PART A

i.) **Docker Pull: -** The Docker pull is a cmd used to download a docker image or repository locally on the host from a public/private registry. Because when we run any container and the docker image is not locally on the machine it first pulls it from the registry. Most images are downloaded from hub.docker.com and that is when we create a custom docker image as we use the main docker image as a base image. An example of a docker pull command is:

a. docker pull python
b. docker pull ubuntu
c. docker pull reactnativecommunity/react-native-android.
d. Docker pull ghost
e. Docker pull openjdk
f. Docker pull myimage:1.0

ii.) **Docker build: -** The Docker build is a cmd used to build an image from the docker file in the current dir and then able to tag the image. The docker build can refer to any files located in that specified Uniform Resource Locator (URL) or in the path. An example of a docker build command is:

a. docker build -t myimage1:1.0 .
b. docker build - < Dockerfile
c. docker build . [that's If you are already in the directory of your file]
d. docker build http://myrepo.git

iii.) **Docker run: -** The Docker run is a cmd first creates a writeable container layer over the specified image and it then starts using the specified command. Docker run can also be used with a combination of docker commit to change the command that a container runs.

Example to run a container Alpine version 4.0 image using the running container web and then expose Port 8080 externally mapped to Port 80 inside the container.

a. Docker container run –name web -p 8080:80 alpine:4.0
b. Docker run -w /user/francis/dir/ -I -t ubuntu pwd (is to set working directory)
c. Docker run –p 127.0.1:80:8080/tcp ubuntu bash [publish port]
d. Docker run –expose 8080 ubuntu bash [expose port]

PART B

CODING ENVIRONMENT VERSION

```
FROM python:latest
WORKDIR /usr/src/app
```

CMPT 353 ASSIGNMENT-ONE

```
COPY . .
EXPOSE 8080
CMD ["bin/bash"]
```

TEXT CODE VERSION
FROM python:latest
WORKDIR /usr/src/app
COPY . .
EXPOSE 8080
CMD [“bin/bash”]

PART C

CODING ENVIRONMENT VERSION

```
const express = require("express");
const bodyParser = require("body-parser");
const {writeFile} = require("fs");
const app = express();
const port = 8080;
const host = '0.0.0.0';

app.use(bodyParser.urlencoded({extended: true}));
app.post("/", (req, res) => {
    const topic = req.body.topic, name = req.body.name, comment =
req.body.comment;
    writeFile('feedback.txt', `${topic}, ${name}, ${comment}`,  {flag: 'a+'},
err => {
        if (!err) {
            res.send("Completed.")
        } else {
            console.error(err)


        }

    });
});
app.listen(port, host);
console.log("Executed correctly...")
```

Text CODE VERSION
const express = require(“express”);
const bodyParser = require(“body-parse”);
const{writeFile} = require(“fs”);

CMPT 353 ASSIGNMENT-ONE

```
const app = express();
const port = 8080;
const host = ‘0.0.0.0’;

app.use(bodyParser.urlencoded({extended: true}));
app.post(“/”, req, res) => {
     const topic = req.body.topic, name = req.body.name, comment = req.body.comment;
writeFile(‘feedback.txt, ${topic}, `&{name}, &{comment}` {flag: ‘a+’}),
err => {
        if (!err) {
           res.send(“Completed.”)
        } else{
        Console.error(err)

     }

  });
});
app.listen(port, host);
console.log(“Executed correctly…”);
```