Docker user guide:

Go get docker on: https://docs.docker.com/desktop/
Once the docker is successfully installed, you can use your cmd/terminal to check via command:

```
$ docker --version
```

And you will see,

```
weizhengs-MacBook-Pro:~ weizhengxu$ docker --version
Docker version 20.10.12, build e91ed57
```

Note that docker needs root privilege. To run docker as non-root user, you will need to add the user to the docker group as follows.

```
usermod -aG docker $USER
```

Then we can use docker to get specific docker image and setup on our machine via command:

```
$ docker image pull <image>
```

And you will see,

```
weizhengs-MacBook-Pro:~ weizhengxu$ docker image pull library/hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:975f4b14f326b05db86e16de00144f9c12257553bba9484fed41f9b6f2257800
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

Then you can check your installation with

```
$ docker image ls
```

And you will see,

```
weizhengs-MacBook-Pro:~ weizhengxu$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    feb5d9fea6a5   3 months ago   13.3kB
```

To run the image, we can use the command:

```
$ docker run XXX
```

And you will see,

```
weizhengs-MacBook-Pro:~ weizhengxu$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

%%%

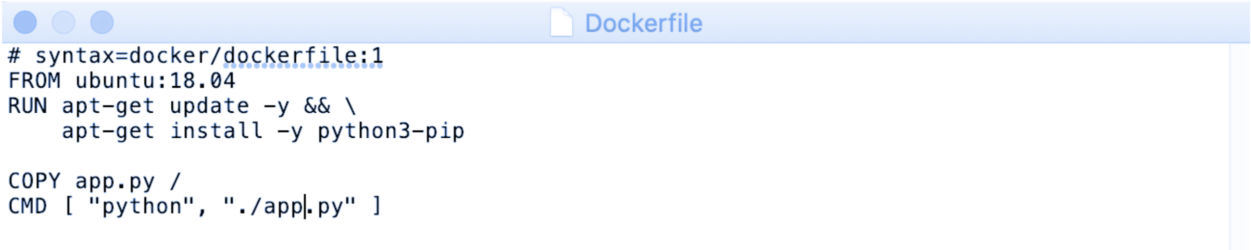
Tutorial on using Dockerfile for building images

Docker builds images automatically by reading the instructions from a `Dockerfile` -- a text file that contains all commands, in order, needed to build a given image. The referenced link is https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

1) Create a directory and go to the directory and create a file named Dockfile. The command used can be as below:

```
$ mkdir cs2510
$ cd cs2510
$ Touch Dockerfile
```

2) Edit the Dockfile



```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04
RUN apt-get update -y && \
    apt-get install -y python3-pip

COPY app.py /
CMD [ "python", "./app.py" ]
```

Some common commands to use:
FROM: This instruction specifies an existing image, and subsequent instructions will be performed based on this image. This image is called the base image. In this example case, we use ubuntu:18.04 as the base image.

COPY: It adds files from your Docker client's current directory.

RUN: It builds your application with make.

CMD: It specifies what command to run within the container.

ENV: To make new software easier to run, you can use ENV to update the PATH environment variable for the software your container installs.
E.g. `ENV PATH=/usr/local/postgres-$PG_MAJOR/bin:$PATH`

3) Build Docker based on Dockerfile

```
$docker image build -f Dockerfile -t cs2510_dk .
```

```
weizhengs-MacBook-Pro:cs2510 weizhengxu$ docker image build -f Dockerfile -t cs2
510_dk .
[+] Building 2.1s (14/14) FINISHED
 => [internal] load build definition from Dockerfile                      0.0s
 => => transferring dockerfile: 530B                                      0.0s
 => [internal] load .dockerignore                                         0.0s
 => => transferring context: 2B                                           0.0s
 => resolve image config for docker.io/docker/dockerfile:1                1.4s
 => [auth] docker/dockerfile:pull token for registry-1.docker.io          0.0s
 => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:42399d4635  0.0s
 => [internal] load build definition from Dockerfile                      0.0s
 => [internal] load .dockerignore                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:18.04           0.4s
 => [auth] library/ubuntu:pull token for registry-1.docker.io             0.0s
 => [1/3] FROM docker.io/library/ubuntu:18.04@sha256:37b7471c1945a2a12e5a  0.0s
 => [internal] load build context                                         0.0s
 => => transferring context: 351B                                         0.0s
 => CACHED [2/3] RUN apt-get update -y &&     apt-get install -y python3-  0.0s
 => CACHED [3/3] COPY app.py /                                            0.0s
 => exporting to image                                                    0.0s
 => => exporting layers                                                   0.0s
 => => writing image sha256:1a48af1fa40f69b538ccbc1db4d0cfe7f6c4bfaa44a23  0.0s
 => => naming to docker.io/library/cs2510_dk                              0.0s
weizhengs-MacBook-Pro:cs2510 weizhengxu$
```

4) Run the image using the command:
   `$sudo docker run cs2510_dk`

```
weizhengs-MacBook-Pro:cs2510 weizhengxu$ sudo docker run cs2510_dk
Hello World!
```

There is the cheat sheet to help you use docker:

Commands Cheat Sheet

docker

phoenixNAP GLOBAL IT SERVICES

Container Lifecycle

docker create [IMAGE]	create a container without starting it
docker rename [CONTAINER_NAME] [NEW_CONTAINER_NAME]	rename a container
docker run [IMAGE]	create and start a container
docker run --rm [IMAGE]	remove a container after it stops
docker run -td [IMAGE]	start a container and keep it running
docker run -it [IMAGE]	create, start the container, and run a command in it
docker run -it-rm [IMAGE]	create, start the container, and run a command in it; after executing, the container is removed
docker rm [CONTAINER]	delete a container if it isn't running
docker update [CONTAINER]	update the configuration of a container

Image Lifecycle

docker build [URL]	create an image from a Dockerfile
docker build -t [URL]	build an image from a Dockerfile and tags it
docker pull [IMAGE]	pull an image from a registry
docker push [IMAGE]	push an image to a registry
docker import [URL/FILE]	create an image from a tarball
docker commit [CONTAINER] [NEW_IMAGE_NAME]	create an image from a container
docker rmi [IMAGE]	remove an image
docker load [TAR_FILE/STDIN_FILE]	load an image from a tar archieve as stdin
docker save [IMAGE] > [TAR_FILE]	save an image to a tar archive stream to stdout with all parent layers, tags, and versions

Networking

docker network ls	list networks
docker network rm [NETWORK]	remove one or more networks
docker network inspect [NETWORK]	show information on one or more networks
docker network connect [NETWORK] [CONTAINER]	connect a container to a network
docker network disconnect [NETWORK] [CONTINAER]	disconnect a container from a network

Start & Stop

docker start [CONTAINER]	start a container
docker stop [CONTAINER]	stop a running container
docker restart [CONTAINER]	stop a running container and start it up again
docker pause [CONTAINER]	pause processes in a running container
docker unpause [CONTAINER]	unpause processes in a container
docker wait [CONTAINER]	block a container until other containers stop
docker kill [CONTAINER]	kill a container by sending SIGKILL to a running container
docker attach [CONTAINER]	attach local standard input, output, and error streams to a running container

Information

docker ps	list running containers
docker ps -a	list running and stopped containers
docker logs [CONTAINER]	list the logs from a running container
docker inspect [OBJECT_NAME/ID]	list low-level information on an object
docker events [CONTAINER]	list real time events from a container
docker port [CONTAINER]	show port (or specific) mapping from a container
docker top [CONTAINER]	show running processes in a container
docker stats [CONTAINER]	show live resource usage statistics of containers
docker diff [CONTINAER]	show changes to files (or directories) on a filesystem
docker images ls	show all locally stored images
docker history [IMAGE]	show history of an image