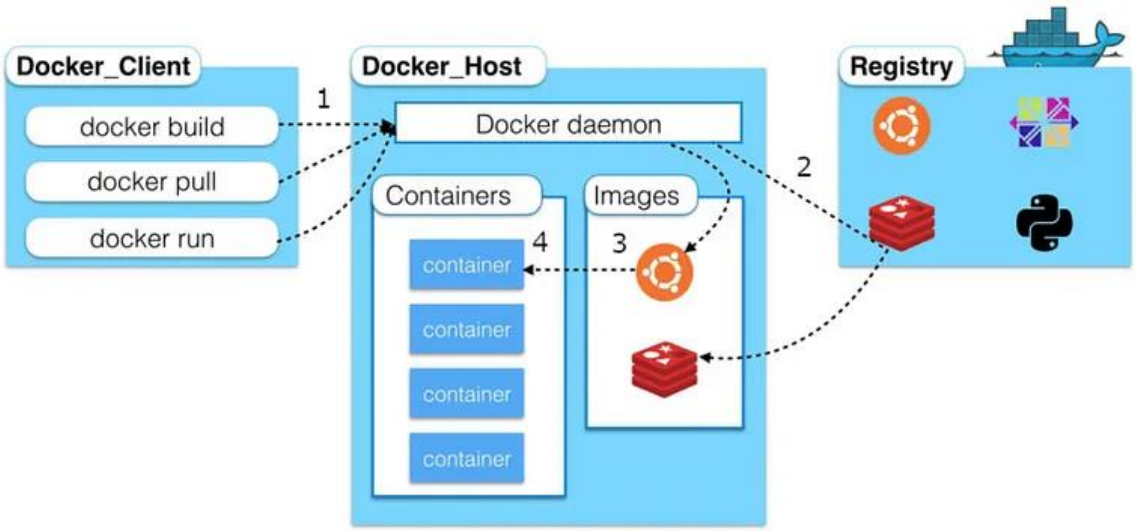
LAB5 - Docker implementation

66543210040-2 Prasert Sooksrikaew
66543210038-6 Chinchettha Namayom



วิชา สถาปัตยกรรมซอฟต์แวร์
Software Architecture

Lab5 นี้ อ้างอิงจาก
https://medium.com/@JeffyJeff/the-beginners-guide-to-docker-fa4c4d3181e7

Codeshare ของ Lab5
https://codeshare.io/DA3XpO

Pre-requisite (สิ่งต้องมีก่อน)

ติดตั้ง Ubuntu Server ใน VM จะเป็น Version 20.04 หรือ 22.04 ก็ได้

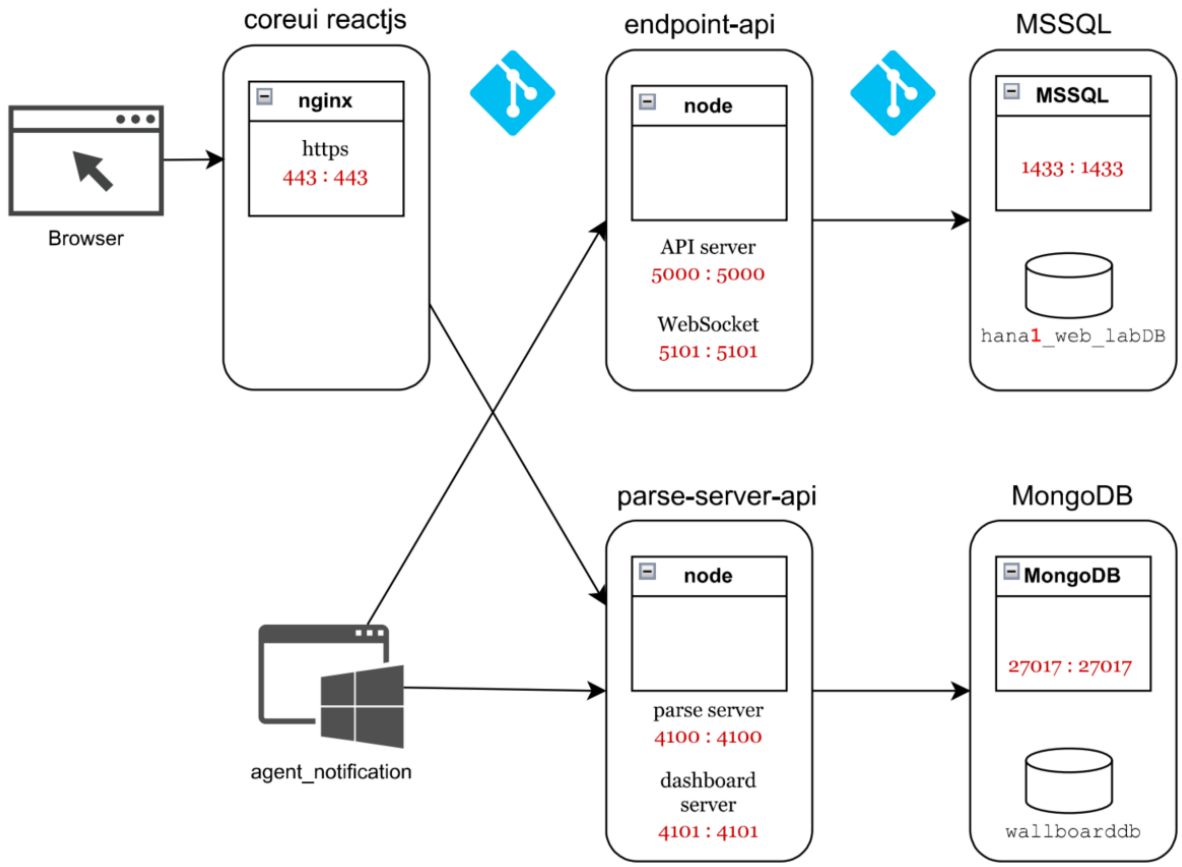
ถ้ากำลังลงใหม้ให้สร้าง user ตามนี้ตอนติดตั้ง หรือ ถ้าใครสร้าง VM แล้ว ให้ทำการ remote login โดย Terminus (download ที่นี่)

Docker Host		
Docker Container		
Service	**DNS name**	**Container map port**
coreui reactjs	coreui, webreport.com	443:443
endpoint-api	endpoint-api	5000:5000, 5101:5101
parse-server-api	parse-server-api	4100:4100, 4101:4101
mssql	mssql	1433:1433
mongodb	mongodb	27017:27017

ให้ทำการ clone repositoty นี้ลงมา (Branch: main)

```
Git clone git@github.com:prasertsook/engse207-hana.git
```

ภาพสำเร็จ ทีต้องทำส่ง



Final Task

ให้แต่ละTeam ส่งงาน final ผมใน slack โดยส่งเป็น direct message มานะครับ กำหนดส่งเป็น ลิ้ง final git repository โดยทำการสร้าง repository สำหรับ final ขึ้นมาใหม่ ในนั้นจะประกอบไปด้วย

1. มี Readme.md สำหรับบอกชื่อกลุ่มและสมาชิกของกลุ่ม

2.ไฟล์เอกสาร PDF อธิบายงาน

3. Directory สำหรับเก็บไฟล์ script และ source code ที่บรรจุลงไปในแต่ละคอนเทนเนอร์

4. มี link video ทดสอบการใช้งานของงาน อยู่ในไฟล์ Readme.md ด้วยครับ

วิธีการเขียน Dockerfile เบื้องต้น

https://codinggun.com/docker/dockerfile/

Container Image
https://www.saladpuk.com/basic/docker-1/images

Dockerfile และการใช้งาน

https://devhub.in.th/learn/docker/dockerfile

easy-create-docker-image-for-express-and-nodejs
https://nextflow.in.th/2017/easy-create-docker-image-for-express-and-nodejs/

เทคนิคการเขียน Dockerfile ให้ดีขึ้นกว่าเดิม

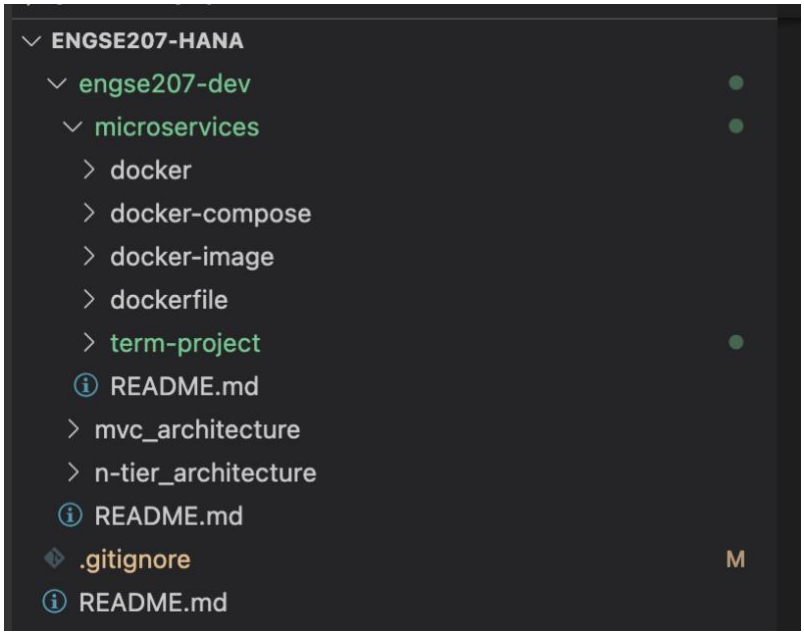
https://igokuz.com/%E0%B9%80%E0%B8%97%E0%B8%84%E0%B8%99%E0%B8%B4%E0%B8%84%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%80%E0%B8%82%E0%B8%B5%E0%B8%A2%E0%B8%99-dockerfile-%E0%B9%83%E0%B8%AB%E0%B9%89%E0%B8%94%E0%B8%B5%E0%B8%82%E0%B8%B6%E0%B9%89%E0%B8%99%E0%B8%81%E0%B8%A7%E0%B9%88%E0%B8%B2%E0%B9%80%E0%B8%94%E0%B8%B4%E0%B8%A1-54407fae9edc

Dockerfile tutorial by example - basics and best practices
https://takacsmark.com/dockerfile-tutorial-by-example-dockerfile-best-practices-2018/

Dockerfile Implementation

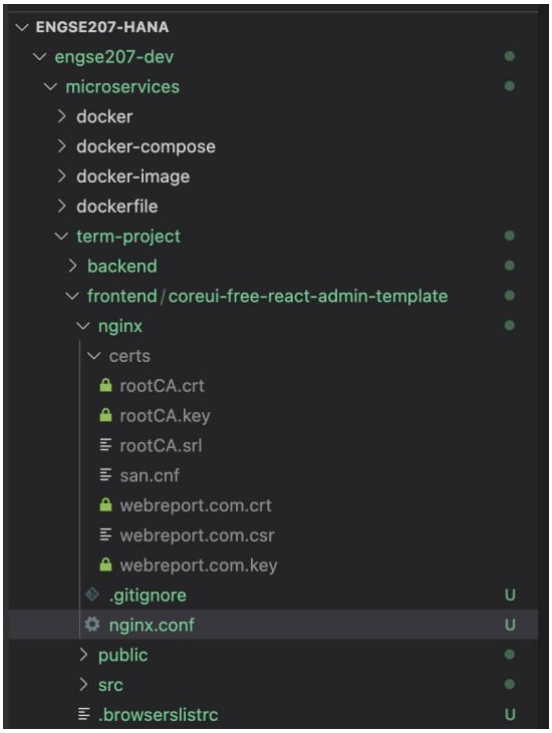
Frontend → coreui reactjs

สร้าง directory term-project สำหรับเก็บ source code program



สร้าง directory frontend ภายใต้ microservices และ copy source code coreui-free-react-admin-template มาไว้ใน frontend และสร้าง directory nginx สำหรับเอาไว้เก็บ certs และ config

สร้าง certs file



```
openssl genrsa -des3 -out rootCA.key 4096
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.crt
openssl genrsa -out webreport.com.key 2048

san.cnf
[ req ]
default_bits       = 2048
prompt             = no
default_md         = sha256
req_extensions     = req_ext
distinguished_name  = req_distinguished_name

[ req_distinguished_name ]
countryName                = US
stateOrProvinceName        = California
localityName               = San Francisco
organizationName           = Internet Widgets Pty Ltd
commonName                 = webreport.com

[ req_ext ]
subjectAltName = @alt_names

[ alt_names ]
DNS.1   = webreport.com
DNS.2   = www.webreport.com


openssl req -new -key webreport.com.key -out webreport.com.csr -config san.cnf
openssl x509 -req -in webreport.com.csr -CA rootCA.crt -CAkey rootCA.key -
CAcreateserial -out webreport.com.crt -days 500 -sha256 -extfile san.cnf -extensions
req_ext

openssl req -in webreport.com.csr -noout -text
openssl x509 -in webreport.com.crt -text -noout

Add a root CA certificate to MacOS
sudo security add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.keychain
rootCA.crt
```

สร้าง nginx.conf

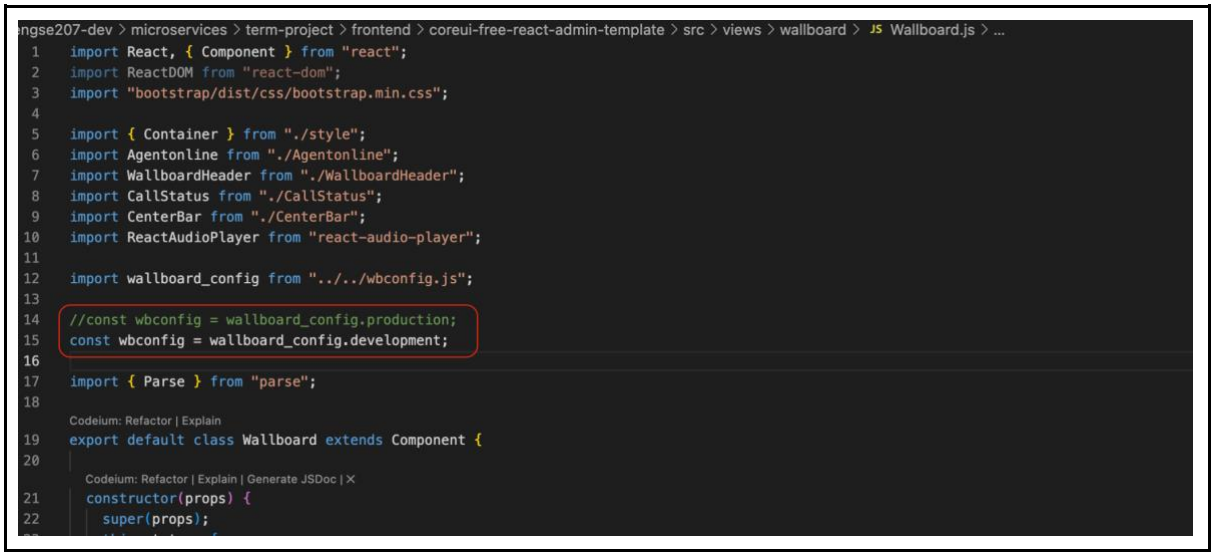
```
# /etc/nginx/nginx.conf

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    # Server block for HTTPS
    server {
        listen 443 ssl;
        server_name webreport.com www.webreport.com;

        ssl_certificate /etc/nginx/ssl/webreport.com.crt;
        ssl_certificate_key /etc/nginx/ssl/webreport.com.key;

        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;

        root /usr/share/nginx/html;
        index index.html;

        location / {
            try_files $uri $uri/ =404;
        }
    }

    # Redirect HTTP to HTTPS
    server {
        listen 80;
        server_name webreport.com www.webreport.com;
        return 301 https://$host$request_uri;
    }
}

```

แก้ไข wbconfig.js เพื่อให้ชี้ไปที่ parse-server-api

engse207-dev > microservices > term-project > frontend > coreui-free-react-admin-template > src > JS wbconfig.js > ...

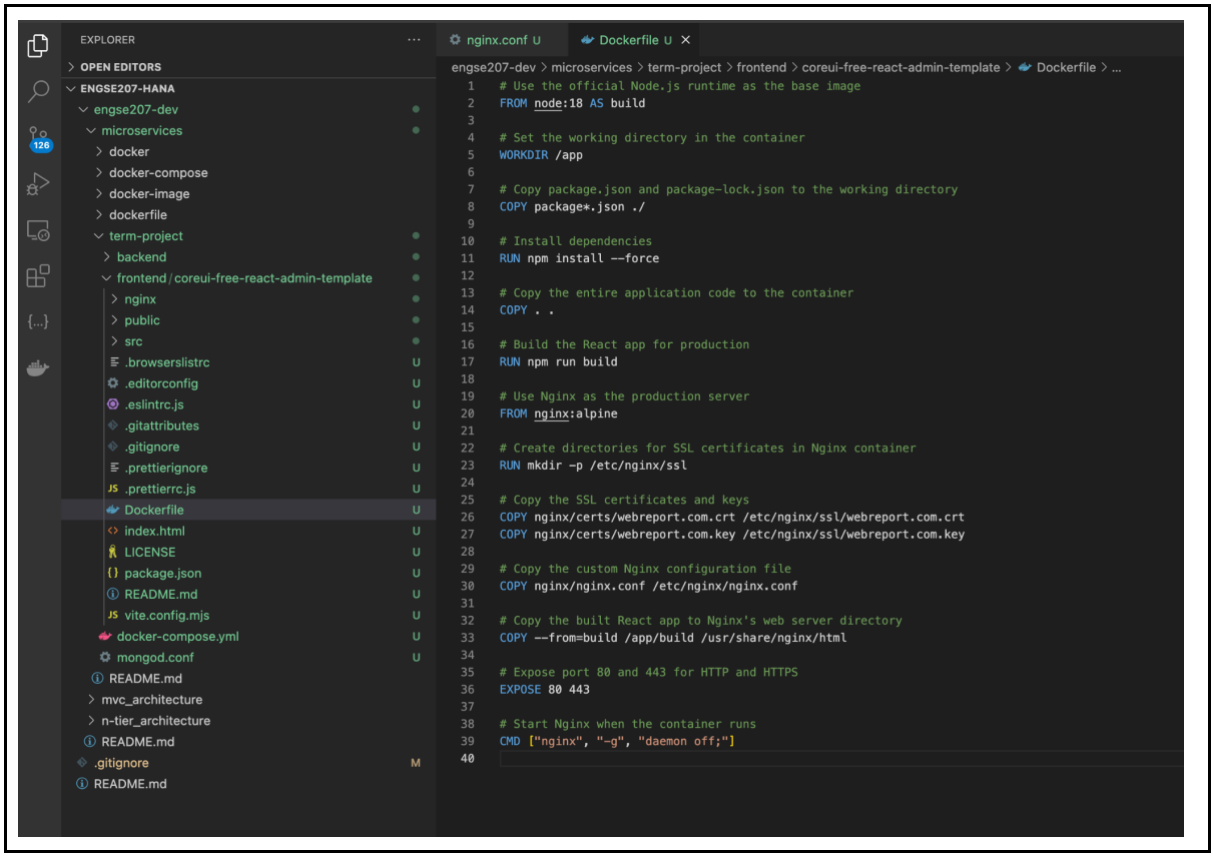
```
const wbconfig = {
    development: {
        hosturl  : 'https://parse-server-api:4100/api',
        wsurl    : 'wss://parse-server-api:4100',
        masterKey     : 'wallboardapi',
        clientKey     : 'wallboardapi',
        javascriptKey : 'wallboardapi',
        appId : "wallboardapi"
    },
    production: {
        hosturl  : 'https://lab-parse-server.se-rmutl.net/api',
        wsurl    : 'wss://lab-parse-server.se-rmutl.net',
        masterKey     : 'wallboardapi',
        clientKey     : 'wallboardapi',
        javascriptKey : 'wallboardapi',
        appId : "wallboardapi"
    }
};
export default wbconfig;

```

แก้ไข Wallboard.js เพื่อให้มาดึง config ในส่วนของ development



```
engse207-dev > microservices > term-project > frontend > coreui-free-react-admin-template > src > views > wallboard > JS Wallboard.js > ...
1   import React, { Component } from "react";
2   import ReactDOM from "react-dom";
3   import "bootstrap/dist/css/bootstrap.min.css";
4
5   import { Container } from "./style";
6   import Agentonline from "./Agentonline";
7   import WallboardHeader from "./WallboardHeader";
8   import CallStatus from "./CallStatus";
9   import CenterBar from "./CenterBar";
10  import ReactAudioPlayer from "react-audio-player";
11
12  import wallboard_config from "../../wbconfig.js";
13
14  //const wbconfig = wallboard_config.production;
15  const wbconfig = wallboard_config.development;
16
17  import { Parse } from "parse";
18
    Codeium: Refactor | Explain
19  export default class Wallboard extends Component {
20
      Codeium: Refactor | Explain | Generate JSDoc | ×
21    constructor(props) {
22      super(props);
```

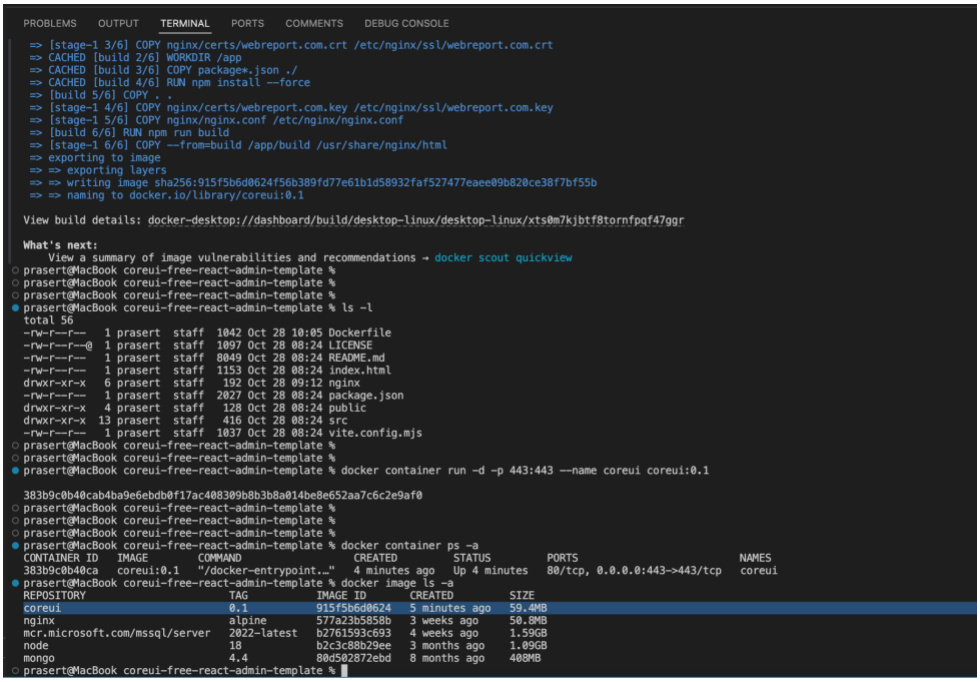
สร้าง Script Dockerfile สำหรับ build image Nginx กับ ssl และ web app coreui



```
EXPLORER
OPEN EDITORS
ENGSE207-HANA
  engse207-dev
    microservices
      docker
      docker-compose
      docker-image
      dockerfile
      term-project
        backend
        frontend/coreui-free-react-admin-template
          nginx
          public
          src
          .browserslistrc U
          .editorconfig U
          .eslintrc.js U
          .gitattributes U
          .gitignore U
          .prettierignore U
          .prettierrc.js U
          Dockerfile U
          index.html U
          LICENSE U
          package.json U
          README.md U
          vite.config.mjs U
        docker-compose.yml U
        mongod.conf U
      README.md
      mvc_architecture
      n-tier_architecture
    README.md
    .gitignore M
  README.md

nginx.conf U   Dockerfile U ×
engse207-dev > microservices > term-project > frontend > coreui-free-react-admin-template > Dockerfile > ...
1   # Use the official Node.js runtime as the base image
2   FROM node:18 AS build
3
4   # Set the working directory in the container
5   WORKDIR /app
6
7   # Copy package.json and package-lock.json to the working directory
8   COPY package*.json ./
9
10  # Install dependencies
11  RUN npm install --force
12
13  # Copy the entire application code to the container
14  COPY . .
15
16  # Build the React app for production
17  RUN npm run build
18
19  # Use Nginx as the production server
20  FROM nginx:alpine
21
22  # Create directories for SSL certificates in Nginx container
23  RUN mkdir -p /etc/nginx/ssl
24
25  # Copy the SSL certificates and keys
26  COPY nginx/certs/webreport.com.crt /etc/nginx/ssl/webreport.com.crt
27  COPY nginx/certs/webreport.com.key /etc/nginx/ssl/webreport.com.key
28
29  # Copy the custom Nginx configuration file
30  COPY nginx/nginx.conf /etc/nginx/nginx.conf
31
32  # Copy the built React app to Nginx's web server directory
33  COPY --from=build /app/build /usr/share/nginx/html
34
35  # Expose port 80 and 443 for HTTP and HTTPS
36  EXPOSE 80 443
37
38  # Start Nginx when the container runs
39  CMD ["nginx", "-g", "daemon off;"]
40
```

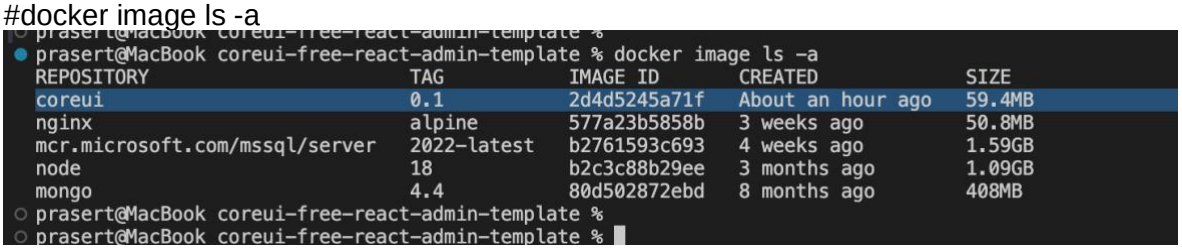
จากนั้นรัน command เพื่อ build image สำหรับ coreui

#docker build -t coreui:0.1 .



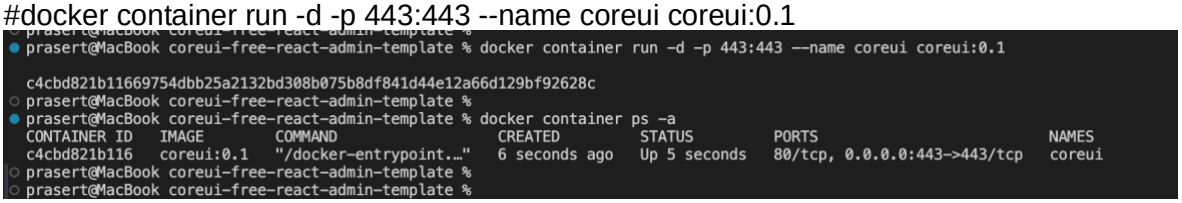
เช็ค image ที่สร้างใช้ command

#docker image ls -a



จากนั้นสร้าง container ชื่อ coreui และรันกับ image นี้

#docker container run -d -p 443:443 --name coreui coreui:0.1



เช็คว่า container รันได้ปกติไหม

#docker container ps -a

ตรวจสอบ nginx.conf และ certs ไฟล์ใน container ว่ามีการ copy ไปแล้วหรือยัง

#docker exec -it coreui /bin/sh;

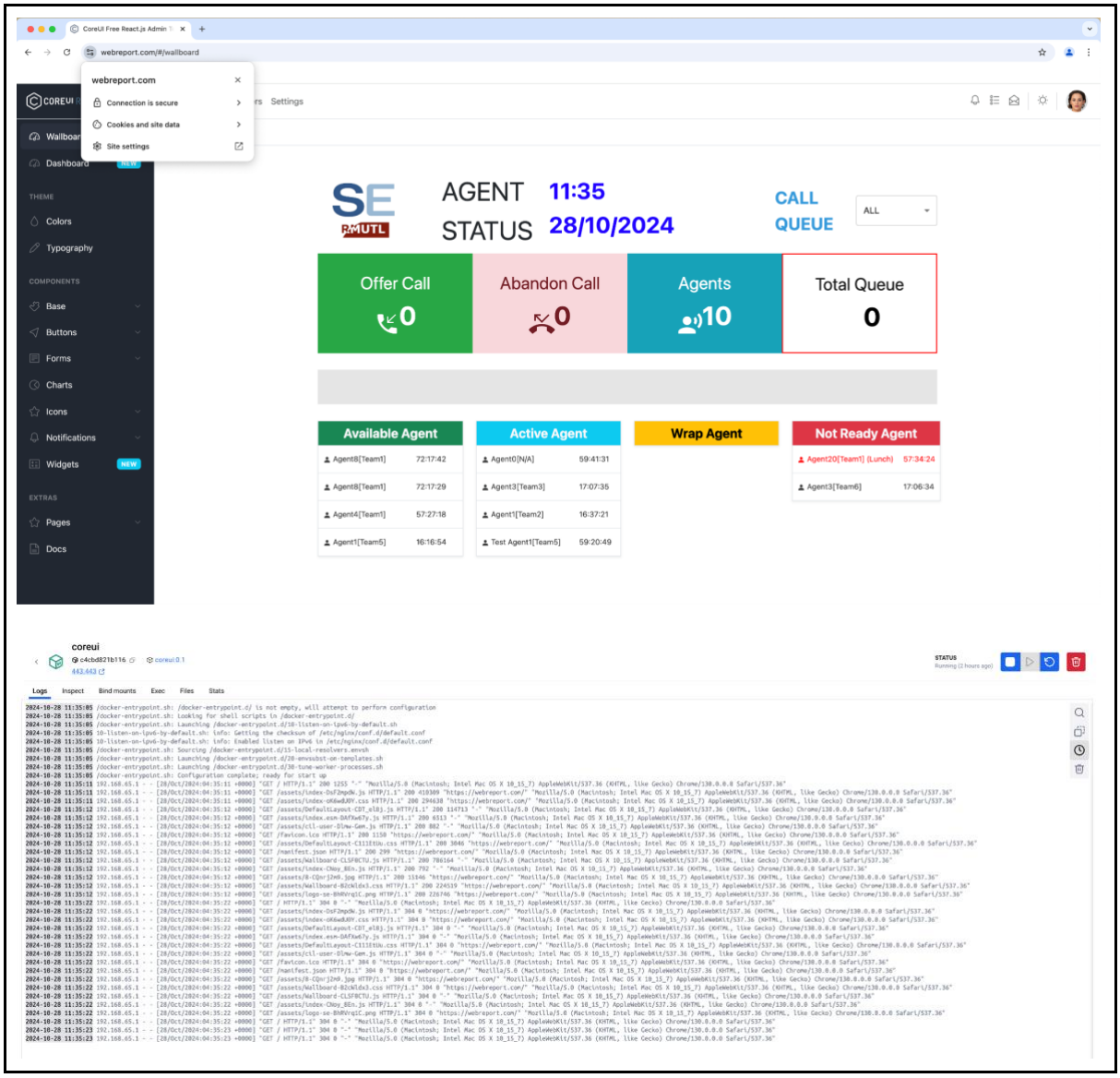
```
prasert@MacBook coreui-free-react-admin-template % docker exec -it coreui
/bin/sh;
/ #
/ #
/ #
/ #
```

```
/ # cat /etc/nginx/nginx.conf
# /etc/nginx/nginx.conf

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    # Server block for HTTPS
    server {
        listen 443 ssl;
        server_name webreport.com www.webreport.com;

        ssl_certificate /etc/nginx/ssl/webreport.com.crt;
        ssl_certificate_key /etc/nginx/ssl/webreport.com.key;

        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;

        root /usr/share/nginx/html;
        index index.html;

        location / {
            try_files $uri $uri/ =404;
        }
    }

    # Redirect HTTP to HTTPS
    server {
        listen 80;
        server_name webreport.com www.webreport.com;
        return 301 https://$host$request_uri;
    }
}
/ # ls -l /etc/nginx/ssl/
total 8
-rw-r--r--    1 root     root          1696 Oct 28 03:36
webreport.com.crt
-rw-------    1 root     root          1704 Oct 28 01:38
webreport.com.key
/ #
/ #
/ # ls -l /usr/share/nginx/html
total 20
-rw-r--r--    1 root     root           497 Oct  2 16:07 50x.html
drwxr-xr-x    2 root     root          4096 Oct 28 04:33 assets
-rw-r--r--    1 root     root          1150 Oct 28 04:33 favicon.ico
-rw-r--r--    1 root     root          1255 Oct 28 04:33 index.html
-rw-r--r--    1 root     root           299 Oct 28 04:33 manifest.json
/ #
```

แก้ไฟล์ hosts ที่เครื่อง client เพื่อที่จะสามารถเรียก name: webreport.com ได้

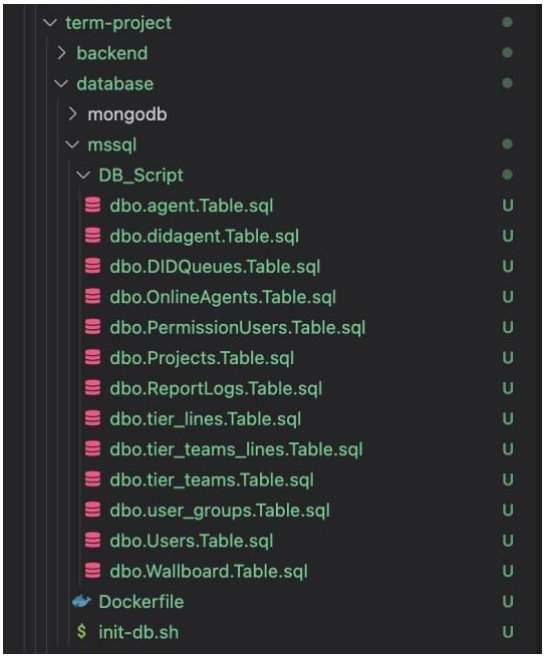
```
  UW PICO 5.09                                          File: /etc/hosts

##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1         localhost webreport.com coreui endpoint-api parse-server-api mssql mongodb
255.255.255.255   broadcasthost
::1               localhost
```

ทดสอบเข้า web กับ URL: https://webreport.com/



Database → MSSQL

สร้าง directory database จากนั้นสร้าง mssql ไว้สำหรับเก็บไฟล์ DB script



สร้าง shell script init-db.sh สำหรับ import DB

engse207-dev > microservices > term-project > database > mssql > $ init-db.sh

```
#!/bin/bash

# Start SQL Server in the background
/opt/mssql/bin/sqlservr &

# Wait for SQL Server to start
echo "Waiting for SQL Server to start..."
sleep 30s

# Check if this is the first initialization by looking for the .initialized file
if [ ! -f /var/opt/mssql/data/.initialized ]; then
  # Create the database
  echo "Creating database hana5_web_labDB..."
  /opt/mssql-tools18/bin/sqlcmd -S localhost -U SA -P "$MSSQL_SA_PASSWORD" -Q "CREATE DATABASE hana5_web_labDB" -N -C
  if [ $? -eq 0 ]; then
    echo "Database hana5_web_labDB created successfully."
  else
    echo "Failed to create database hana5_web_labDB." >&2
    exit 1
  fi

  # Run SQL scripts in the /docker-entrypoint-initdb.d directory within the created database
  echo "Running SQL scripts in /docker-entrypoint-initdb.d/..."
  for entry in /docker-entrypoint-initdb.d/*.sql
  do
    echo "Executing script: $entry on hana5_web_labDB..."
    /opt/mssql-tools18/bin/sqlcmd -S localhost -U SA -P "$MSSQL_SA_PASSWORD" -d hana5_web_labDB -i "$entry" -N -C
    if [ $? -eq 0 ]; then
      echo "$entry ran successfully."
    else
      echo "Error running $entry" >&2
      exit 1
    fi
  done

  # Create a file to indicate the database has been initialized
  touch /var/opt/mssql/data/.initialized
else
  echo "Database already initialized. Skipping initialization scripts."
fi

# Keep the container running in the foreground
wait
```

สร้าง Dockerfile สำหรับ build image mssql

engse207-dev > microservices > term-project > database > mssql > Dockerfile > ...

```
# Dockerfile for mssql (SQL Server)
FROM mcr.microsoft.com/mssql/server:2022-latest

# Set environment variables for SQL Server
ENV ACCEPT_EULA=Y
ENV MSSQL_SA_PASSWORD=Sa@2024%

# Copy the DB script to the init directory
COPY DB_Script/*.sql /docker-entrypoint-initdb.d/

# Copy the init-db.sh script to the init directory
COPY init-db.sh /docker-entrypoint-initdb.d/init-db.sh

# Expose SQL Server port
EXPOSE 1433

# Run the initialization script as the entrypoint
CMD [ "/bin/bash", "/docker-entrypoint-initdb.d/init-db.sh" ]
```

รัน command build image

#docker build --platform=linux/amd64 -t mssql:0.1 .

```
prasert@MacBook mssql % sudo docker build --platform=linux/amd64 -t mssql:0.1 .
Password:
Sorry, try again.
Password:
[+] Building 0.1s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 561B
 => WARN: SecretsUsedInArgOrEnv: Do not use ARG or ENV instructions for sensitive data (ENV "MSSQL_SA_PASSWORD") (line 6)
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 19)
 => [internal] load metadata for mcr.microsoft.com/mssql/server:2022-CU14-ubuntu-22.04
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM mcr.microsoft.com/mssql/server:2022-CU14-ubuntu-22.04
 => [internal] load build context
 => => transferring context: 3.12kB
 => CACHED [2/4] WORKDIR /app
 => CACHED [3/4] COPY DB_Script /app/DB_Script
 => CACHED [4/4] COPY init-db.sh /app/init-db.sh
 => exporting to image
 => => exporting layers
 => => writing image sha256:024258c52d97c88a63af3c2e656abc13bcf514ac29494acc811e6b7f7a5bd920
 => => naming to docker.io/library/mssql:0.1

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/9z5fwq0llgcr01ysgpvl964di

 2 warnings found (use docker --debug to expand):
 - SecretsUsedInArgOrEnv: Do not use ARG or ENV instructions for sensitive data (ENV "MSSQL_SA_PASSWORD") (line 6)
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 19)

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
prasert@MacBook mssql %
```

เช็ค image ที่สร้าง

#docker image ls -a

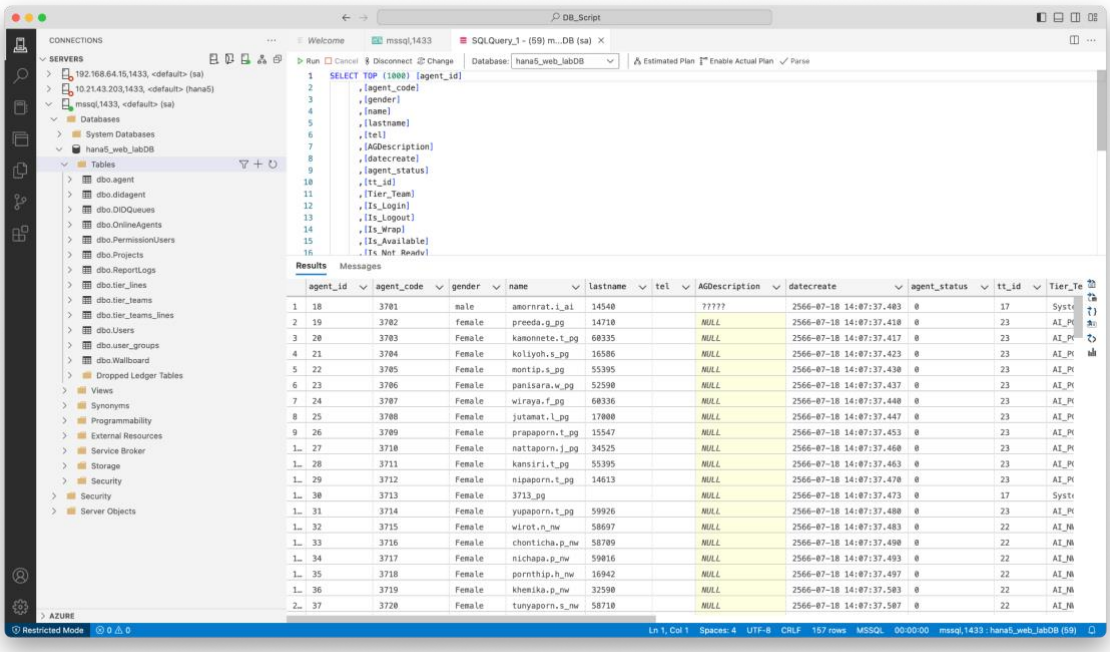
```
prasert@MacBook mssql % docker image ls -a
REPOSITORY                          TAG                  IMAGE ID       CREATED          SIZE
mssql                               0.1                  024258c52d97   27 minutes ago   1.59GB
endpoint-api                        0.1                  2fc22fd25f1b   3 hours ago      1.22GB
coreui                              0.1                  2d4d5245a71f   6 hours ago      59.4MB
nginx                               alpine               577a23b5858b   3 weeks ago      50.8MB
mcr.microsoft.com/mssql/server      2022-CU14-ubuntu-22.04   004f8fbb046c   3 months ago     1.59GB
node                                18                   b2c3c88b29ee   3 months ago     1.09GB
mongo                               4.4                  80d502872ebd   8 months ago     408MB
prasert@MacBook mssql %
```

จากนั้นสั่งรัน container mssql

#docker container run -d -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=Sa@2024%" -p 1433:1433 --name mssql mssql:0.1

```
prasert@MacBook mssql % docker container ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
1e1b6278e6ca   mssql:0.1          "/opt/mssql/bin/perm…"   10 minutes ago   Up 10 minutes   0.0.0.0:1433->1433/tcp                                 mssql
723c5f65c76f   endpoint-api:0.1   "docker-entrypoint.s…"   3 hours ago      Up 3 hours      0.0.0.0:5000->5000/tcp, 0.0.0.0:5101->5101/tcp         endpoint-api
c4cbd821b116   coreui:0.1         "/docker-entrypoint.…"   6 hours ago      Up 6 hours      80/tcp, 0.0.0.0:443->443/tcp                           coreui
prasert@MacBook mssql %
```

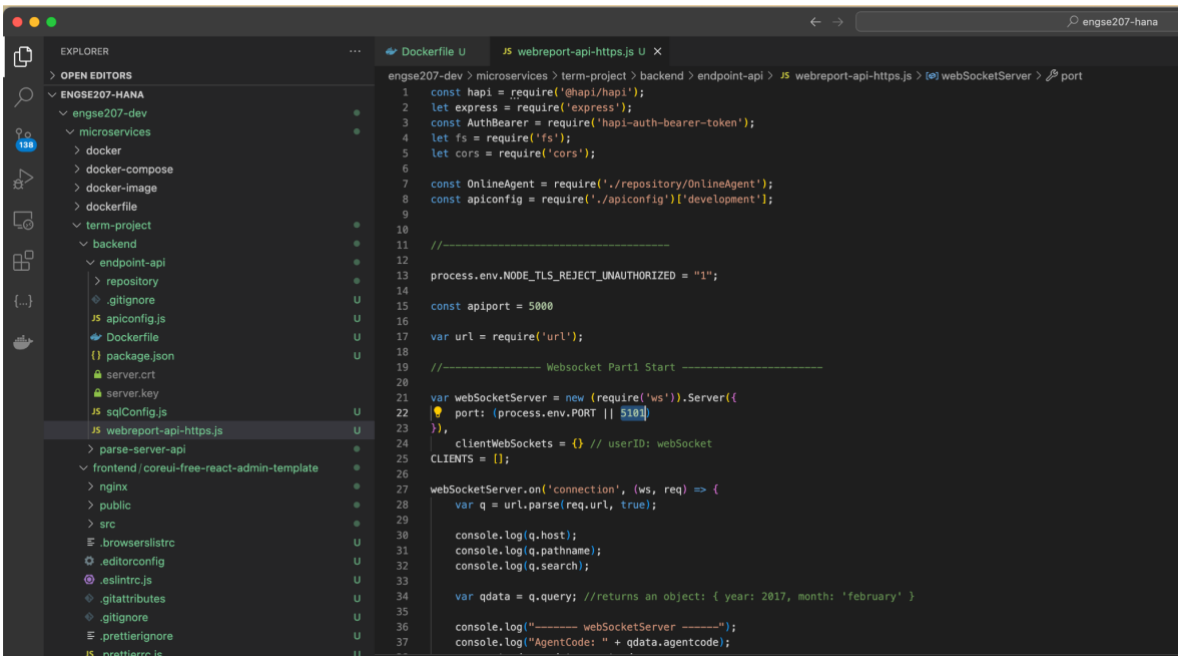
ทดสอบใช้ Azure Data Studio query ดูข้อมูลว่ามี import แล้วหรือยัง



Backend → endpoint-api

สร้าง directory backend จากนั้น copy server cert และ endpoint-api ไว้ภายใต้

และทำการเปลี่ยน API Port: 5000 WebSocket Port: 5101



แก้ Database connection ให้ชี้ไปที่ mssql container ที่เป็น DB ของ endpoint-api

สร้าง Script Dockerfile สำหรับ endpoint-api

```
# Dockerfile for endpoint-api (Node.js)
FROM node:18

# Set working directory
WORKDIR /app

# Copy package files and install dependencies
COPY package*.json ./
RUN npm install

# Copy application source code
COPY . .

# Copy SSL certificates to the container (if necessary for HTTPS)
COPY server.crt /app/server.crt
COPY server.key /app/server.key

# Expose ports for the endpoint-api
EXPOSE 5000
EXPOSE 5101

# Start the application
CMD ["node", "webreport-api-https.js"]

```

จากนั้นรัน command เพื่อ build image

#docker build -t endpoint-api:0.1 .

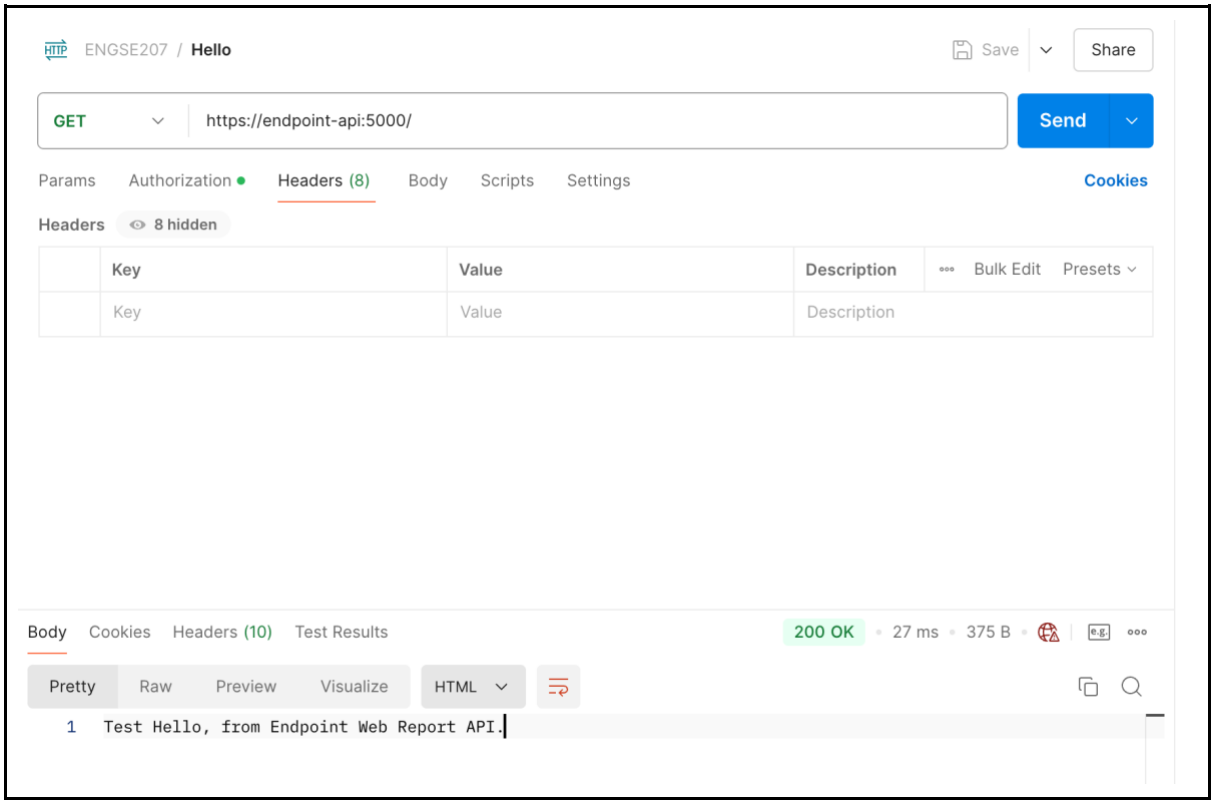
```
PROBLEMS  OUTPUT  TERMINAL  PORTS  COMMENTS  DEBUG CONSOLE

prasert@MacBook endpoint-api % sudo docker build -t endpoint-api:0.1 .
Password:
[+] Building 0.1s (12/12) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 512B
 => [internal] load metadata for docker.io/library/node:18
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/7] FROM docker.io/library/node:18
 => [internal] load build context
 => => transferring context: 14.42kB
 => CACHED [2/7] WORKDIR /app
 => CACHED [3/7] COPY package*.json ./
 => CACHED [4/7] RUN npm install
 => [5/7] COPY . .
 => [6/7] COPY server.crt /app/server.crt
 => [7/7] COPY server.key /app/server.key
 => exporting to image
 => => exporting layers
 => => writing image sha256:1ff8ac7901e55125347ce66ab68538b9c5eae69baaf3f57ad325bf97d923384d
 => => naming to docker.io/library/endpoint-api:0.1

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/rijwoyinvrbcglw0pys5k0rce

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
prasert@MacBook endpoint-api %
prasert@MacBook endpoint-api %
prasert@MacBook endpoint-api % docker image ls
REPOSITORY                     TAG           IMAGE ID       CREATED          SIZE
endpoint-api                   0.1           1ff8ac7901e5   5 seconds ago    1.22GB
coreui                         0.1           2d4d5245a71f   3 hours ago      59.4MB
nginx                          alpine        577a23b5858b   3 weeks ago      50.8MB
mcr.microsoft.com/mssql/server 2022-latest   b2761593c693   4 weeks ago      1.59GB
node                           18            b2c3c88b29ee   3 months ago     1.09GB
mongo                          4.4           80d502872ebd   8 months ago     408MB
prasert@MacBook endpoint-api %
```

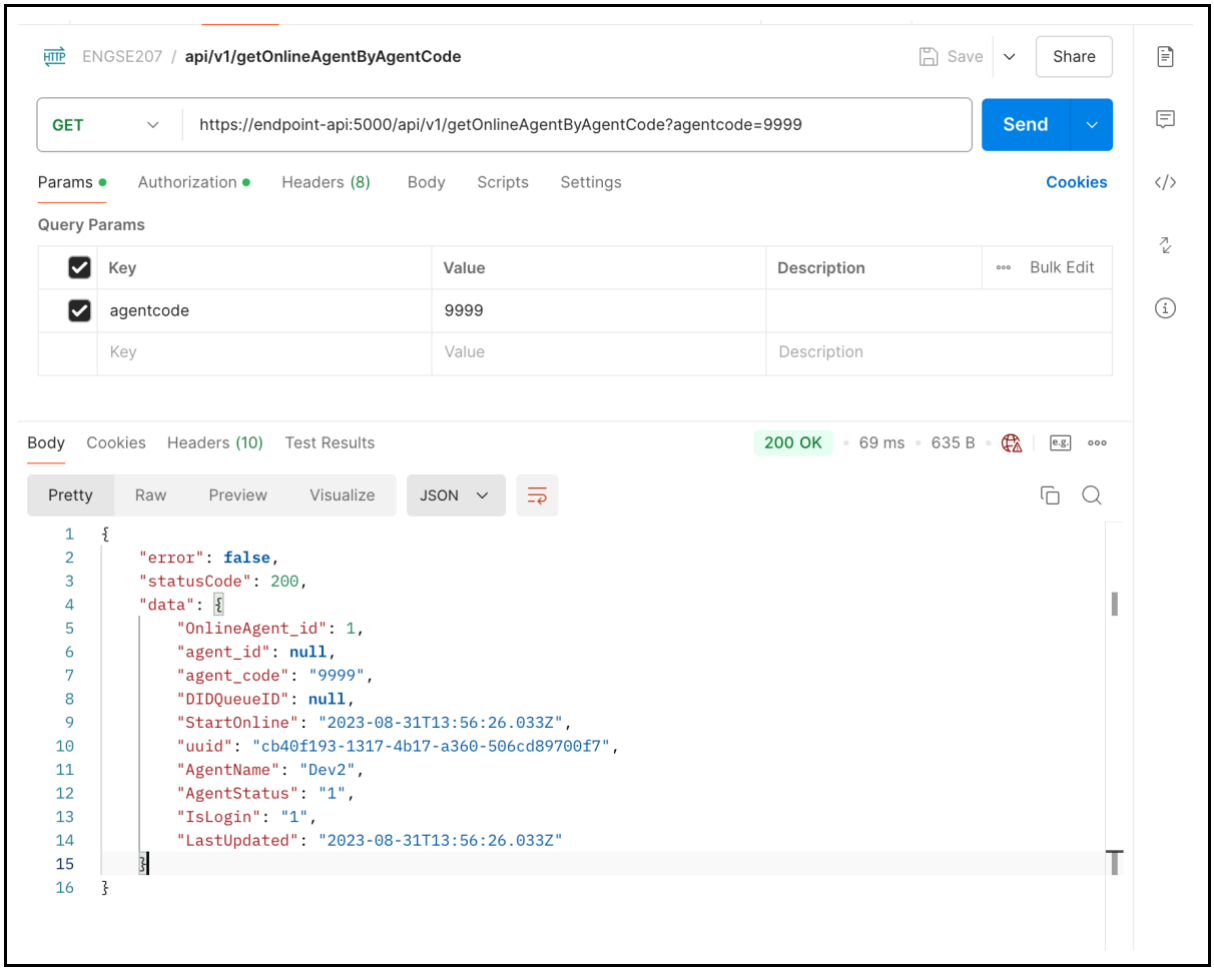
รัน endpoint-api กับ container

#docker container run -d -p 5000:5000 -p 5101:5101 --name endpoint-api endpoint-api:0.1

```
prasert@MacBook endpoint-api % docker container run -d -p 5000:5000 -p 5101:5101 --name endpoint-api endpoint-api:0.1
cd716e7b53bd2f14b7c7328ecaca7ef7759b83ce420c0bc028dbc8d293641b26
prasert@MacBook endpoint-api %
prasert@MacBook endpoint-api %
prasert@MacBook endpoint-api % docker container ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS                                                NAMES
cd716e7b53bd   endpoint-api:0.1   "docker-entrypoint.s…"   2 seconds ago   Up 2 seconds   0.0.0.0:5000->5000/tcp, 0.0.0.0:5101->5101/tcp   endpoint-api
c4cbd821b116   coreui:0.1         "/docker-entrypoint.…"   3 hours ago     Up 3 hours     80/tcp, 0.0.0.0:443->443/tcp                         coreui
prasert@MacBook endpoint-api %
```

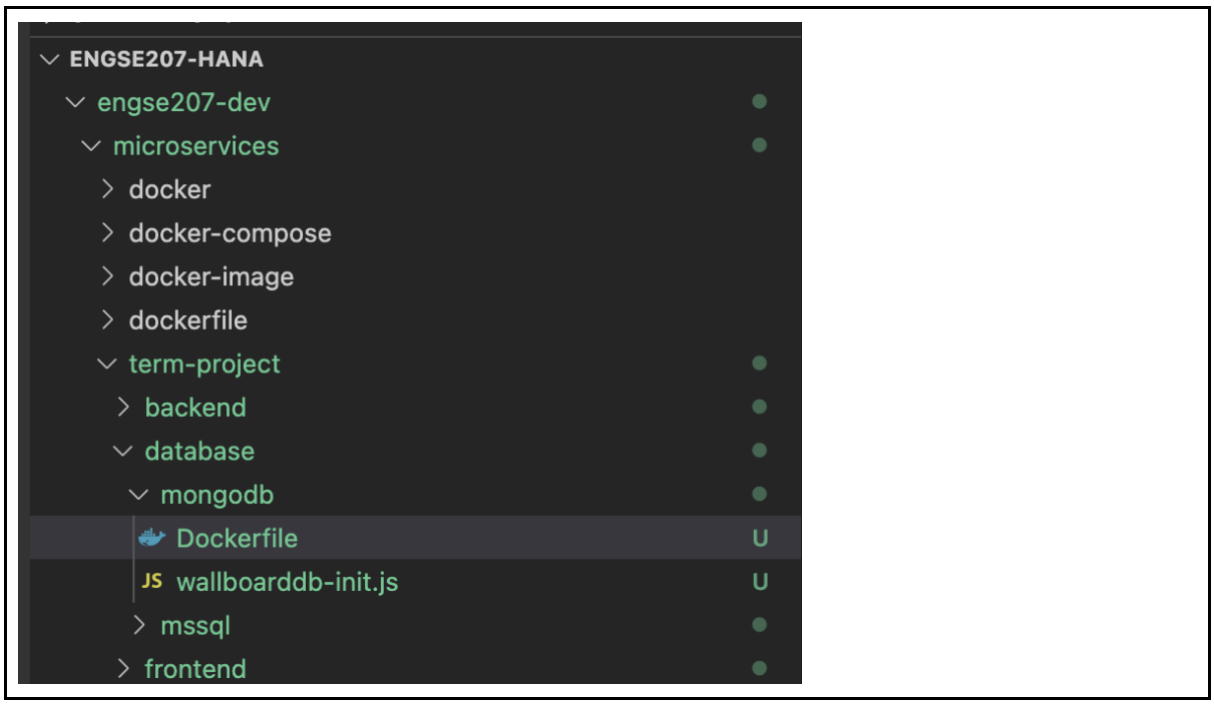
จากนั้นใช้ Postman test API connection





Database → MongoDB

สร้าง directory mongodb สำหรับเก็บไฟล์ wallboarddb-init.js และ Dockerfile



สร้างไฟล์ wallboarddb-init.js เป็น script สำหรับสร้าง DB และ users ใน MongoDB

engse207-dev > microservices > term-project > database > mongodb > JS wallboarddb-init.js > ...

```
// Switch to wallboarddb database and create wallboarduser
db = db.getSiblingDB('wallboarddb');
db.createUser({
    user: "wallboarduser",
    pwd: "WB1qazxsw2",
    roles: [{ role: "readWrite", db: "wallboarddb" }]
});

// Switch to admin database and create wallboardadmin
db = db.getSiblingDB('admin');
db.createUser({
    user: "wallboardadmin",
    pwd: "WB1qazxsw2",
    roles: [
        { role: "userAdminAnyDatabase", db: "admin" },
        "readWriteAnyDatabase"
    ]
});
```

สร้างไฟล์ Dockerfile สำหรับ build image MongoDB

engse207-dev > microservices > term-project > database > mongodb > Dockerfile > ...

```
# Use MongoDB 4.4 official image
FROM mongo:4.4

# Set environment variables for root user creation
ENV MONGO_INITDB_ROOT_USERNAME=admin
ENV MONGO_INITDB_ROOT_PASSWORD=@se2024
ENV MONGO_INITDB_DATABASE=wallboarddb

# Copy the custom-init.js script to the init directory
COPY wallboarddb-init.js /docker-entrypoint-initdb.d/wallboarddb-init.js

# Expose MongoDB port
EXPOSE 27017
```

docker-entrypoint-initdb.d

Vinayak Salunkhe · Oct 7, 2023 · 1 min read

The `docker-entrypoint-initdb.d` directory is used to store shell or SQL scripts that you want to be executed when a Docker container is started for the first time. Here are some key points:

1. **Initialization**: When a container is started for the first time, a new database with the specified name will be created and initialized with the provided configuration variables.
2. **Script Execution**: It will execute files with extensions `.sh`, `.sql` and `.sql` that are found in `/docker-entrypoint-initdb.d`. The scripts inside that directory will only get executed once - when initializing a fresh instance.
3. **Order of Execution**: The `/docker-entrypoint-initdb.d/init.sql` is executed the moment your database container starts running, while your `entrypoint.sh` is executed the moment your web container starts running. Since your web container depends on your database container, the SQL script will always be executed ahead of your entrypoint.

จากนั้นรัน command เพื่อ build image mongodb

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  COMMENTS  DEBUG CONSOLE

prasert@MacBook mongodb % sudo docker build -t mongodb:0.1 .

Password:
[+] Building 0.1s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 418B
 => WARN: SecretsUsedInArgOrEnv: Do not use ARG or ENV instructions for sensitive data (ENV "MONGO_INITDB_ROOT_PASSWORD") (line 6)
 => [internal] load metadata for docker.io/library/mongo:4.4
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 41B
 => [1/2] FROM docker.io/library/mongo:4.4
 => CACHED [2/2] COPY wallboarddb-init.js /docker-entrypoint-initdb.d/wallboarddb-init.js
 => exporting to image
 => => exporting layers
 => => writing image sha256:2c9042bcf2a63e5ea43859685910298f5d0d8cbad4068de074fc3e9141a813f1
 => => naming to docker.io/library/mongodb:0.1

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/o65gppe9uxvbe8yt359hau1sp

 1 warning found (use docker --debug to expand):
 - SecretsUsedInArgOrEnv: Do not use ARG or ENV instructions for sensitive data (ENV "MONGO_INITDB_ROOT_PASSWORD") (line 6)

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
prasert@MacBook mongodb % docker image ls
REPOSITORY                      TAG           IMAGE ID       CREATED          SIZE
mongodb                         0.1           2c9042bcf2a6   50 minutes ago   408MB
mssql                           0.1           841de0baa972   19 hours ago     1.59GB
endpoint-api                    0.1           2fc22fd25f1b   22 hours ago     1.22GB
coreui                          0.1           2d4d5245a71f   25 hours ago     59.4MB
nginx                           alpine        577a23b5858b   3 weeks ago      50.8MB
mcr.microsoft.com/mssql/server  2022-latest   b2761593c693   4 weeks ago      1.59GB
node                            18            b2c3c88b29ee   3 months ago     1.09GB
mongo                           4.4           80d502872ebd   8 months ago     408MB
prasert@MacBook mongodb %
```

รัน mongodb กับ container

#docker container run -d -p 27017:27017 --name mongodb mongodb:0.1

```
prasert@MacBook mongodb % docker container run -d -p 27017:27017 --name mongodb mongodb:0.1

0e9fcc5091b788747bd2b5b794b4a84547039507a27e6e8020a7d50baad1d4c5
prasert@MacBook mongodb %
prasert@MacBook mongodb % docker container ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                      NAMES
0e9fcc5091b7   mongodb:0.1    "docker-entrypoint.s…"   13 minutes ago   Up 13 minutes   0.0.0.0:27017->27017/tcp   mongodb
```

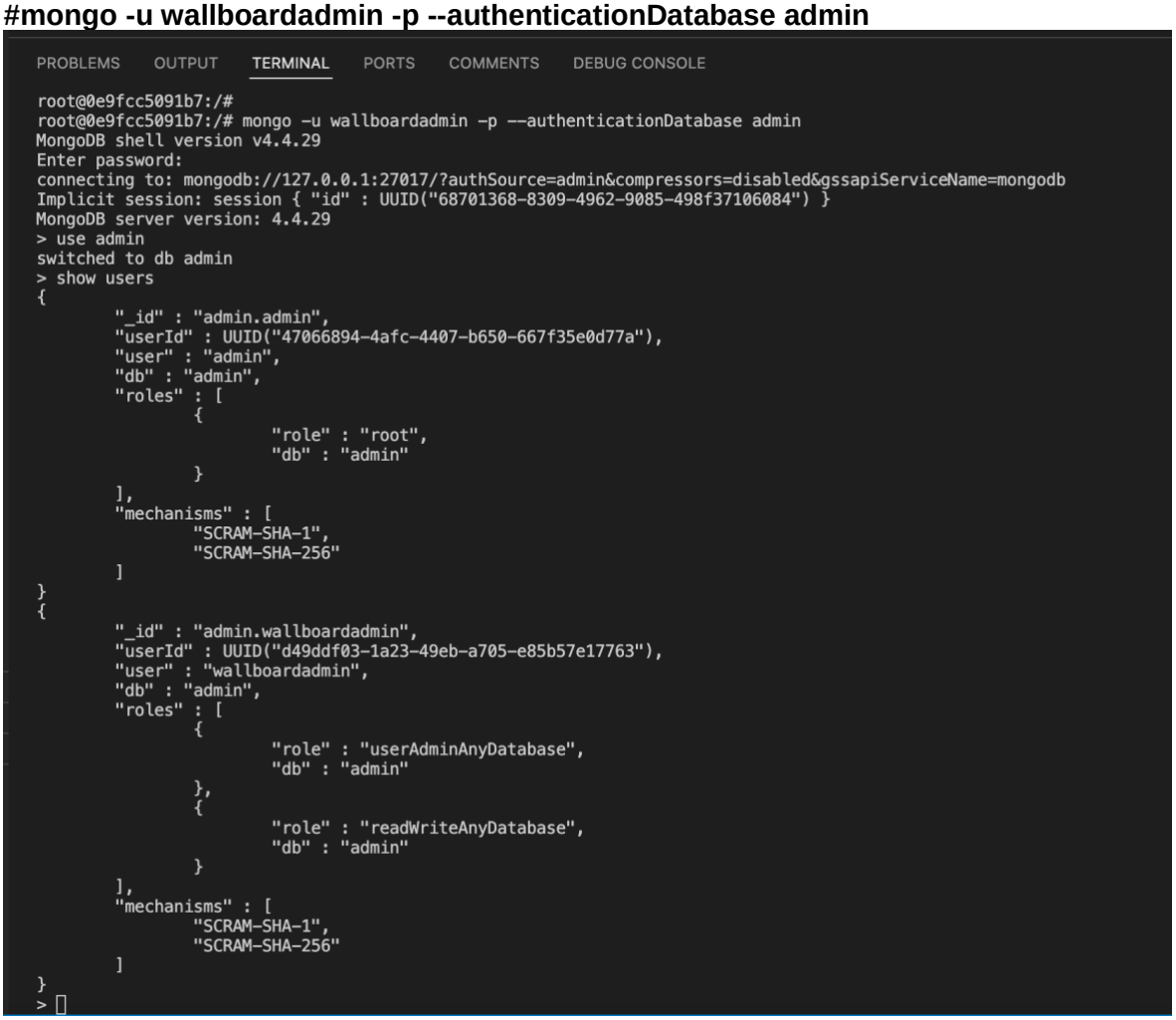
เข้าไปที่ shell ของ mongodb container เพื่อตรวจสอบความถูกต้องของ DB

เมื่อเราทดสอบ mongo shell แบบไม่มีการ auth จะไม่สามารถ query data ใดๆ ได้

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  COMMENTS  DEBUG CONSOLE

 *  Executing task: docker exec -it 0e9fcc5091b788747bd2b5b794b4a84547039507a27e6e8020a7d50baad1d4c5 bash

root@0e9fcc5091b7:/#
root@0e9fcc5091b7:/# mongo
MongoDB shell version v4.4.29
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b985a0a8-fa07-4bd6-a23f-8c6d586071ad") }
MongoDB server version: 4.4.29
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
> use admin
switched to db admin
> show users
uncaught exception: Error: command usersInfo requires authentication :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DB.prototype.getUsers@src/mongo/shell/db.js:1659:15
shellHelper.show@src/mongo/shell/utils.js:914:9
shellHelper@src/mongo/shell/utils.js:819:15
@(shellhelp2):1:1
>
```

จากนั้นลอง auth login จะสามารถ query data users ได้

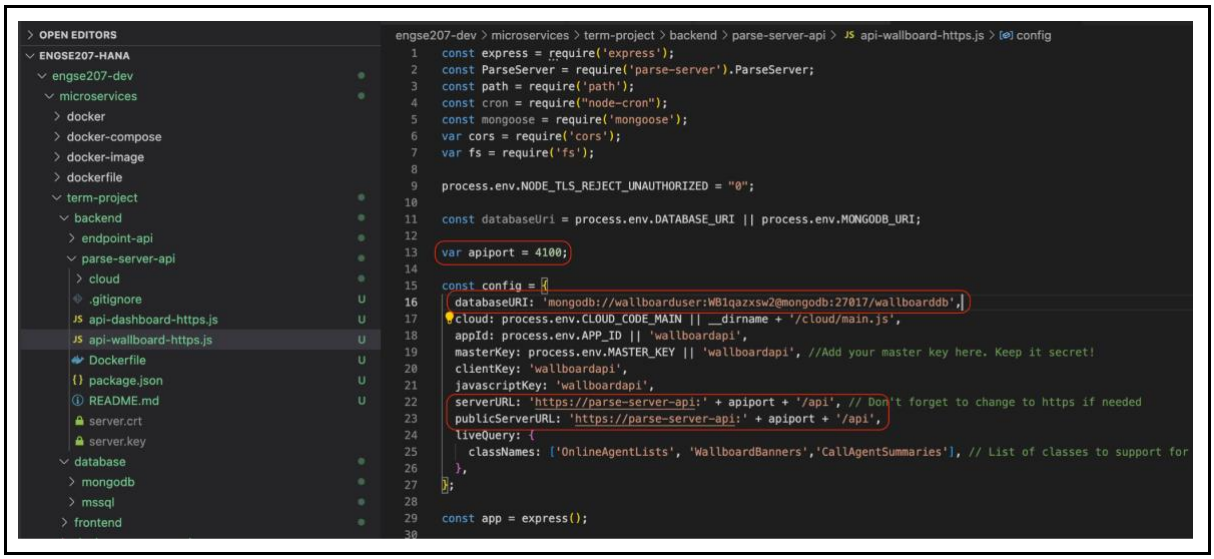
#mongo -u wallboardadmin -p --authenticationDatabase admin



Backend → parse-server-api

copy server cert และ parse-server-api ไว้ภายใต้ directory backend

แก้ไขไฟล์ api-wallboard-https.js ทำการเปลี่ยน Parse Server API Port: 4100 และ Server URL และ mongodb connection



แก้ไขไฟล์ api-dashboard-https.js เปลี่ยน Parse Server API Port: 4100 และ Parse Dashboard Port 4101 และ Server URL

สร้าง setup-classes.sh shell script สำหรับสร้าง classNames: ['OnlineAgentLists', 'WallboardBanners','CallAgentSummaries']

engse207-dev > microservices > term-project > backend > parse-server-api > $ setup-classes.sh

```sh
#!/bin/sh

echo "Waiting for Parse Server to start..."
sleep 30s

# Check if the setup has already been run
if [ -f "/app/class-setup-done" ]; then
  echo "Classes have already been created. Skipping setup."
  exit 0
fi

# Define Parse Server URL and credentials
PARSE_SERVER_URL="https://parse-server-api:4100/api/schemas"
APP_ID="wallboardapi"
MASTER_KEY="wallboardapi"

# Create each class if it hasn't been created before
curl -k -X POST -H "X-Parse-Application-Id: $APP_ID" -H "X-Parse-Master-Key: $MASTER_KEY" -H "Content-Type: application/json" -d '{"className": "OnlineAgentLists"}' $PARSE_SERVER_URL/OnlineAgentLists
curl -k -X POST -H "X-Parse-Application-Id: $APP_ID" -H "X-Parse-Master-Key: $MASTER_KEY" -H "Content-Type: application/json" -d '{"className": "WallboardBanners"}' $PARSE_SERVER_URL/WallboardBanners
curl -k -X POST -H "X-Parse-Application-Id: $APP_ID" -H "X-Parse-Master-Key: $MASTER_KEY" -H "Content-Type: application/json" -d '{"className": "CallAgentSummaries"}' $PARSE_SERVER_URL/CallAgentSummaries

# Create the status file to prevent re-running
touch /app/class-setup-done
echo "Classes have been created, and setup flag has been set."

```

สร้าง Script Dockerfile สำหรับ parse-server-api

engse207-dev > microservices > term-project > backend > parse-server-api > Dockerfile > ...

```dockerfile
# Dockerfile for parse-server-api (Node.js)
FROM node:18

# Set working directory
WORKDIR /app

# Copy package files and install dependencies
COPY package*.json ./
RUN npm install

# Copy application source code
COPY . .

# Copy SSL certificates to the container (if necessary for HTTPS)
COPY server.crt /app/server.crt
COPY server.key /app/server.key

# Copy the setup script into the container
COPY setup-classes.sh /app/setup-classes.sh

# Ensure setup script has executable permissions
RUN chmod +x /app/setup-classes.sh

# Expose ports for the backend
EXPOSE 4100
EXPOSE 4101

# Run the setup script on the first run and start the applications
CMD ["sh", "-c", "/app/setup-classes.sh & node api-wallboard-https.js & node api-dashboard-https.js"]

```

จากนั้นรัน command เพื่อ build image

#docker build -t parse-server-api:0.1 .

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  COMMENTS  DEBUG CONSOLE

 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 23)
 => [internal] load metadata for docker.io/library/node:18
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/7] FROM docker.io/library/node:18
 => [internal] load build context
 => => transferring context: 13.76kB
 => CACHED [2/7] WORKDIR /app
 => [3/7] COPY package*.json ./
 => [4/7] RUN npm install
 => [5/7] COPY . .
 => [6/7] COPY server.crt /app/server.crt
 => [7/7] COPY server.key /app/server.key
 => exporting to image
 => => exporting layers
 => => writing image sha256:713a096592a41fa074c5a9a442eb18f1fdf5f5bc7791fac41909d51bb7c2b7fa
 => => naming to docker.io/library/parse-server-api:0.1

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/tad2ijhed430485obde7gcyfg

 1 warning found (use docker --debug to expand):
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 23)

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
prasert@MacBook parse-server-api % docker image ls
REPOSITORY                      TAG           IMAGE ID       CREATED          SIZE
parse-server-api                0.1           713a096592a4   24 seconds ago   1.75GB
mssql                           0.1           887e96d74846   56 minutes ago   1.59GB
mongodb                         0.1           2c9042bcf2a6   3 hours ago      408MB
endpoint-api                    0.1           2fc22fd25f1b   24 hours ago     1.22GB
coreui                          0.1           2d4d5245a71f   27 hours ago     59.4MB
nginx                           alpine        577a23b5858b   3 weeks ago      50.8MB
mcr.microsoft.com/mssql/server  2022-latest   b2761593c693   4 weeks ago      1.59GB
node                            18            b2c3c88b29ee   3 months ago     1.09GB
mongo                           4.4           80d502872ebd   8 months ago     408MB
```

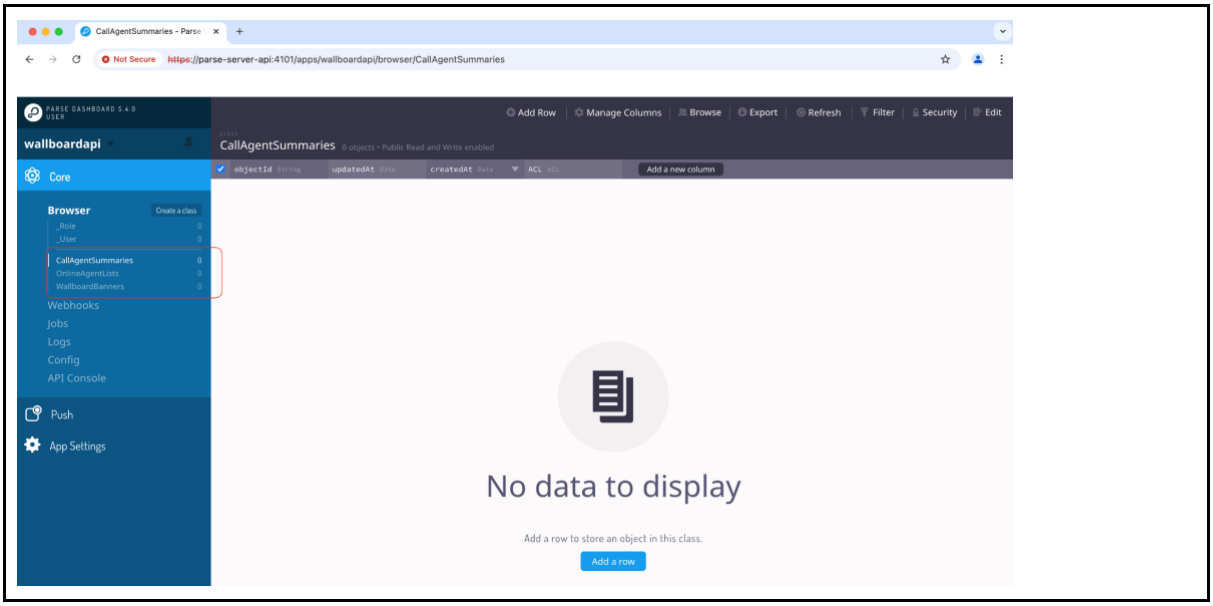
รัน parse-server-api กับ container

#docker container run -d -p 4100:4100 -p 4101:4101 --name parse-server-api parse-server-api:0.1

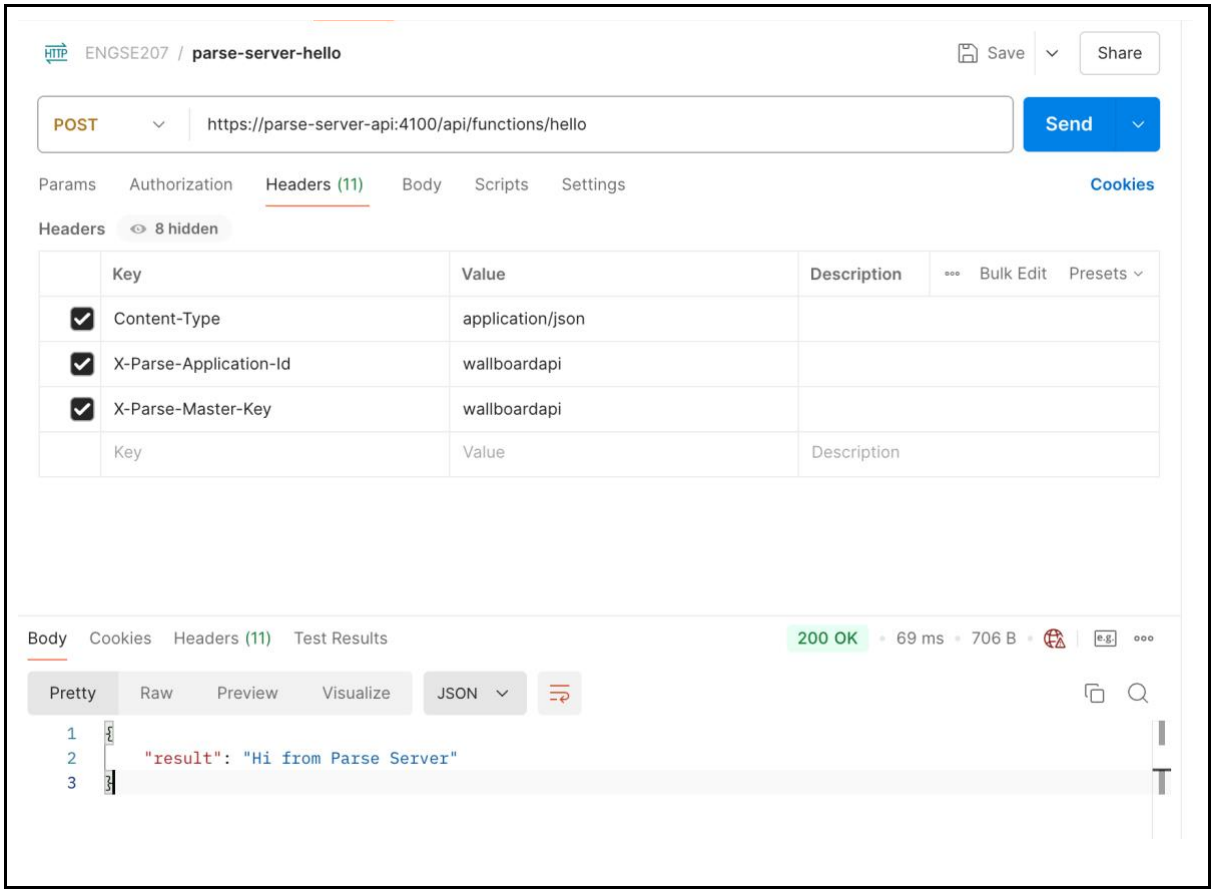
```
prasert@MacBook parse-server-api % docker container run -d -p 4100:4100 -p 4101:4101 --name parse-server-api parse-server-api:0.1

0763eb68103de6bcae1cc661ee32cfb880809acfed44695065e30d8d3ead9e7a
prasert@MacBook parse-server-api % docker container ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                                                                NAMES
0763eb68103d   parse-server-api:0.1  "docker-entrypoint.s…"   14 seconds ago   Up 14 seconds   0.0.0.0:4100-4101->4100-4101/tcp                                     parse-server-api
a0cda6506aae   mongodb:0.1           "docker-entrypoint.s…"   38 minutes ago   Up 38 minutes   0.0.0.0:27017->27017/tcp                                             mongodb
d6db89283eb5   mssql:0.1             "/opt/mssql/bin/perm…"   38 minutes ago   Up 38 minutes   0.0.0.0:1433->1433/tcp                                               mssql
70c00747853e   coreui:0.1            "/docker-entrypoint.…"   38 minutes ago   Up 38 minutes   80/tcp, 0.0.0.0:443->443/tcp                                         coreui
d1fb3f4b689b   endpoint-api:0.1      "docker-entrypoint.s…"   38 minutes ago   Up 38 minutes   0.0.0.0:5000->5000/tcp, 0.0.0.0:5101->5101/tcp                       endpoint-api
```

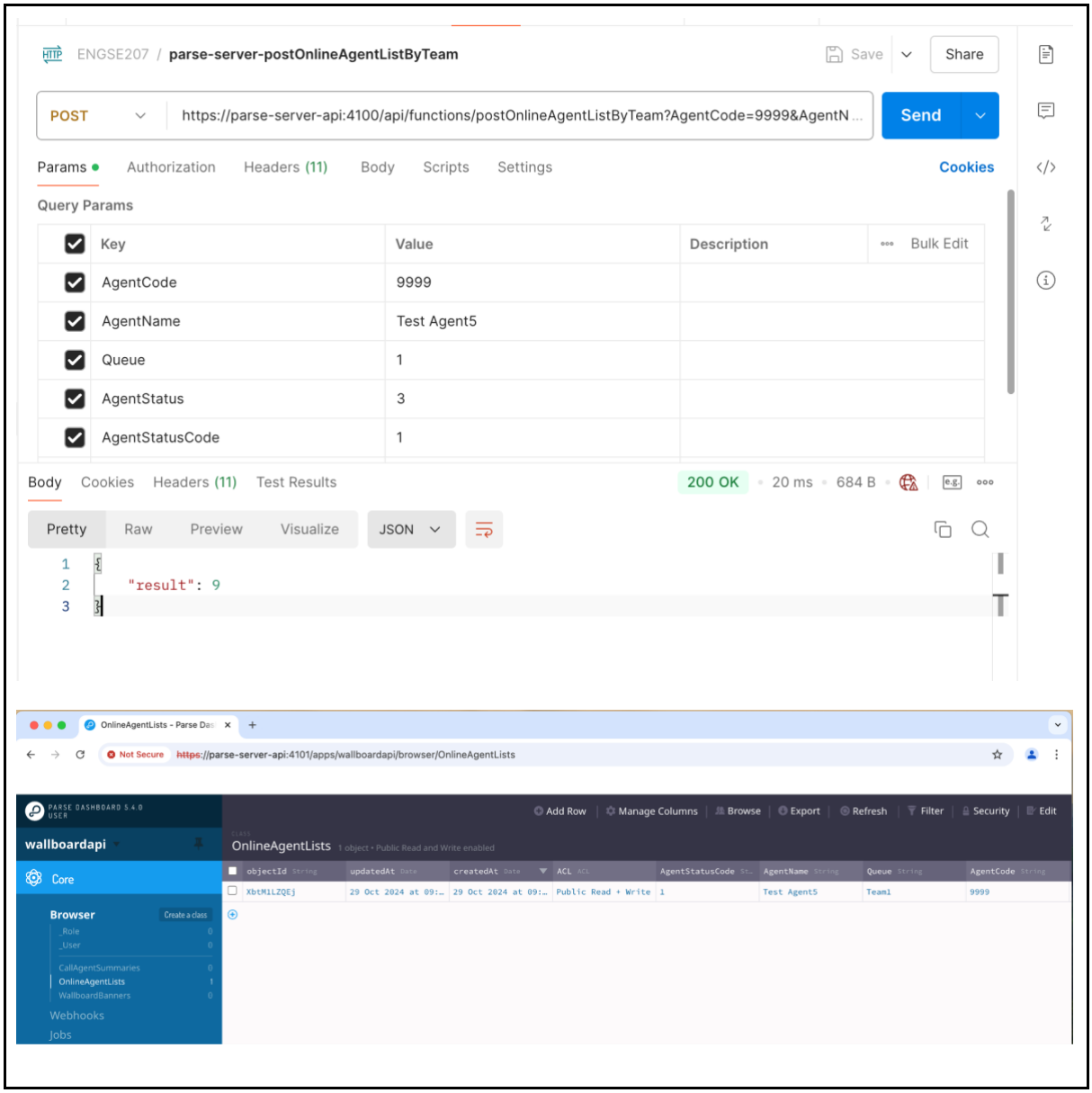
ตรวจสอบว่า classNames ได้ถูกสร้างแล้วหรือยัง on Parse Dashboard https://parse-server-api:4101/

classNames: ['OnlineAgentLists', 'WallboardBanners','CallAgentSummaries']



จากนั้นใช้ Postman test API connection



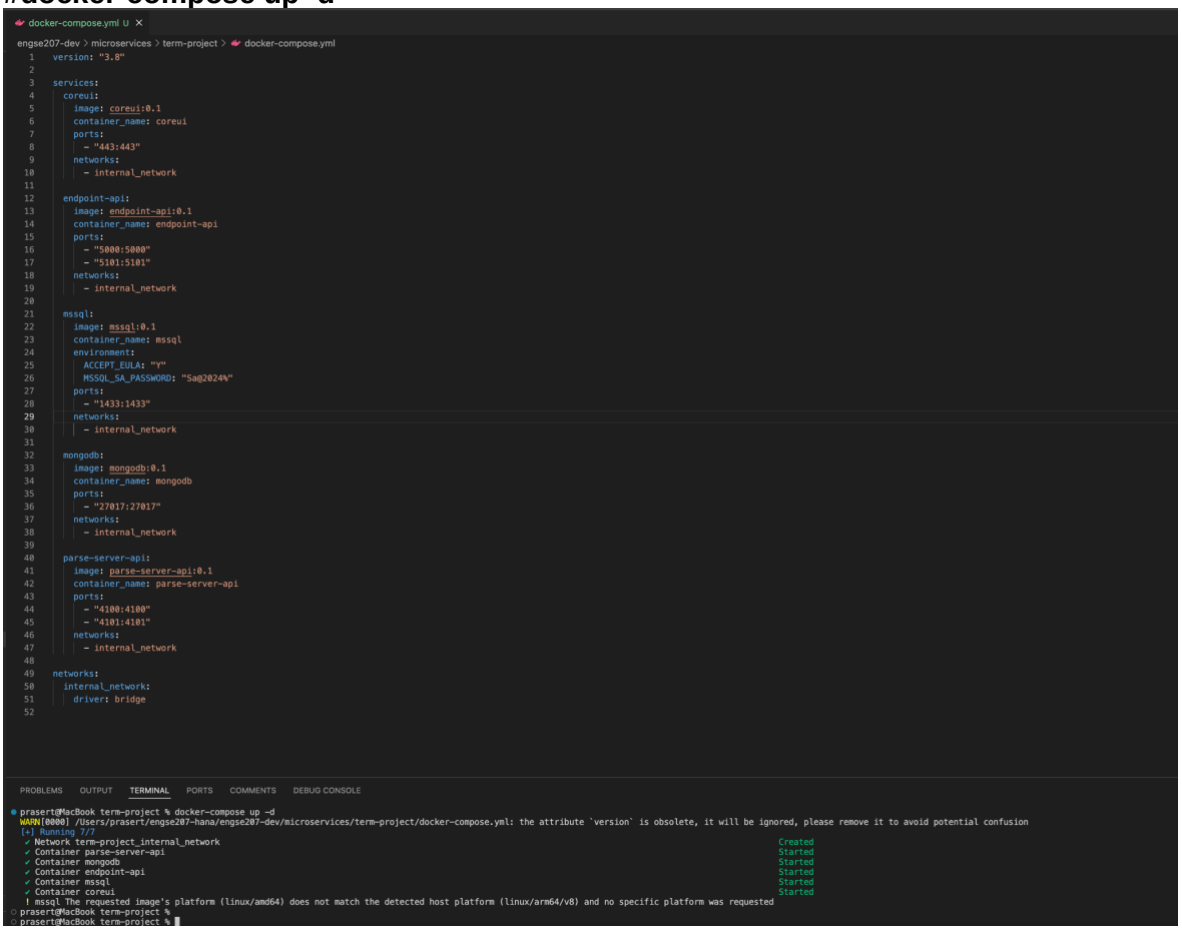

ENGSE207 / parse-server-postOnlineAgentListByTeam
Save
Share
POST
https://parse-server-api:4100/api/functions/postOnlineAgentListByTeam?AgentCode=9999&AgentN...
Send
Params
Authorization
Headers (11)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
AgentCode
9999
AgentName
Test Agent5
Queue
1
AgentStatus
3
AgentStatusCode
1
Body
Cookies
Headers (11)
Test Results
200 OK
20 ms
684 B
Pretty
Raw
Preview
Visualize
JSON
{
"result": 9
}

OnlineAgentLists - Parse Das
Not Secure
https://parse-server-api:4101/apps/wallboardapi/browser/OnlineAgentLists
PARSE DASHBOARD 5.4.0
USER
wallboardapi
Core
Browser
Create a class
_Role
_User
CallAgentSummaries
OnlineAgentLists
WallboardBanners
Webhooks
Jobs
Add Row
Manage Columns
Browse
Export
Refresh
Filter
Security
Edit
CLASS
OnlineAgentLists
1 object • Public Read and Write enabled
objectId String
updatedAt Date
createdAt Date
ACL ACL
AgentStatusCode St...
AgentName String
Queue String
AgentCode String
XbtM1LZQEj
29 Oct 2024 at 09:...
29 Oct 2024 at 09:...
Public Read + Write
1
Test Agent5
Team1
9999

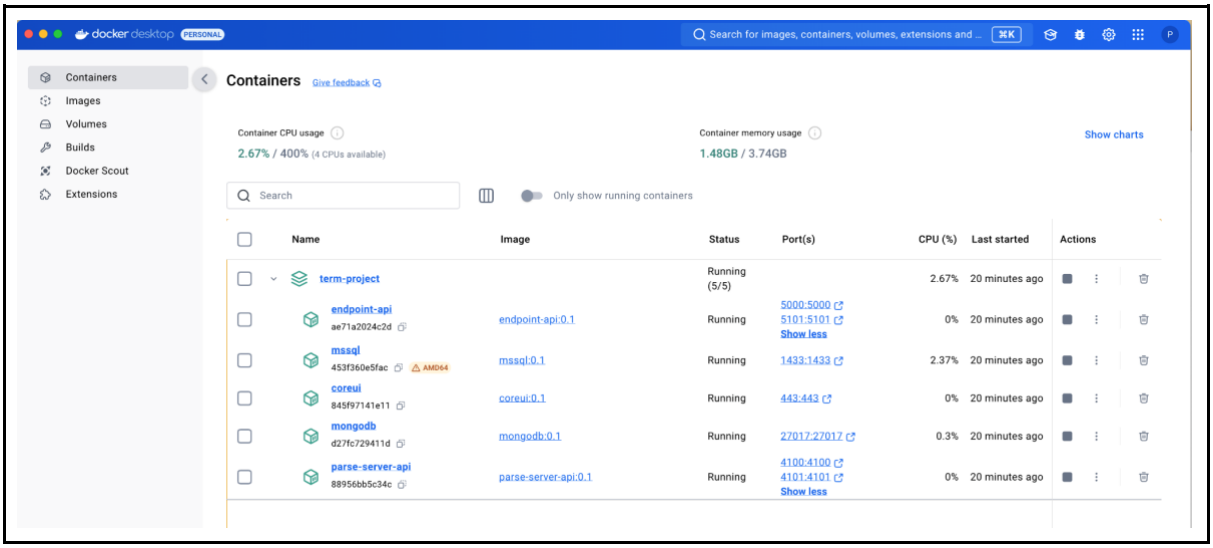
ใช้ docker-compose ในการบริหารจัดการ container ซึ่งง่ายและสะดวก

สร้างไฟล์ docker-compose.yml ภายใต้ directory term-project

สามารถ start container ทั้งหมดได้ด้วย command
#docker-compose up -d

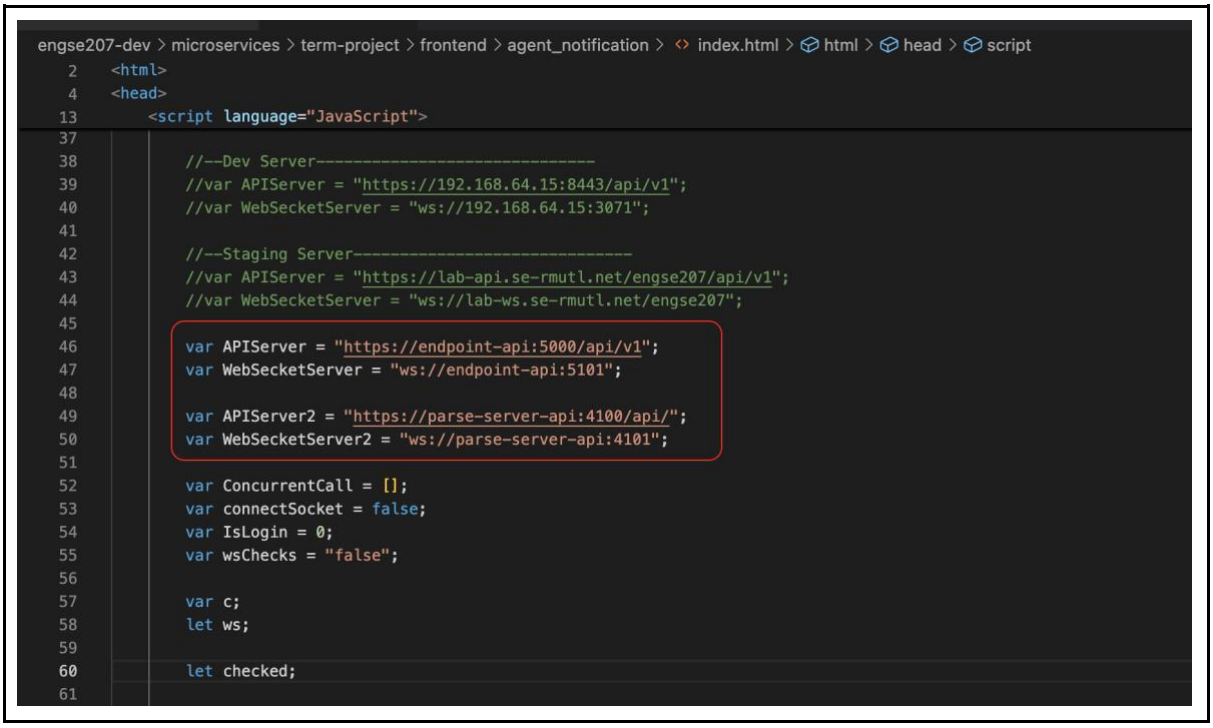


Container list



Frontend → agent_notification Electron.js

แก้ไฟล์ index.html ส่วนของ APIServer และ WebSecketServer ให้ชี้มาที่ parse-server-api และ endpoint-api



```
engse207-dev > microservices > term-project > frontend > agent_notification > index.html > html > head > script
 2   <html>
 4   <head>
13       <script language="JavaScript">
37
38           //--Dev Server------------------------------
39           //var APIServer = "https://192.168.64.15:8443/api/v1";
40           //var WebSecketServer = "ws://192.168.64.15:3071";
41
42           //--Staging Server-------------------------------
43           //var APIServer = "https://lab-api.se-rmutl.net/engse207/api/v1";
44           //var WebSecketServer = "ws://lab-ws.se-rmutl.net/engse207";
45
46           var APIServer = "https://endpoint-api:5000/api/v1";
47           var WebSecketServer = "ws://endpoint-api:5101";
48
49           var APIServer2 = "https://parse-server-api:4100/api/";
50           var WebSecketServer2 = "ws://parse-server-api:4101";
51
52           var ConcurrentCall = [];
53           var connectSocket = false;
54           var IsLogin = 0;
55           var wsChecks = "false";
56
57           var c;
58           let ws;
59
60           let checked;
61
```

จากนั้น npm install และ npm start agent_notification app ทดสอบ connect agentcode 1234

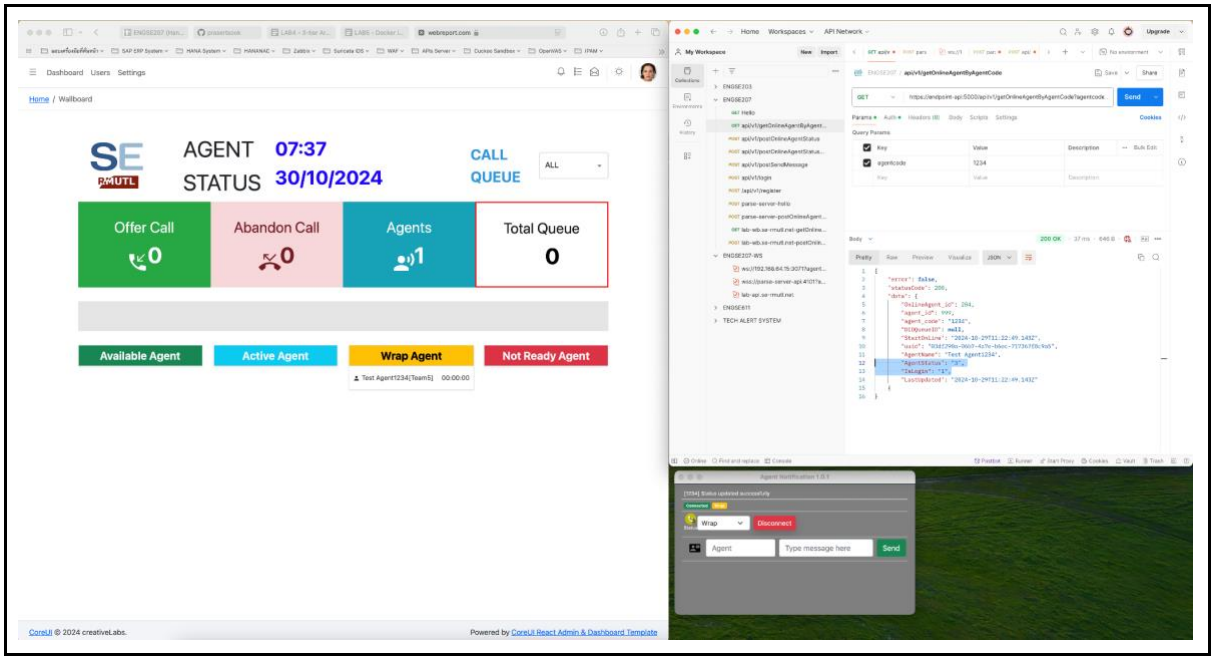
เปลี่ยน agent status ไปยัง Wrap จะเห็นว่า Wallboard update status realtime

Wallboard จะติดต่อกับ parse-server-api

ส่วน agent_notification จะติดต่อกับทั้งสอง parse-server-api และ endpoint-api

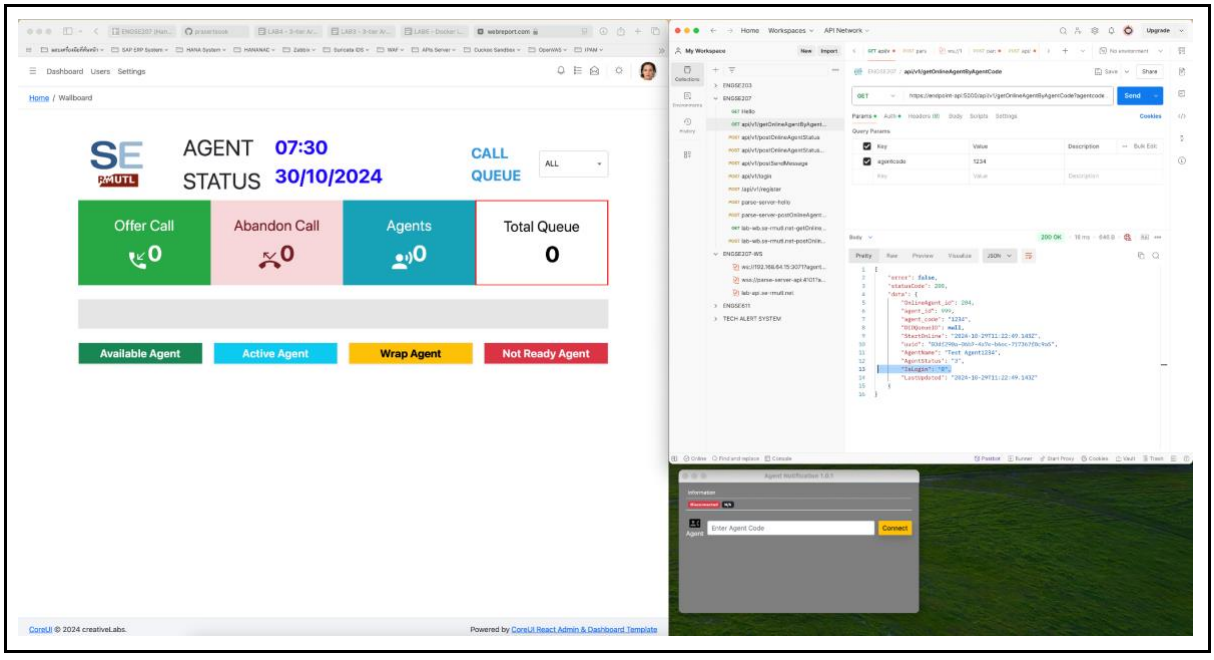
จากนั้นใช้ postman get data https://endpoint-api:5000/api/v1/getOnlineAgentByAgentCode?agentcode=1234

จะเห็นว่า "AgentStatus": "3" ก็คือ Wrap และ "IsLogin": "1"

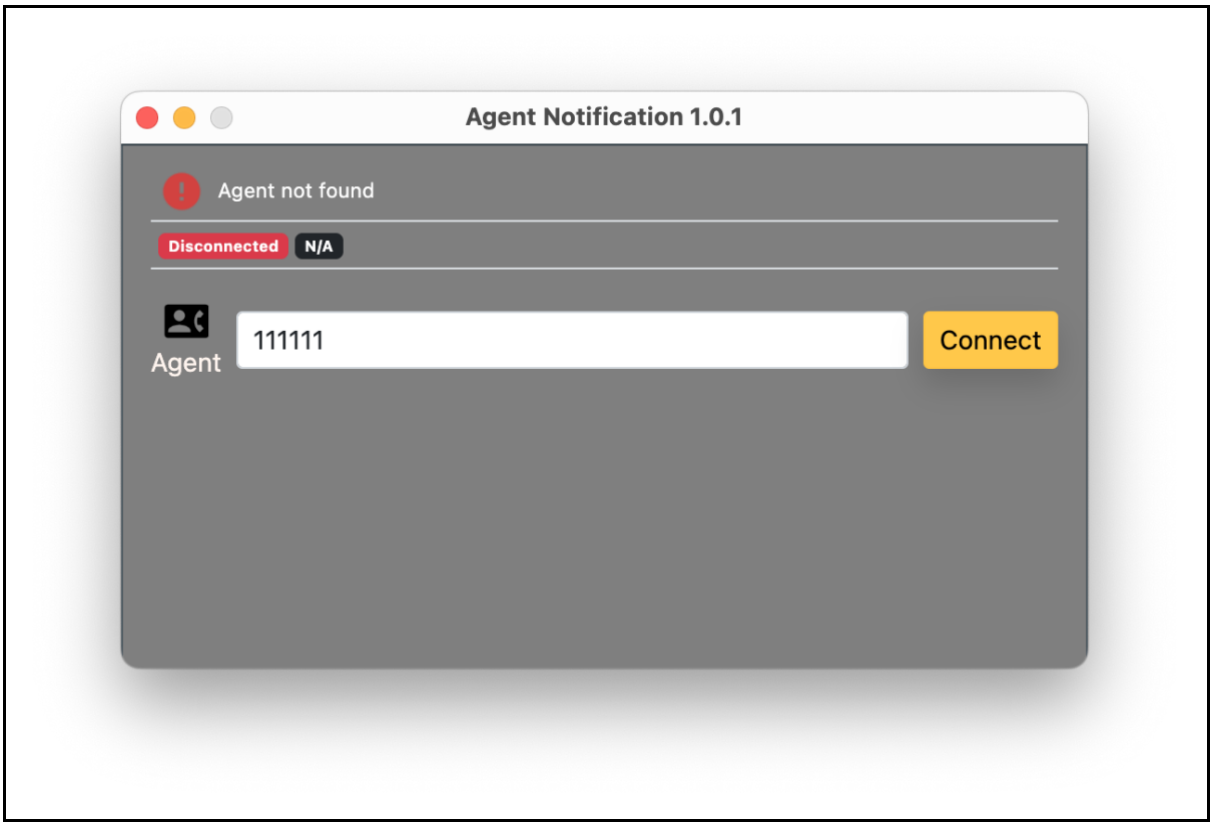


และเมื่อกด disconnect agent_notification ที่ Wallboard agentcode 1234 จะหายไป

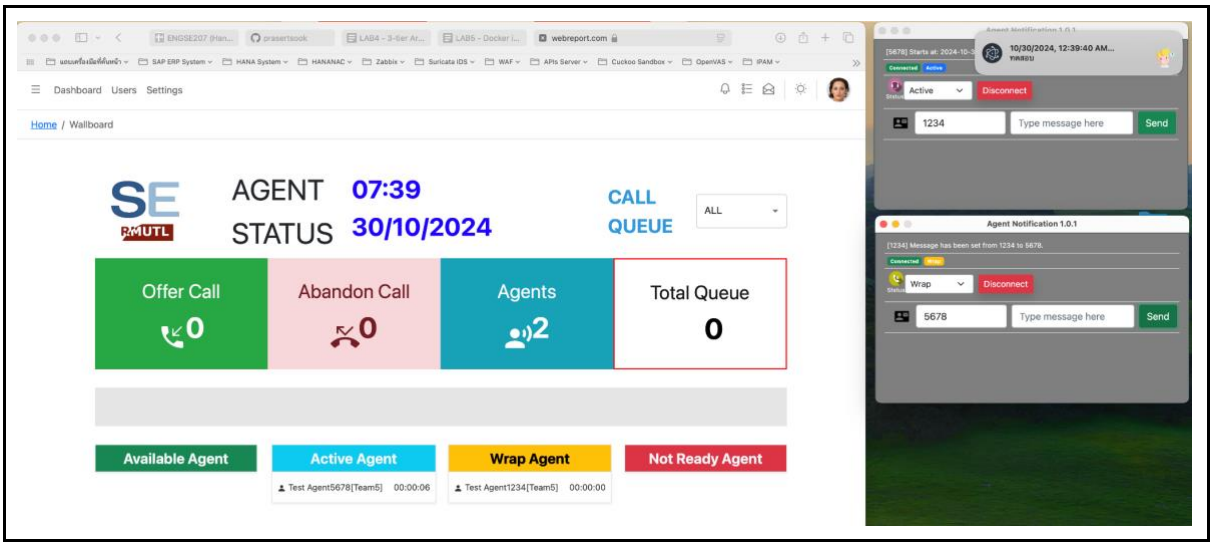
และเมื่อไปเช็ค endpoint-api จะเห็นว่า "IsLogin": "0" คือ logout ออกจากระบบ



ทดสอบพยายาม connect กับ AgenCode ที่ไม่มีอยู่ในระบบ ตัว agent_notification จะวิ่งไปเช็คที่ endpoint-api



ทดสอบส่งข้อความ



Link VDO ทดสอบการใช้งาน: LAB5 - Docker implementation.mov