Docker

Getting Started

Setting up your computer

```
#test Docker installtion
docker run hello-world
```

Hello World

Playing with Busybox

run a Busybox container on systemand get a taste of the docker run command.

```
#pull command fetches the busybox image from the Docker registry and
saves it to system
docker pull busybox

#see a list of all images on system
docker images
```

Docker Run

```
#run a Docker container based on this image
##the Docker client finds the image
##loads up the container
##runs a command in that container
docker run busybox

docker run busybox echo "hello from busybox"

#show all containers that are currently running
docker ps

docker ps -a

#run more than just one command in a container
docker run -it busybox sh
/ # ls
```

```
/ # uptime
/ # exit

docker run --help

#deleting containers
docker rm 305297d7a235 ff0a5c3750b9

#deletes all containers that have a status of exited
##-q only returns the numeric IDs
##-f filters output based on conditions provided
docker rm $(docker ps -a -q -f status=exited)
docker container prune

#delete images
docker rmi
```

Terminology

- *Images* - The blueprints of our application which form the basis of containers. In the demo above, we used the `docker pull` command to download the **busybox** image.
- *Containers* - Created from Docker images and run the actual application. We create a container using `docker run` which we did using the busybox image that we downloaded. A list of running containers can be seen using the `docker ps` command.
- *Docker Daemon* - The background service running on the host that manages building, running and distributing Docker containers. The daemon is the process that runs in the operating system which clients talk to.
- *Docker Client* - The command line tool that allows the user to interact with the daemon. More generally, there can be other forms of clients too - such as Kitematic which provide a GUI to the users.
- *Docker Hub* - A registry of Docker images. You can think of the registry as a directory of all available Docker images. If required, one can host their own Docker registries and can use them for pulling images.

Webapps with Docker

Static Sites

pull a Docker image from Docker Hub, run the container and see how easy it is to run a webserver.

```
#download and run the image
##--rm automatically removes the container when it exits
```

```
##-it specifies an interactive terminal which makes it easier to kill
the container with Ctrl+C (on windows)
docker run --rm -it prakhar1989/static-site

#detached mode: terminal is not attached to the running container
##-d will detach our terminal
##-P will publish all exposed ports to random ports
##--name corresponds to a name we want to give
docker run -d -P --name static-site prakhar1989/static-site

#see the ports
docker port static-site

http://localhost:32769

#specify a custom port to which the client will forward connections to
the container
docker run -p 8888:80 prakhar1989/static-site

#stop a detached container
docker stop static-site
```

Docker Images

TAG - refers to a particular snapshot of the image

IMAGE ID - is the corresponding unique identifier for that image

An image akin to a git repository - images can be committed with changes and have multiple version. If you don't provide a specific version number, the client defaults to latest

```
#pull a specific version of ubuntu image
docker pull ubuntu:18.04

#search for images
docker search
```

Base images are images that have no parent image, usually images with an OS like ubuntu, busybox or debian.

Child images are images that build on base images and add additional functionality.

Official images are images that are officially maintained and supported by the folks at Docker. These are typically one word long. In the list of images above, the python, ubuntu, busybox and hello-world images are official images.

User images are images created and shared by users like you and me. They build on base images and add additional functionality. Typically, these are formatted as user/image-name.

Our First Image

```
#clone a fun little Flash app
git clone https://github.com/prakhar1989/docker-curriculum.git
cd docker-curriculum/flask-app
```

All user images are based on a base image.

Since our application is written in Python, the base image we are going to use will be Python 3.

```
#check the available size of /dev/mapper/rehel-root
df -h
```

```
#check all docker images
docker images

#remove docker images, can remove multiple at one time by adding a space
and another IMAGE ID behind
##example of IMAGE ID: 2ddec3130c1a
docker rmi -f <IMAGE ID>
```

```
#check currently running docker containers
docker ps

#check all docker containers including paused one
docker ps -a

#remove docker containers, can remove multiples at one time by adding
space and another CONTAINER ID behind
##exmaple of CONTAINER ID: dac3d7b03556
docker rm -f <CONTAINER ID>
```

```
#check the syntax of a function
yum
yum -help
yum install -help
```