” Says to search for the docker file in the current directory
* *docker images*
  + will show you the images that you have created
* Command:
  + *docker run* (to build a Docker container given the Docker image) (run the docker image)
  + *–name ml\_api* (specifies the name of the image) *-d* (runs it in the background) *-p 8000:5000* (the ports that we want open) -rm (remove something after we are done)
* *docker ps*
  + shows live the containers that are running
* *docker logs* CONTAINER\_ID
  + to check the logs of the container

**Releasing/Deploying to Heroku with Docker**

* Show .circleci
  + *setup\_remove\_docker* is a CircleCI feature that allows you to build docker images in CircliCI
  + HEROKU\_EMAIL and HEROKY\_API\_KEY are new environment variables that we will need to set
    - registry.heroky.com is a registry of Docker images that Heroku mantains
      * allows us to work with Docker images within Heroku
  + 131 make commands
  + 135 Heroku container is telling Heroku that we’ll be using a Docker image when we do our release instead of using the command line as we did before
* Show Makefile
  + 6: Creating the Docker image from the dockerfile
    - This time in the Linux syntax instead of Windows syntax
    - -t tags our images
      * So for any given image we can trace back to the exact git commit
  + 9: We take our built image and push it to the Heroku registry