Week2 Project Review

Task 1: Microservice1: data service development

Code, Configure, Build, and Run DataApi

Go to DataApi Project Directory

a. Add, edit code and configuration, and save all resources

b. Open a terminal, go to DataApi project directory. Run gradle bootjar to create application jar file

<DataApi_project_directory> **gradle bootjar**

c. Run the app using java command

<DataApi_project_directory**> java -jar ./build/libs/mccdataapi.jar**

Task 2: Microservice2: auth service (security service) development

Code, Configure, Build, and Run AuthApi

Go to AuthApi Project Directory

a. Add, edit code and configuration, and save all resources

b. Open a terminal, go to AuthApi project directory. Run gradle bootjar to create application jar file

<AuthApi_project_directory> **gradle bootjar**

c. Run the app using java command

<AuthApi_project_directory> **java -jar ./build/libs/mccauthapi.jar**

Task 3: Testing service integration (AuthApi <--> DataApi) and service endpoints

1. Start postman

2. Test data api service end points without security token

-- GET request http://localhost:8080/api/customers

This should fail. Verify.

What is the error message?

3. Test auth api service end point to get a token to login

-- POST request http://localhost:8081/auth/token

-- payload (body) must be valid name and password value pairs

{“name”: "Bruce", "password" : "pass"}

4. See the response and copy the JWT token string

5. Use the JWT token to test data api service end points

-- GET request

-- http://localhost:8080/api/customers

-- Header: Authorization: Bearer, Token:<JWT Token string>

6. What is the result?

7. Test other data api end points

8. Test auth api service end point to register a new customer

-- POST request

-- http://localhost:8081/auth/register

-- payload (body) must be a new customer data

{“name":"fox", "email":"fox@webage.com" "password":"foxy"}

what is the response? 201 created? OR error?

9. Test the data api service end points to view the list of customers

-- GET request

-- http://localhost:8080/api/customers

10. Run the react app client and test client-service integration

-- go to Downloads/project/ReactClients/day8

-- unzip project-react-client.zip

-- Open a terminal, and cd to project-react-client

-- run npm command to install node packages (modules)

- npm install

-- run npm command to start development server

- npm start

-- Open a browser to load react app and test the app

- http://localhost:3000

-- test UI pages

Task 4: Build docker images for the services

project-data-api-dir>**docker build --tag dataapi:v1.0 .**

project-auth-api-dir>**docker build --tag authapi:v1.0 .**

docker images

Task 5. Create a docker network, run docker containers and add containers to network

1. Create a docker network

docker network create mccnetwork

2. Start the data api container

docker run -d --name api -p 8080:8080 dataapi:v1.0

3. Add the data api container to your docker network

docker network connect mccnetwork api

4. Inspect the docker network and find the ip address of data api

docker network inspect mccnetwork

You will get some message like:

"Containers": {

"6dd5dffbe8275fbd01d26cc4247fc17263759d27cce66c58a4f4cd648ae29346": {

"Name": "api",

"EndpointID":
"781744572b8df6e58d0174bce7da8a68d96f60448b30496f50b3c1b6f7528a0d",

"MacAddress": "02:42:ac:13:00:02",

"**IPv4Address": "172.18.0.2/16",**

"IPv6Address": ""

}

},

See the ip address of the container api. In this case **IPv4Address: 172.18.0.2**

use the IP address of api container as an env variable value to auth service docker container

5. How to configure auth container to use an env variable and get the value in code?

5.a Configuration

auth api Dockerfile Instructions - add an ARG and ENV variable

---

ARG targethost=localhost:8080

ENV API_HOST=$targethost

Dockerfile

```
FROM gradle:jdk10 as builder
COPY --chown=gradle:gradle . /app
WORKDIR /app
RUN gradle bootJar

FROM openjdk:8-jdk-alpine
EXPOSE 8081
VOLUME /tmp
ARG targethost=localhost:8080
ENV API_HOST=$targethost
ARG LIBS=app/build/libs
COPY --from=builder ${LIBS}/ /app/lib
ENTRYPOINT ["java","-jar","./app/lib/project-auth-api-0.0.1-SNAPSHOT.jar"]
```

5.b Code

in the auth service code, read the variable value with a code

String apiHost = System.getenv("API_HOST");

String apiURL = "http://" + apiHost + "/api/customers";

//use the variable apiURL to connect to data Api

---

6. **Rebuild the auth service container image**

project-auth-api-dir>**docker build --tag authapi:v1.0 .**

7. Start the auth service container, pass the env variable name and value pair

project-auth-api-dir>**docker run -d --name=auth -p 8081:8081 --env API_HOST=172.18.0.2:8080 authapi:v1.0**

8. Add the auth service container to docker network

docker network connect mccnetwork auth

9. Inspect the docker network and find the ip addresses of data api and auth service

you will see two containers with ip addresses


```
"Containers": {
        "6dd5dffbe8275fbd01d26cc4247fc17263759d27cce66c58a4f4cd648ae29346": {
            "Name": "api",
            "EndpointID":
"781744572b8df6e58d0174bce7da8a68d96f60448b30496f50b3c1b6f7528a0d",
            "MacAddress": "02:42:ac:13:00:02",
            "IPv4Address": "172.18.0.2/16",
            "IPv6Address": ""
        },
        "dc30fed0722185d432ea138d4f6b264b6dea2303286e0210d0540d1a59c6696d": {
```

```
            "Name": "auth",
            "EndpointID":
"a7c13bdc55b0d3a670f8b2af1dc42de099915081ba8a07294001f141e36751e1",
            "MacAddress": "02:42:ac:13:00:03",
            "IPv4Address": "172.18.0.3/16",
            "IPv6Address": ""
        }
```

Task 6. Test service end points

How to integrate front end client with auth docker container and api docker container?

Our docker containers addresses are:

api container address: 172.18.0.2:8080

auth container address: 172.18.0.3:8081

1. Go to client project. You must modify two files in the client project,

to forward the requests to ip addresses of auth and api in docker network.

clientprojectdir>**/src/setupproxy.js**

clientprojectdir> **default.conf**

2. Edit /src/setuproxy.js file.

the default host for both /api and /account has been configured as localhost

Replace the localhost with ip addresses of api and auth container.

```
/*   app.use(proxy('/api', { target: 'http://localhost:8080/' }));
   app.use(proxy('/account', { target: 'http://localhost:8081/' }));*/

   app.use(proxy('/api', { target: 'http://172.18.0.2:8080/' }));
   app.use(proxy('/account', { target: 'http://172.18.0.3:8081/' }));
```

Save the file.

---

3. Edit the default.conf file.

Modify the ip addresses of proxy location of /api and /account

```
# OPTIONAL: For an API server you want to proxy
 location /api {
  # proxy_pass http://10.111.48.90:8080;
   proxy_pass http://172.18.0.2:8080;
   proxy_http_version 1.1;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection ‘upgrade’;
   proxy_set_header Host $host;
   proxy_cache_bypass $http_upgrade;
 }
 location /account {
  # proxy_pass http:///10.104.111.92:8081;
   proxy_pass http://172.18.0.3:8081;
   proxy_http_version 1.1;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection ‘upgrade’;
   proxy_set_header Host $host;
   proxy_cache_bypass $http_upgrade;
 }
```

---

4. Open a terminal, cd to client project directory

5. Open the Dockerfile and review, you do not have to do any changes here.

6. Build a docker image for the client

osboxes@osboxes:~/Desktop/demomccproject/project-react-client$ **docker build --tag reactclient:v1.0 .**

7. Run a docker container using the reactclient:v1.0 image

osboxes@osboxes:~/Desktop/demomccproject/project-react-client$ **docker run --name react -it -p 3000:80 reactclient:v1.0**

this will start the container.

8. Open another terminal. Add the react client container to existing docker network, that is mccnetwork

osboxes@osboxes:~$ **docker network connect mccnetwork react**

9. Inspect the network.

osboxes@osboxes:~$ **docker network inspect mccnetwork**

You will see three ip addresses, one for api, one for auth and one for react.

"Containers": {

"0ebfc7973872c7e1e087f1898c71cbf0fbe806c249ff11d977dfd0b51a346dcb": {

"Name": "react",

"EndpointID": "21decfd7e90806653ec04ef6261a461cd1d88170fb618253391ba2d599bccc08",

"MacAddress": "02:42:ac:12:00:04",

"**IPv4Address": "172.18.0.4**/16",

"IPv6Address": ""

},

"24ea8471b8ef81f7fd5d921f8734c13354768e05eab2255a72fe5dcec4d9d3f0": {

"Name": "api",

"EndpointID": "fc5eb89b905760f576c38f489cfaf25abfc29e9d678cc7c7a057d1429a4821e1",

"MacAddress": "02:42:ac:12:00:02",

"**IPv4Address": "172.18.0.2**/16",

```
            "IPv6Address": ""
        },
        "643be31558b86e792c05e5573e528bcb77c3bb87a69745dc7d4728bf6bbf4b88": {
            "Name": "auth",
            "EndpointID":
"b4503ac9fab5db39a1bcdd9ec703b2476c3eaea33bf836470fbb3953726cff26",
            "MacAddress": "02:42:ac:12:00:03",
            "IPv4Address": "172.18.0.3/16",
            "IPv6Address": ""
        }
    }
```


10. Run a command docker ps -a

osboxes@osboxes:~$ **docker ps -a**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

0ebfc7973872 reactclient:v1.0 "nginx -g 'daemon of…" 28 minutes ago Up 28 minutes 0.0.0.0:3000->80/tcp react

643be31558b8 authapi:v1.0 "java -jar ./app/lib…" 44 minutes ago Up 44 minutes 8080/tcp, 0.0.0.0:8081->8081/tcp auth

24ea8471b8ef dataapi:v1.0 "java -jar ./app/lib…" 46 minutes ago Up 46 minutes 0.0.0.0:8080->8080/tcp api

Notice the PORTS column

reactclient 0.0.0.0:3000

authapi 0.0.0.0:8081

dataapi 0.0.0.0:8080

use these addresses in postman, and in browser to test service end points.

11. Open a browser, go to react app using the new tcp address

http://0.0.0.0:3000

test login and register requests.

12. Open postman, Test the end points using the new tcp addresses

http://0.0.0.0:8081/account/token

http://0.0.0.0:8081/account/register

http://0.0.0.0:8080/api/customers