How To Docker

. Install Docker and Virtualbox
    a. brew install docker
    b. brew install docker-machine
    c. Install Virtualbox from Managed Software Center
1. https://docs.docker.com/machine/get-started/#prerequisite-information
    a. docker-machine create --driver virtualbox Char
2. https://docs.docker.com/machine/reference/ip/
    a. docker-machine ip Char
3. https://docs.docker.com/machine/reference/env/
    a. docker-machine env Char
    b. eval $(docker-machine env Char)
4. https://hub.docker.com/_/hello-world/
    a. docker pull hello-world
5. docker run hello-world
6. https://hub.docker.com/_/nginx/
    a. https://docs.docker.com/engine/reference/commandline/run/#options
    b. named overlord
        i. --name
    c. background task
        i. -d
    d. restart on its own
        i. --restart always
    e. have its 80 port attached to the 5000 port of Char.
        i. -p 5000:80
    f. docker run --name overlord -d -p 5000:80 --restart always nginx
7. https://stackoverflow.com/questions/43692961/how-to-get-ip-address-of-running-docker-container
    a. docker inspect -f "{{ .NetworkSettings.IPAddress }}" overlord
8. https://docs.docker.com/engine/reference/commandline/run/#options
    a. The -it instructs Docker to allocate a pseudo-TTY connected to the container's stdin; creating an interactive bash shell in the container.
    b. docker run --rm -it alpine /bin/sh && docker ps
9. docker run --rm -it debian /bin/sh
    a. apt-get update -y
    b. apt-get upgrade -y
    c. apt-get install gcc -y
    d. apt-get install git -y
10. https://docs.docker.com/engine/reference/commandline/volume_create/
    a. https://docs.docker.com/engine/admin/volumes/volumes/
    b. docker volume create hatchery
11. https://docs.docker.com/engine/reference/commandline/volume_ls/
    a. docker volume ls

12. https://docs.docker.com/samples/library/mysql/
    a. restart on its own in case of error
        i. --restart on-failure
    b. root password of the database should be Kerrigan\
        i. MYSQL_ROOT_PASSWORD=Kerrigan
    c. database is stored in the hatchery volume
        i. -v hatchery:/var/lib/mysql
    d. the container directly creates a database named zerglings
        i. -e MYSQL_DATABASE=zerglings
    e. the container itself is named spawning-pool.
        i. --name spawning-pool
    f. docker run -d --restart on-failure -v hatchery:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=Kerrigan -e MYSQL_DATABASE=zerglings --name spawning-pool mysql
13. https://docs.docker.com/engine/reference/commandline/exec/
    a. docker exec spawning-pool env
14. https://docs.docker.com/samples/library/wordpress/
    a. it should be able to use the spawning-pool container as a database service.
        i. --link spawning-pool:mysql
    b. docker run -d --name lair -p 8080:80 --link spawning-pool:mysql wordpress
15. https://hub.docker.com/r/phpmyadmin/phpmyadmin/
    a. it should be able to explore the database stored in the spawning-pool container.
        i. --link spawning-pool:db
    b. docker run -d --name roach-warden -p 8081:80 --link spawning-pool:db phpmyadmin/phpmyadmin
16. https://docs.docker.com/engine/reference/commandline/logs/
    a. docker logs spawning-pool --follow
17. docker ps
18. docker restart overlord

19. https://docs.docker.com/engine/reference/commandline/run
    a. Create container
        i. docker run -dt --name Abathur -v ~/Abathur:/root -p 3000:3000 python:2-slim
            1. name Abathur
                a. -name Abathur
            2. It will be a Python container, 2-slim version
                a. Python:2-slim
            3. its /root folder will be bound to a HOME folder on your host
                a. https://docs.docker.com/engine/reference/commandline/run/#mount-volume--v-read-only
                b. -v ~/Abathur:/root
            4. its 3000 port will be bound to the 3000 port of your virtual machine
                a. https://docs.docker.com/engine/reference/commandline/run/#publish-or-expose-port--p-expose
                b. https://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/#connect-using-network-port-mapping
                c. -p 3000:3000
    b. docker exec Abathur pip install flask
        i. Flask micro-framework in its latest version
            1. Python container -> use pip to install flask
    c. Create hello.py
        i. http://flask.pocoo.org/docs/0.12/quickstart/
            1. echo "from flask import Flask
               app = Flask(__name__)
               @app.route('/')
               def hello_world():
                   return '<h1>Hello World</h1>'" > hello.py
    d. Copy hello.py to container
        i. https://docs.docker.com/engine/reference/commandline/cp/
        ii. docker cp hello.py Abathur:/root/hello.py
    e. Run flask and serve hello.py
        i. docker exec --env FLASK_APP=hello.py Abathur flask run --host=0.0.0.0 --port=3000
        ii. https://docs.docker.com/engine/reference/commandline/exec/#options
            1. --env FLASK_APP=/root/hello.py

iii. http://flask.pocoo.org/docs/0.12/quickstart/
    1. flask run --host=0.0.0.0 --port=3000
20. https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/
    a. docker swarm init --advertise-addr $(docker-machine ip Char)

21. See Question 1
    a. docker-machine create --driver virtualbox Aiur
22. https://stackoverflow.com/questions/35046165/how-do-i-switch-between-active-docker-machines-on-osx
    a. eval $(docker-machine env Aiur)
    b. docker swarm join --token $(docker swarm join-token worker -q) $(docker-machine ip Char):2377
    c. eval $(docker-machine env Char)
    d. docker node ls
23. https://docs.docker.com/engine/reference/commandline/network_create/#extended-description
    a. docker network create --driver overlay overmind
24. https://docs.docker.com/engine/reference/commandline/service_create/
    a. named orbital-command
        i. --name orbital-command
    b. on the overmind network
        i. --network overmind
    c. rabbitmq SERVICE
        i. https://docs.docker.com/samples/library/rabbitmq/
        ii. rabbitmq
    d. define a specific user and password for the RabbitMQ service
        i. https://www.rabbitmq.com/configure.html#define-environment-variables
        ii. --env RABBITMQ_DEFAULT_USER=Protoss --env RABBITMQ_DEFAULT_PASS=KerriganBlows
    e. docker service create --name orbital-command --network overmind --env RABBITMQ_DEFAULT_USER=Protoss --env RABBITMQ_DEFAULT_PASS=KerriganBlows rabbitmq
25. https://docs.docker.com/engine/reference/commandline/service_ls/
    a. docker service ls
26. https://hub.docker.com/r/42school/engineering-bay/
    a. Launch a 42school/engineering-bay service
        i. 42school/engineering-bay
    b. two replicas
        i. --replicas 2
    c. named engineering-bay
        i. --name engineering-bay
    d. will be on the overmind network
        i. --network overmind
    e. docker service create --name engineering-bay --replicas 2 --network overmind --env OC_USERNAME=Protoss --env OC_PASSWD=KerriganBlows 42school/engineering-bay

27. https://docs.docker.com/engine/reference/commandline/service_logs/#description
    a. https://docs.docker.com/engine/reference/commandline/service_ps/
    b. docker service logs $(docker service ps engineering-bay -f "name=engineering-bay.1" -q) -f
28. https://hub.docker.com/r/42school/marine-squad/
    a. Launch a 42school/marine-squad service
        i. 42school/marine-squad
    b. two replicas
        i. --replicas 2
    c. named engineering-bay
        i. --name marines
    d. will be on the overmind network
        i. --network overmind
    e. docker service create --name marines --replicas 2 --network overmind --env OC_USERNAME=Protoss --env OC_PASSWD=KerriganBlows 42school/marine-squad
29. https://docs.docker.com/engine/reference/commandline/service_ps/
    a. docker service ps marines
30. https://docs.docker.com/engine/reference/commandline/service_scale/
    a. docker service scale marines=20
31. https://docs.docker.com/engine/reference/commandline/service_rm/
    a. https://docs.docker.com/engine/reference/commandline/service_ls/
    b. docker service rm $(docker service ls -q)
32. http://blog.baudson.de/blog/stop-and-remove-all-docker-containers-and-images
    a. docker rm -f $(docker ps -aq)
33. https://docs.docker.com/engine/reference/commandline/rmi/
    a. docker rmi -f $(docker images -q)
34. https://docs.docker.com/machine/reference/rm/
    a. docker-machine rm Aiur -y

Dockerfiles

- https://docs.docker.com/engine/reference/builder/#format
- A text document that contains all the commands a user could call on the command line to assemble an image.
- docker build . = make .
    - Can specify a path or url where url can be git repo
- Instructions are not case sensitive but the convention is for them to be uppercase to distinguish them more from arguments
- Lines that begin with "#" are considered comments
- http://goinbigdata.com/docker-run-vs-cmd-vs-entrypoint/
- StarDockerfiles must start with a FROM instruction
- FROM
    - https://docs.docker.com/engine/reference/builder/#from
    - Must be followed by the base image you wish to build from
- RUN
    - https://docs.docker.com/engine/reference/builder/#run
    - Executes any commands on the layer on top of the image
- CMD
    - https://docs.docker.com/engine/reference/builder/#cmd
    - Can only be one CMD per dockerfile
    - The main purpose of a CMD is to provide defaults for an executing container.
        - If it does not specify an executable, or there is no CMD, then an ENTRYPOINT must be defined
    - When used in the shell or exec formats, the CMD instruction sets the command to be executed when running the image.
- LABEL
    - https://docs.docker.com/engine/reference/builder/#label
    - The LABEL instruction adds metadata to an image. A LABEL is a key-value pair.
- EXPOSE
    - https://docs.docker.com/engine/reference/builder/#expose
    - The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime.
- ENV
    - https://docs.docker.com/engine/reference/builder/#env
    - Sets environment variables in the container
- ADD
    - https://docs.docker.com/engine/reference/builder/#add
    - The ADD instruction copies new files, directories or remote file URLs from <src> and adds them to the filesystem of the image at the path <dest>.

- COPY
  - https://docs.docker.com/engine/reference/builder/#copy
  - The COPY instruction copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.
  - COPY should always be used where the magic of ADD is not required.
- ENTRYPOINT
  - https://docs.docker.com/engine/reference/builder/#entrypoint
  - Allows you to configure a container that will run as an executable.
  - You can use the exec form of ENTRYPOINT to set fairly stable default commands and arguments and then use either form of CMD to set additional defaults that are more likely to be changed.
  - You can specify a plain string for the ENTRYPOINT and it will execute in /bin/sh -c. This form will use shell processing to substitute shell environment variables
  - Understand how CMD and ENTRYPOINT interact
- VOLUME
  - https://docs.docker.com/engine/reference/builder/#volume
  - The VOLUME instruction creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers.
- USER
  - https://docs.docker.com/engine/reference/builder/#user
  - Sets the user name (or UID) and optionally the user group (or GID) to use when running the image.
- WORKDIR
  - https://docs.docker.com/engine/reference/builder/#workdir
  - Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile.
  - Can be used multiple times in a dockerfile
  - The first time you use the command, the path should be absolute
  - If a relative path is passed, it will be relative to the last WORKDIR command
- ARG
  - https://docs.docker.com/engine/reference/builder/#arg
  - The ARG instruction defines a variable that users can pass at build-time to the builder with the docker build command using the --build-arg <varname>=<value> flag.
- ONBUILD
  - https://docs.docker.com/engine/reference/builder/#onbuild
  - The ONBUILD instruction adds to the image a trigger instruction to be executed at a later time, when the image is used as the base for another build.
- STOPSIGNAL
  - https://docs.docker.com/engine/reference/builder/#stopsignal
  - The STOPSIGNAL instruction sets the system call signal that will be sent to the container to exit.
- HEALTHCHECK

    - https://docs.docker.com/engine/reference/builder/#healthcheck
    - The HEALTHCHECK instruction tells Docker how to test a container to check that it is still working.
- SHELL
    - https://docs.docker.com/engine/reference/builder/#shell
    - The SHELL instruction allows the default shell used for the shell form of commands to be overridden.
- Best practices for writing Dockerfiles
    - https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/
    - Containers should be ephemeral
        - If one should be stopped and destroyed, a new one can be built and put in place with an absolute minimum of set-up and configuration.
    - Use a .dockerignore file
    - Use multi-stage builds
    - Avoid installing unnecessary packages
    - Each container should have only one concern
        - Decoupling applications into multiple containers makes it much easier to scale horizontally and reuse containers.
    - Minimize the number of layers
    - Sort multi-line arguments alphanumerically
    - Build cache
        - If you do not want to use the cache at all you can use the --no-cache=true option on the docker build command.
        - Otherwise Docker will use its cache to try and find a matching image

Helpful Links

- 1
    - https://wiki.alpinelinux.org/wiki/Alpine_Linux_package_management#Update_the_Package_list
    - https://github.com/klingtnet/docker-vim/blob/master/Dockerfile
- 2
    - http://forum.teamspeak.com/threads/132611-Install-teamspeak-3-server-on-Debian-8
    - https://github.com/GlThibault/Docker/blob/master/01_dockerfiles/ex01/Dockerfile
- 3
    - http://guides.rubyonrails.org/getting_started.html
    - https://github.com/GlThibault/Docker/blob/master/01_dockerfiles/ex02/Dockerfile
- 4
    - https://hub.docker.com/_/golang/
    - https://gogs.io/docs/installation/install_from_source
    - https://github.com/GlThibault/Docker/blob/master/01_dockerfiles/ex03/Dockerfile