# Container Image Management

mkdir node-app

cd node-app

vi app.js

const http = require('http');

const hostname = '0.0.0.0';

const port = 80;

const server = http.createServer((req, res) => {

    res.statusCode = 200;

      res.setHeader('Content-Type', 'text/plain');

        res.end('Hello Docker Chief\n');

});

server.listen(port, hostname, () => {

    console.log('Server running at http://%s:%s/', hostname, port);

});

process.on('SIGINT', function() {

    console.log('Caught interrupt signal and will exit');

    process.exit();

});

# save the file with :wq

vi Dockerfile

# Use an official Node runtime as the parent image

FROM node:6

# Set the working directory in the container to /app

WORKDIR /app

# Copy the current directory contents into the container at /app

ADD . /app

# Make the container's port 80 available to the outside world

EXPOSE 80

# Run app.js using node when the container launches

CMD ["node", "app.js"]

# save the file with :wq

## Create a Container Image

docker build -t node-app:0.1 -f Dockerfile .

docker images

docker images | grep node-app

## Creating Version 0.2

Make changes in app.js

        res.end('Hello Docker Chief – Day 02\n');

docker build -t node-app:0.2 -f Dockerfile .

docker images | grep node-app

docker run --name node-app1 -dt -p 85:80 node-app:0.1

docker run --name node-app2 -dt -p 87:80 node-app:0.2

docker ps

curl http://localhost:85

curl http://localhost:87

docker logs node-app1

docker logs node-app02

## Tagging Images

docker tag node-app:0.1 <docker-hub-username>/node-app:0.1

docker images

docker push <docker-hub-username>/node-app:0.1

Above step will fail, without a Login

docker login

docker push <docker-hub-username>/node-app:0.1

docker tag node-app:0.2 <docker-hub-username>/node-app:0.2

docker push <docker-hub-username>/node-app:0.2

## Python Application:

mkdir python-app

cd python-app

vi app.py

from flask import Flask

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

@app.route("/")

def hello():

return "welcome to Simplilearn!! successfully done !!"

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

app.run()

vi requirement.txt

Flask

vi Dockerfile

*FROM python*

*WORKDIR /app*

*ADD . /app*

*RUN pip install -r requirements.txt*

*EXPOSE 80*

*CMD [“python”, “app.py”]*

### Multistage Build

vi Dockerfile

FROM alpine AS stage1  
LABEL maintainer="Collabnix"  
RUN echo "Welcome to Docker Labs!" > /opt/index.html

FROM nginx:alpine  
COPY --from=stage1 /opt/index.html /usr/share/nginx/html/  
ENTRYPOINT ["nginx", "-g", "daemon off;"]

Note:

RUN actually runs a command and commits the result;

CMD does not execute anything at build time