1. sudo docker run hello-world
   1. run = create + start
2. docker stores images in cache
3. default commands
   1. docker container run busybox ls – ls here is the default commands
      1. “container” here is the context, developers suggest this
   2. Default commands are commands that run right after a container is called
4. sudo docker container ls or sudo docker container ls -a
   1. command is the default command
   2. names are names of famous people
   3. ports are list of container ports mapped to the host
5. sudo docker create hello-world
   1. creates a hello-world (which is the image) container, but not running it
   2. it returns the id of the container that was just created
   3. status is now created, whereas for run (or start) the status was exited
6. sudo docker start <id of created container>
   1. now the status is exited
7. sudo docker system prune –all
   1. to clean all containers
8. sudo docker container logs <container id>
   1. simulate the logs of the container
9. sudo docker container stop <container id>
   1. docker gives about 10 seconds for the container to stop any processes before killing it
10. sudo docker container kill <container id>
    1. docker kills it immediately
11. sudo docker container rm <container id>
    1. use -f for force removal
12. sudo docker container inspect <container id>
    1. returns a json format data
13. sudo docker container run redis -> sudo docker container exec -it <container id> redis-cli or sudo docker container exec -it <container id> sh (for ssh)
    1. run a command in an already running container
    2. -it means interactive mode
14. Sudo docker container run -it <image name> sh
    1. start a docker container with shell
15. sudo docker container run -it –name jason alphine:latest /bin/sh
    1. apk add –update redis
    2. sudo docker container commit jason jason-image
    3. sudo docker image ls -a
    4. a manual way to create an image but a better way is to use Dockerfile
16. Create a Dockerfile:
    1. nano Dockerfile -> no extension
    2. keywords in dockerfile must be in upper case
    3. sudo docker build .
    4. sudo docker build -t haiya . -> -t for tag, here ‘haiya’ is the tag
    5. tar files are archive files that are used to store one or more files in a single location. It is commonly used in unix and linux os and often used to distribute large collections of files. TAR files are not compressed by default, but they can be using gzip to create a TAR.GZ file (aka tarball). TAR can be extracted and opened using software such as tar on unix or linux or with tools like 7zip or winzip on windows.
    6. Make a html file (for copy later), make a tar file
    7. Cd / means root directory
    8. Cd ~ means home directory
    9. Make a tar file:
       1. tar -cf a.tar a.txt b.txt
       2. tar -cf a.tar mydir (to create a tar file with a directory)
    10. FTP is for file transfer protocol and runs on port 21
    11. Ports are appended at the end of ip addresses, typically IPAddresses:Port\_Num
    12. Copy . /app
    13. Add <url or tar> /app
        1. Add automatically unpack compressed files (ie tarballs and zip files) and it can also download files from a URL and unpack them
17. docker run --name my-container -d -p 8080:80 my-image:latest
18. host machine -> vm -> docker container -> flask
    1. if flask receives a request on port 5001, it will send back a request to 5001
    2. a response from the docker container will be sent back so if the mapping was 80:90 then docker will send a response back to 80