**Docker Image**

We will try to make a web based app using python

1. We need a base container first
   1. $docker run -it ubuntu bash
2. We update all its packages
   1. $apt-get update
3. Now we install python
   1. $apt-get install -y python3
4. Test that python is properly installed
   1. $python3
   2. exit()
5. Install pip
   1. $apt-get install python3-pip -y
6. Verify pip
   1. $pip3 –version
7. Install flask
   1. $pip3 install flask
8. Copy the python code(at the end of this document) into /opt/app.py
   1. $cat >/opt/app.py
9. Make sure that the above code is properly indented.
10. Go to the /opt folder and verify that the app.py exists
    1. $cd /opt
    2. $cat app.py
11. Run the flask
    1. $FLASK\_APP=app.py flask run --host=0.0.0.0
12. Check the browser for output
    1. On firefox open 172.17.0.2:5000
    2. 172.17.0.2:5000/how are you
    3. The IP above can be different on your machine. Use the IP as per your machine output
13. Stop the server in the terminal with CTL-c
14. Find all commands used
    1. $history
15. On a test editor on docker host, list all the commands which were used
16. We will have something like this
    1. apt-get update
    2. apt-get install -y python3
    3. python3
    4. apt-get install python3-pip -y
    5. pip3 install flask
    6. cat >/opt/app.py
    7. cd /opt/
    8. cat app.py
    9. FLASK\_APP=app.py flask run --host=0.0.0.0
17. Eliminate the non essential commands
    1. apt-get update
    2. apt-get install -y python3
    3. apt-get install python3-pip -y
    4. pip3 install flask
    5. cat >/opt/app.py
    6. FLASK\_APP=app.py flask run --host=0.0.0.0
18. Add actions before each of these lines
    1. FROM ubuntu
    2. RUN apt-get update
    3. RUN apt-get install -y python3
    4. RUN apt-get install python3-pip -y
    5. RUN pip3 install flask
    6. COPY app.py /opt/app.py
    7. ENTRYPOINT FLASK\_APP=/opt/app.py flask run --host=0.0.0.0
19. The first line is always a FROM statement. This denotes the base image. All docker images are based on some base image.
20. ENTRYPOINT signifies the command or operation which starts the application. This is also added as metadata and not an image layer.
21. It is not necessary to capitalize the first instruction but is a standard practice.
22. Create a working directory in the docker host
    1. $mkdir myPyApp
23. Save the file above in that directory with name “Dockerfile”.
24. Copy the app.py(code at the end of this file) into that directory
25. cd to that directory
    1. $cd myPyApp
    2. $ls
    3. Ensure that this contains 2 files, Dockerfile and app.py
26. The directory containing the application and dependencies is referred to as the build context
27. From the current location build the image
    1. $docker build . -t jkjha/myapp
28. Check if this image is build
    1. $docker image ls
29. Login with your docker account. If you do not have one, create it on dockerhub
    1. $docker login
    2. Password will be requested. Provide the same
30. Push your image to the repository
    1. $docker push jkjha/myApp
31. Verify on docker hub if your image has been successfully pushed

app.py

import os

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def main():

return "Welcome!"

@app.route('/how are you')

def hello():

return 'I am good, how about you?'

if \_\_name\_\_ == "\_\_main\_\_":

app.run()