**DESCRIPTION**

**Docker Images**

Blueprint for a container

[This video from VMWare draws an analogy to docker images being like Java classes with containers being analogous to Java objects](https://www.youtube.com/watch?v=EnJ7qX9fkcU&t=4s)

Images form a kind of heirarchy. One image will be "From" another with added info, dependencies, commands, applications, etc. The added info and command each form a new layer on the image. With each of these layers being indicated in a the Dockerfile that defines what's needed for the image.

Images are named and tagged with the version. They also have an id which uniquely identifies them.

**Existing Docker Images**

Pull images from some existing registry(repository of images). The default configuration is from the DockerHub.

* docker pull \*image name\*
* docker run \*image name\* (this will pull the image if it doesn't already exist in the local system)

**Building Our Own Images**

* Dockerfile
* From existing container
  + docker commit

We can then push our images to a given registry including DockerHub

**Managing Images**

We can use the docker CLI to manage the images on our local system. We can list out the existing images, get their details, remove and update them.

Additionally, we can use the CLI to aid in connecting to a registry to quickly and easily distribute changes. In this case updating the software is as simple as updating the image.

Users can easily pull new images and spin up containers/applications with the modifications made.

**References**

* Video linked to above from VMWare (This is also in the Containerization References)
* [References listed in Refernces in Containerization Refs.](https://app.revature.com/admin-v2/)