Yuwen Liu

11219371

yul905

Part A

1. pull: Download docker images from a registry EX: using `docker pull python` to download *python* to local mechine from the registry
2. build: build docker images from a Dockerfile and a context(set of file located in the specifidpath or url).EX: `docker build -t myimage .` can creat an image named 'myimage' by dockerfile in corrent path
3. run: create a writeable container layer over the specified image and then starts it using the specified command.EX: `docker run --rm myinage` the comment will create a container base on myimage and reomve the comtainer after runnng
4. ps: list the running containers EX: `docker ps -a` the comment will show all container whatever is running or unrunning
5. stop: Stop one or more running containers: EX: `docker stop -t 20 container` the comment will stop the container1 after 10s

Part B

```
FROM python:latest

EXPOSE 8080

RUN mkdir /usr/src/app

WORKDIR /usr/src/app

CMD ["/bin/bash"]
```

Part C

```
'use strict';

// load package
const express = require('express');
const bodyParser = require("body-parser");
const fs = require('fs');


const PORT = 8080;
const HOST = '0.0.0.0';
```

```
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

app.post('/save', (req, res) => {
  var fileanme = 'feedback.txt'

  // debugger
  const {topic, name, comment} = req.body;
  // TODO write to file (maybe using fs)

  fs.writeFileSync('./' + fileanme, topic + ', ' + name + ', ' + comment + '\n'
    if (err) {
      console.error(err)
      return
    }
    res.send('ok');
  });

  res.send('done');
});

app.use('/', express.static('./'));


app.listen(PORT, HOST);

console.log(`Now listening on http://${HOST}:${PORT}`);
```