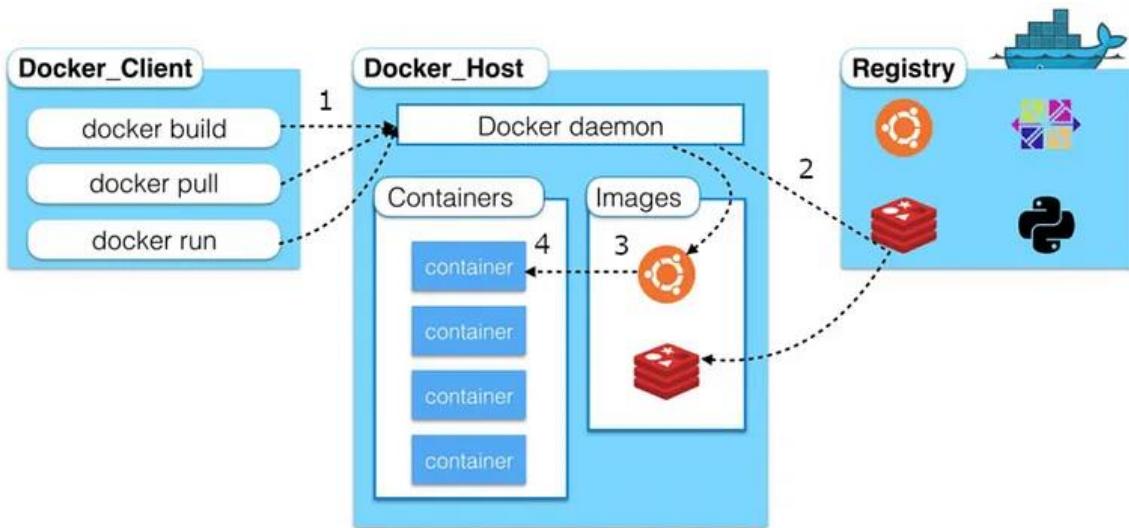


LAB5 - Docker implementation
66543210040-2 Prasert Sooksrikaew
66543210038-6 Chinchettha Namayom



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)

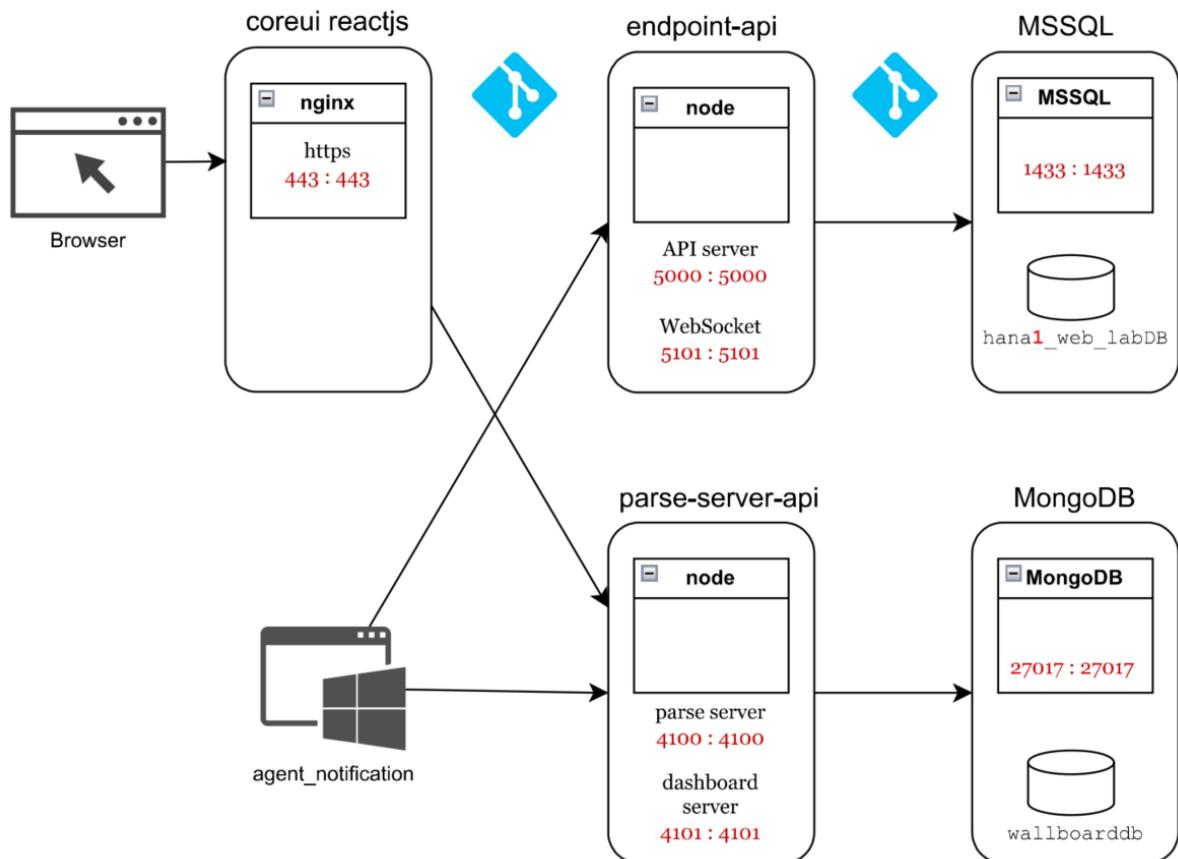
ติดตั้ง Ubuntu Server ใน VM จะเป็น Version 20.04 หรือ 22.04 ก็ได้
ถ้ากำลังลงใหม่ให้สร้าง user ตามนี้ตอนติดตั้ง หรือ ถ้าไครสร้าง VM แล้ว ให้ทำการ remote login โดย [Terminus \(download ที่นี่\)](#)

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 repository นี้ลงมา (Branch: main)

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

ภาพสำเร็จ ที่ต้องทำส่ง



Final Task

ให้แต่ละ Team ส่งงาน final ผ่าน slack โดยส่งเป็น direct message มา에게รับ
กำหนดส่งเป็น ลิ้ง final git repository โดยทำการสร้าง repository สำหรับ final ขึ้นมาใหม่
ในนั้นจะประกอบไปด้วย

1. มี **Readme.md** สำหรับบอกชื่อคุณและนามชีกของคุณ
2. ไฟล์เอกสาร PDF อธิบายงาน
3. Directory สำหรับเก็บไฟล์ script และ source code ที่บรรจุลงไว้ในแต่ละตอนเทนเนอร์
4. มี link video ทดสอบการใช้งานของงาน อยู่ในไฟล์ **Readme.md** ด้วยครับ

วิธีการเขียน Dockerfile เป็นง่าย

<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://iqokuz.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>

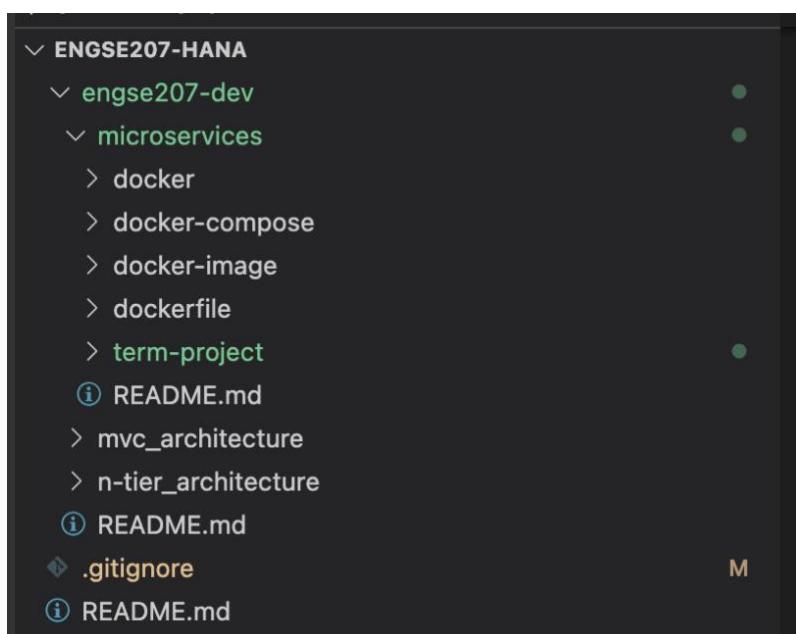
Dockerfile tutorial by example - basics and best practices

<https://takacsmark.com/dockerfile-tutorial-by-example-dockerfile-best-practices-2018/>

Dockerfile Implementation

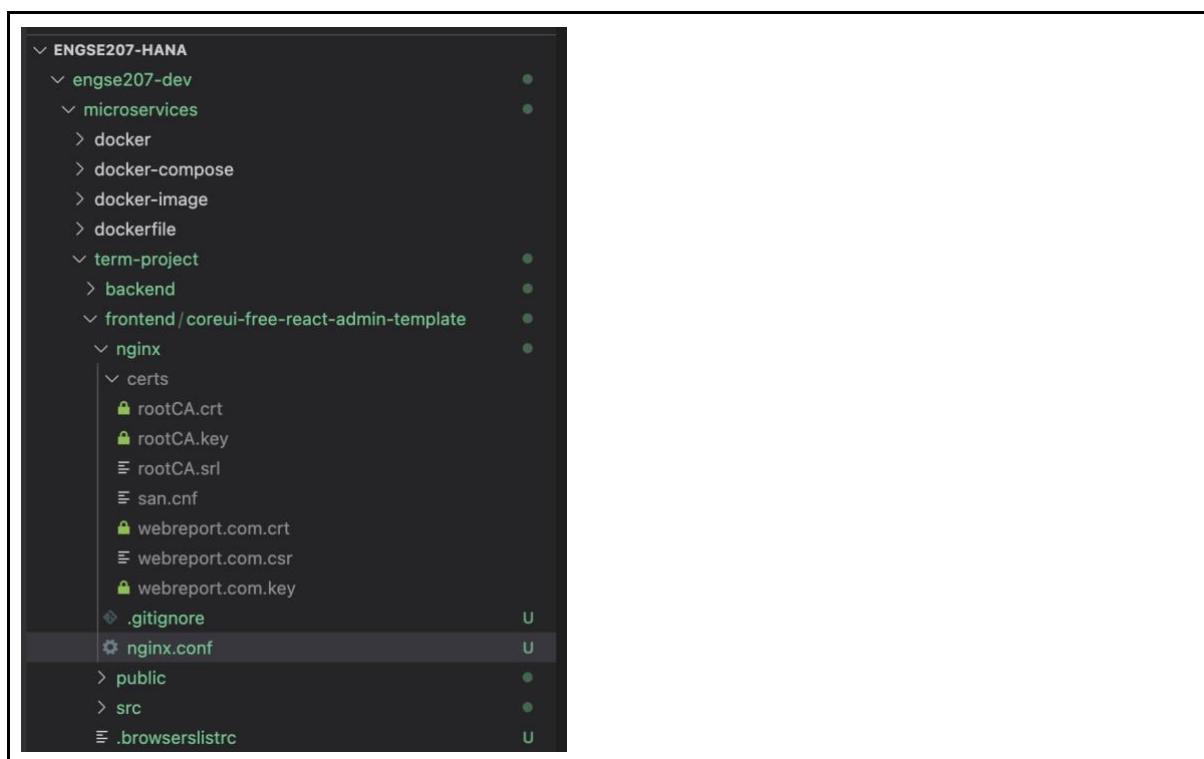
Frontend → coreui reactjs

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

```
1  # /etc/nginx/nginx.conf
2
3  events {
4      worker_connections 1024;
5  }
6
7  http {
8      include /etc/nginx/mime.types;
9      default_type application/octet-stream;
10
11     # Server block for HTTPS
12     server {
13         listen 443 ssl;
14         server_name webreport.com www.webreport.com;
15
16         ssl_certificate /etc/nginx/ssl/webreport.com.crt;
17         ssl_certificate_key /etc/nginx/ssl/webreport.com.key;
18
19         ssl_protocols TLSv1.2 TLSv1.3;
20         ssl_ciphers HIGH:!aNULL:!MD5;
21         ssl_prefer_server_ciphers on;
22
23         root /usr/share/nginx/html;
24         index index.html;
25
26         location / {
27             try_files $uri $uri/ =404;
28         }
29     }
30
31     # Redirect HTTP to HTTPS
32     server {
33         listen 80;
34         server_name webreport.com www.webreport.com;
35         return 301 https://$host$request_uri;
36     }
37 }
38 }
```

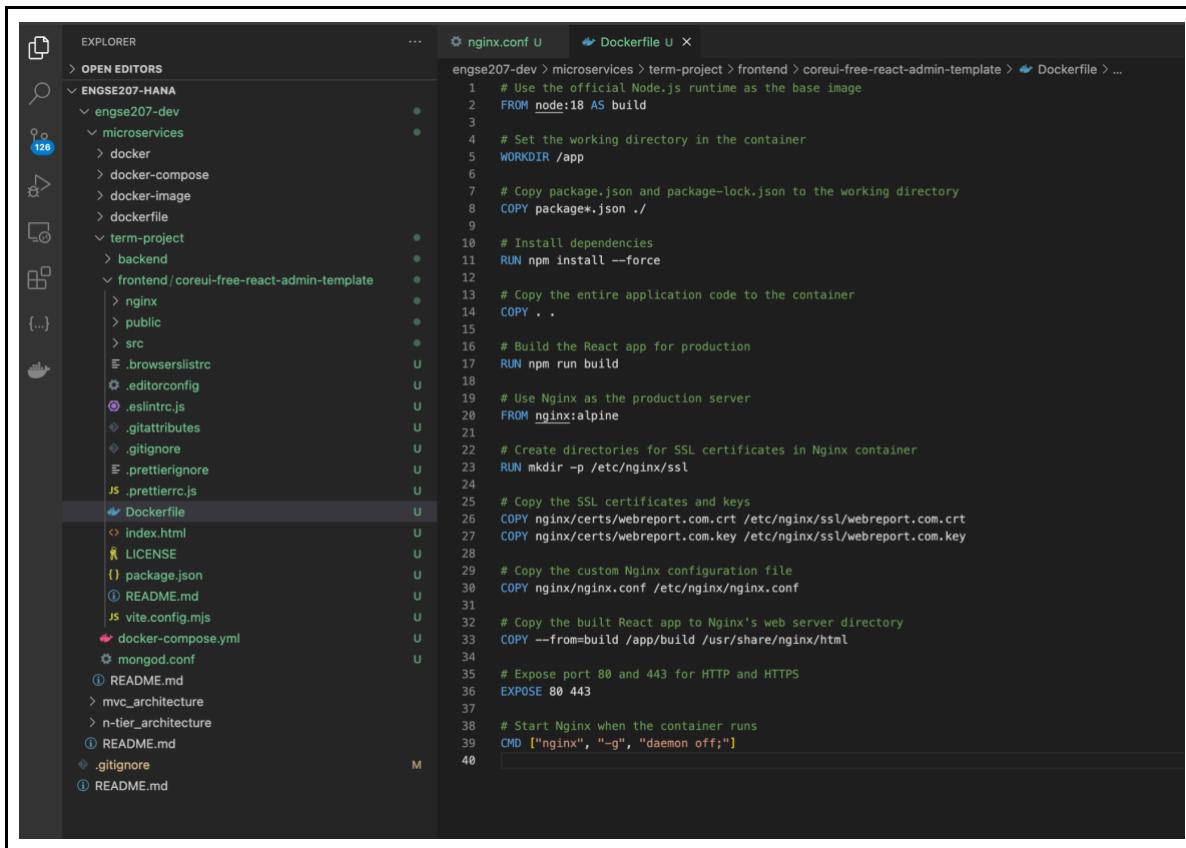
แก้ไข wbconfig.js เพื่อให้เข้าไปที่ parse-server-api

```
engse207-dev > microservices > term-project > frontend > coreui-free-react-admin-template > src > JS wbconfig.js > ...
1  const wbconfig = {
2      development: {
3          hosturl : 'https://parse-server-api:4100/api',
4          wsurl   : 'wss://parse-server-api:4100',
5          masterKey : 'wallboardapi',
6          clientKey : 'wallboardapi',
7          javascriptKey : 'wallboardapi',
8          appId : "wallboardapi"
9      },
10     production: {
11         hosturl : 'https://lab-parse-server.se-rmutil.net/api',
12         wsurl   : 'wss://lab-parse-server.se-rmutil.net',
13         masterKey : 'wallboardapi',
14         clientKey : 'wallboardapi',
15         javascriptKey : 'wallboardapi',
16         appId : "wallboardapi"
17     }
18 };
19 export default wbconfig;
20 |
```

แก้ไข Wallboard.js เพื่อให้นำดึง 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 | X
21   constructor(props) {
22     super(props);
23   }
24 }
```

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



จากนั้นรัน command เพื่อ build image สำหรับ coreui

```
#docker build -t coreui:0.1 .
```

```
PROBLEMS OUTPUT TERMINAL PORTS COMMENTS DEBUG CONSOLE
=> [stage-1 3/6] COPY nginx/certs/webreport.com.crt /etc/nginx/ssl/webreport.com.crt
=> CACHED [build 3/6] WORKDIR /app
=> CACHED [build 3/6] COPY package.json .
=> [build 4/6] RUN npm install --force
=> [build 5/6] COPY .
=> [stage-1 4/6] COPY nginx/certs/webreport.com.key /etc/nginx/ssl/webreport.com.key
=> [stage-1 5/6] COPY nginx/nginx.conf /etc/nginx/nginx.conf
=> [build 6/6] RUN npm run build
=> [stage-1 6/6] COPY --from=build /app/build /usr/share/nginx/html
=> exporting to image
=> exporting layers
=> writing image sha256:915f5b6d0624f5b0389fd77e61b1d58932faf527477eaee09b020ce38f7bf55b
=> > naming to docker.io/library/coreui:0.1
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/xts0m7kjbt8tornfpqf47ggr

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview
○ prasert@MacBook coreui-free-react-admin-template %
○ prasert@MacBook coreui-free-react-admin-template %
○ prasert@MacBook coreui-free-react-admin-template %
● prasert@MacBook coreui-free-react-admin-template % ls -l
total 56
-rw-r--r-- 1 prasert staff 1042 Oct 28 10:05 Dockerfile
-rw-r--r--@ 1 prasert staff 1097 Oct 28 08:24 LICENSE
-rw-r--r-- 1 prasert staff 8049 Oct 28 08:24 README.md
-rw-r--r-- 1 prasert staff 1153 Oct 28 08:24 index.html
drwxr-xr-x 6 prasert staff 192 Oct 28 09:12 nginx
-rw-r--r-- 1 prasert staff 2027 Oct 28 08:24 package.json
drwxr-xr-x 4 prasert staff 102 Oct 28 08:24 public
drwxr-xr-x 13 prasert staff 416 Oct 28 08:24 src
-rw-r--r-- 1 prasert staff 1037 Oct 28 08:24 vite.config.mjs
○ prasert@MacBook coreui-free-react-admin-template %
○ prasert@MacBook coreui-free-react-admin-template %
● prasert@MacBook coreui-free-react-admin-template % docker container run -d -p 443:443 --name coreui coreui:0.1
383b9c0b40cab4ba96ebdb0f17a408309bb3b8a014be0e652aa7c6c2e9af0
○ prasert@MacBook coreui-free-react-admin-template %
○ prasert@MacBook coreui-free-react-admin-template %
○ prasert@MacBook coreui-free-react-admin-template %
● prasert@MacBook coreui-free-react-admin-template % docker container ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
383b9c0b40ca coreui:0.1 "/docker-entrypoint..." 4 minutes ago Up 4 minutes 80/tcp, 0.0.0.0:443->443/tcp coreui
● prasert@MacBook coreui-free-react-admin-template % docker container ls -a
REPOSITORY TAG IMAGE ID CREATED SIZE
coreui 0.1 915f5b6d0624 5 minutes ago 59.4MB
nginx alpine 577a23b585b 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 coreui-free-react-admin-template %
```

เช็ค image ที่สร้างใช้ command

```
#docker image ls -a
```

```
prasert@MacBook coreui-free-react-admin-template %
● prasert@MacBook coreui-free-react-admin-template % docker image ls -a
REPOSITORY TAG IMAGE ID CREATED SIZE
coreui 0.1 2d4d5245a71f About an hour ago 59.4MB
nginx alpine 577a23b585b 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 coreui-free-react-admin-template %
○ prasert@MacBook coreui-free-react-admin-template %
```

จากนั้นสร้าง container ชื่อ coreui และรันกับ image นี้

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

```
prasert@MacBook coreui-free-react-admin-template %
● prasert@MacBook coreui-free-react-admin-template % docker container run -d -p 443:443 --name coreui coreui:0.1
c4cbd821b11669754dbb25a2132bd308b75b8df841d44e12a66d129bf9262c
○ prasert@MacBook coreui-free-react-admin-template %
● prasert@MacBook coreui-free-react-admin-template % docker container ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c4cbd821b116 coreui:0.1 "/docker-entrypoint..." 6 seconds ago Up 5 seconds 80/tcp, 0.0.0.0:443->443/tcp coreui
○ prasert@MacBook coreui-free-react-admin-template %
○ prasert@MacBook coreui-free-react-admin-template %
```

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;
        }
    }

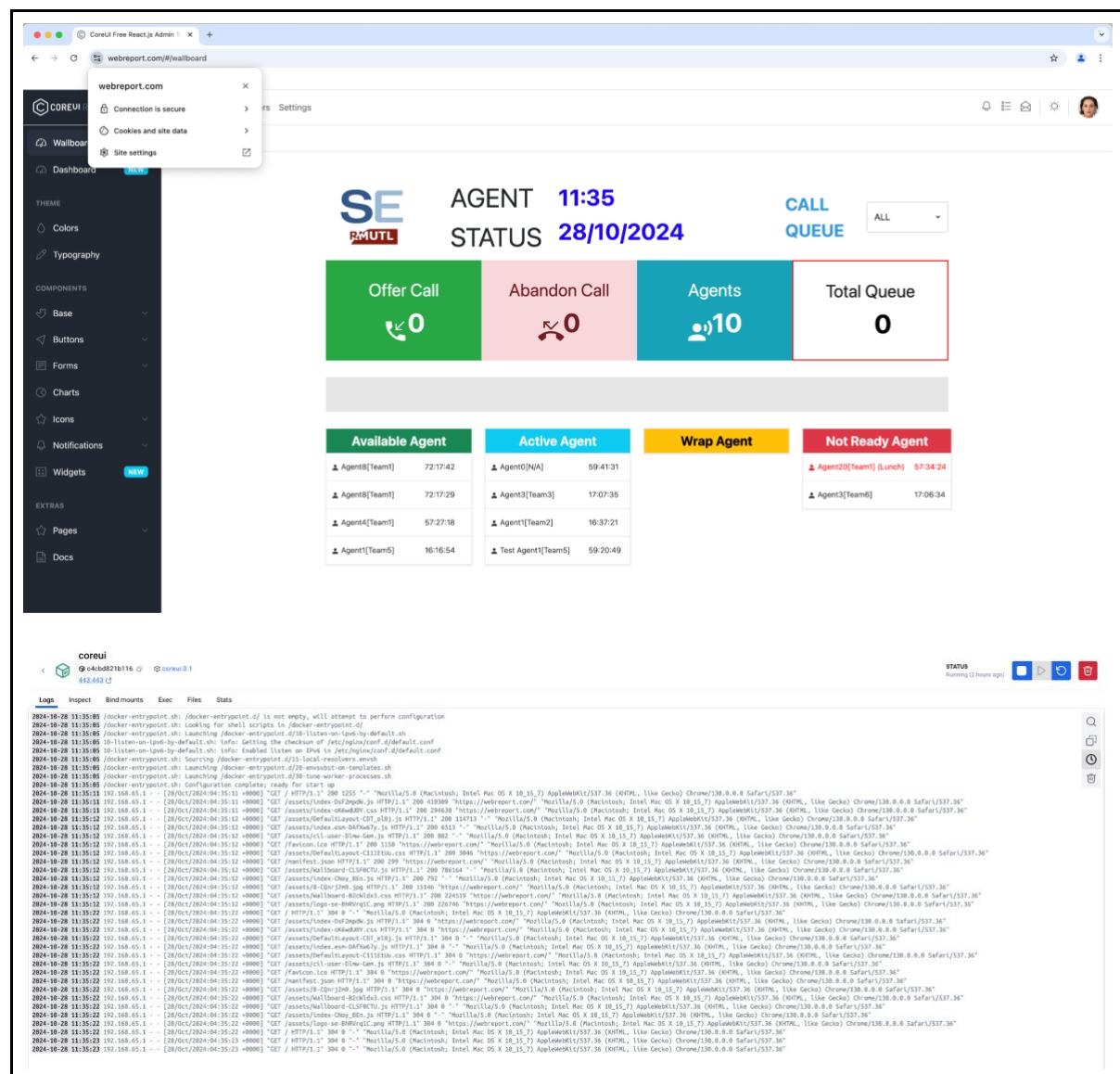
    # Redirect HTTP to HTTPS
    server {
        listen 80;
        server_name webreport.com www.webreport.com;
        return 301 https://$host$request_uri;
    }
}
/ # ls -l /etc/nginx/ssl/
total 8
-rw-r--r--  1 root      root          1696 Oct 28 03:36
webreport.com.crt
-rw-----  1 root      root          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

```
term-project
  > backend
  > database
    > mongodb
  > mssql
    > DB_Script
      > dbo.agent.Table.sql
      > dbo.didagent.Table.sql
      > dbo.DIDQueues.Table.sql
      > dbo.OnlineAgents.Table.sql
      > dbo.PermissionUsers.Table.sql
      > dbo.Projects.Table.sql
      > dbo.ReportLogs.Table.sql
      > dbo.tier_lines.Table.sql
      > dbo.tier_teams_lines.Table.sql
      > dbo.tier_teams.Table.sql
      > dbo.user_groups.Table.sql
      > dbo.Users.Table.sql
      > dbo.Wallboard.Table.sql
    > Dockerfile
  $ init-db.sh
```

สร้าง shell script init-db.sh สำหรับ import DB

```
engse207-dev > microservices > term-project > database > mssql > $ init-db.sh
1  #!/bin/bash
2
3  # Start SQL Server in the background
4 /opt/mssql/bin/sqlserver &
5
6  # Wait for SQL Server to start
7 echo "Waiting for SQL Server to start..."
8 sleep 30s
9
10 # Check if this is the first initialization by looking for the .initialized file
11 if [ ! -f /var/opt/mssql/data/.initialized ]; then
12   # Create the database
13   echo "Creating database hana5_web_labDB..."
14   /opt/mssql-tools18/bin/sqlcmd -S localhost -U SA -P "$MSSQL_SA_PASSWORD" -Q "CREATE DATABASE hana5_web_labDB" -N -C
15   if [ $? -eq 0 ]; then
16     echo "Database hana5_web_labDB created successfully."
17   else
18     echo "Failed to create database hana5_web_labDB." >&2
19     exit 1
20   fi
21
22  # Run SQL scripts in the /docker-entrypoint-initdb.d directory within the created database
23  echo "Running SQL scripts in /docker-entrypoint-initdb.d/..."
24  for entry in /docker-entrypoint-initdb.d/*.sql
25  do
26    echo "Executing script: $entry on hana5_web_labDB..."
27    /opt/mssql-tools18/bin/sqlcmd -S localhost -U SA -P "$MSSQL_SA_PASSWORD" -d hana5_web_labDB -i "$entry" -N -C
28    if [ $? -eq 0 ]; then
29      echo "$entry ran successfully."
30    else
31      echo "Error running $entry" >&2
32      exit 1
33    fi
34  done
35
36  # Create a file to indicate the database has been initialized
37  touch /var/opt/mssql/data/.initialized
38 else
39  echo "Database already initialized. Skipping initialization scripts."
40 fi
41
42 # Keep the container running in the foreground
43 wait
```

สร้าง Dockerfile สำหรับ build image mssql

```
engse207-dev > microservices > term-project > database > mssql > 📄 Dockerfile > ...
1  # Dockerfile for mssql (SQL Server)
2  FROM mcr.microsoft.com/mssql/server:2022-latest
3
4  # Set environment variables for SQL Server
5  ENV ACCEPT_EULA=Y
6  ENV MSSQL_SA_PASSWORD=Sa@2024%
7
8  # Copy the DB script to the init directory
9  COPY DB_Script/*.sql /docker-entrypoint-initdb.d/
10
11 # Copy the init-db.sh script to the init directory
12 COPY init-db.sh /docker-entrypoint-initdb.d/init-db.sh
13
14 # Expose SQL Server port
15 EXPOSE 1433
16
17 # Run the initialization script as the entrypoint
18 CMD [ "/bin/bash", "/docker-entrypoint-initdb.d/init-db.sh" ]
19 |
```

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 .dockignore
=> => 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
```

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	004f8fb0b46c	3 months ago	1.59GB
node	18	b2c3c88b29ee	3 months ago	1.09GB
mongo	4.4	80d502872ebd	8 months ago	408MB

จากนั้นล็งรัน container mssql

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

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.s..."	6 hours ago	Up 6 hours	80/tcp, 0.0.0.0:443->443/tcp	coreui

ทดสอบใช้ Azure Data Studio query ดูข้อมูลว่ามี import แล้วหรือยัง

The screenshot shows the Azure Data Studio interface with a query results grid. The query is:

```
SELECT TOP (1000) [agent_id]
      ,[agent_code]
      ,[gender]
      ,[name]
      ,[lastname]
      ,[tel]
      ,[AGDescription]
      ,[datecreate]
      ,[agent_status]
      ,[tt_id]
      ,[Tier_Team]
      ,[Is_Login]
      ,[Is_Logout]
      ,[Is_Wrap]
      ,[Is_Available]
      ,[Is_Not_Ready]
```

The results grid displays 20 rows of data from the 'agent' table. The columns correspond to the selected fields in the query. The data includes various agent details such as name, gender, telephone number, and status information.

Backend → endpoint-api

สร้าง directory backend จากนั้น copy server cert และ endpoint-api ไว้ภายใน
และทำการเปลี่ยน API Port: 5000 WebSocket Port: 5101

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "engse207-HANA". The "backend" folder contains "endpoint-api" which has "repository", ".gitignore", "apiconfig.js", "Dockerfile", "package.json", "server.crt", "server.key", "sqlConfig.js", and "webreport-api-https.js".
- Dockerfile:** The Dockerfile is visible in the background.
- Code Editor:** The file "webreport-api-https.js" is open. The code defines an Hapi API and sets up a WebSocket server on port 5101, mapping it to process.env.PORT || 5101. It also logs connection details and database query results.

```
const hapi = require('@hapi/hapi');
let express = require('express');
const AuthBearer = require('hapi-auth-bearer-token');
let fs = require('fs');
let cors = require('cors');

const OnlineAgent = require('./repository/OnlineAgent');
const apiconfig = require('./apiconfig')['development'];

process.env.NODE_TLS_REJECT_UNAUTHORIZED = "1";

const apiport = 5000

var url = require('url');

//----- Websocket Part1 Start -----
var webSocketServer = new (require('ws')).Server({
  port: (process.env.PORT || 5101)
}),
  clientWebSockets = {} // userID: webSocket
CLIENTS = [];

webSocketServer.on('connection', (ws, req) => {
  var q = url.parse(req.url, true);

  console.log(q.host);
  console.log(q.pathname);
  console.log(q.search);

  var qdata = q.query; //returns an object: { year: 2017, month: 'february' }

  console.log("----- webSocketServer -----");
  console.log("AgentCode: " + qdata.agentcode);
});
```

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

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "engse207-dev". The "backend" folder contains "endpoint-api" which has "sqlConfig.js".
- Code Editor:** The file "sqlConfig.js" is open. It defines a dbconfig object with development and production environments. Both environments use 'mssql' as the server, 'hana5_web_labDB' as the database, 'sa' as the user, and 'Sa@2024%' as the password. The port is 1433 for both. The options for encryption, timeout, and other connection parameters are identical for both environments.

```
var dbconfig = {
  development: {
    server: 'mssql',
    database:'hana5_web_labDB',
    user:'sa',
    password:'Sa@2024%',
    port: 1433,
    options:{
      encrypt: true,
      setTimeout: 12000,
      enableArithAbort: true,
      trustServerCertificate: true,
      trustedconnection: true,
      instancename: 'mssql' // SQL Server instance name
    }
  },
  production: {
    server: 'mssql',
    database:'hana5_web_labDB',
    user:'sa',
    password:'Sa@2024%',
    port: 1433,
    options:{
      encrypt: true,
      setTimeout: 12000,
      enableArithAbort: true,
      trustServerCertificate: true,
      trustedconnection: true,
      instancename: 'mssql' // SQL Server instance name
    }
  }
};

module.exports = dbconfig;
```

สร้าง Script Dockerfile สำหรับ endpoint-api

```
1  # Dockerfile for endpoint-api (Node.js)
2  FROM node:18
3
4  # Set working directory
5  WORKDIR /app
6
7  # Copy package files and install dependencies
8  COPY package*.json .
9  RUN npm install
10
11 # Copy application source code
12 COPY .
13
14 # Copy SSL certificates to the container (if necessary for HTTPS)
15 COPY server.crt /app/server.crt
16 COPY server.key /app/server.key
17
18 # Expose ports for the endpoint-api
19 EXPOSE 5000
20 EXPOSE 5101
21
22 # Start the application
23 CMD ["node", "webreport-api-https.js"]
24
```

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: <https://dockerscanner.com/builds/1ff8ac7901e55125347ce66ab68538b9c5eae69baaf3f57ad325bf97d923384d>

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	2dd45245a71f	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
```

```
cd716e7b53bd2f14b7c7328ecaca7ef7759b83ce42c0bc028dbc8d293641b26
```

```
o prasert@MacBook endpoint-api %
```

```
o 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

```
o prasert@MacBook endpoint-api %
```

จากนั้นใช้ Postman test API connection

The screenshot shows the Postman application interface. At the top, it says "HTTP ENGSE207 / Hello". Below that is a search bar with "GET" selected and the URL "https://endpoint-api:5000/". To the right of the search bar are "Save" and "Share" buttons. Underneath the search bar, there are tabs for "Params", "Authorization", "Headers (8)", "Body", "Scripts", and "Settings". The "Headers" tab is currently active. It shows a table with one row and two columns: "Key" and "Value". The "Key" column contains "Key" and the "Value" column contains "Value". To the right of the table are "Description", "Bulk Edit", and "Presets" buttons. Below the header section, there are tabs for "Body", "Cookies", "Headers (10)", and "Test Results". The "Test Results" tab is active and shows a green "200 OK" status bar with "27 ms", "375 B", and a globe icon. Below the status bar, there are buttons for "Pretty", "Raw", "Preview", "Visualize", and "HTML". The "HTML" button is currently selected and has a dropdown arrow. To the right of these buttons are "Copy" and "Search" icons. The main content area displays the response body: "1 Test Hello, from Endpoint Web Report API."

The screenshot shows the Postman interface with the following details:

- URL:** https://endpoint-api:5000/api/v1/getOnlineAgentByAgentCode?agentcode=9999
- Method:** GET
- Query Params:**

Key	Value	Description
agentcode	9999	
Key	Value	Description
- Body:** JSON response (Pretty):


```

1 {
2   "error": false,
3   "statusCode": 200,
4   "data": [
5     {
6       "OnlineAgent_id": 1,
7       "agent_id": null,
8       "agent_code": "9999",
9       "DIDQueueID": null,
10      "StartOnline": "2023-08-31T13:56:26.033Z",
11      "uuid": "cb40f193-1317-4b17-a360-506cd89700f7",
12      "AgentName": "Dev2",
13      "AgentStatus": "1",
14      "IsLogin": "1",
15      "LastUpdated": "2023-08-31T13:56:26.033Z"
16    }
  ]
}
      
```
- Status:** 200 OK
- Time:** 69 ms
- Size:** 635 B

Database → MongoDB

สร้าง directory mongodb สำหรับเก็บไฟล์ wallboarddb-init.js และ Dockerfile

The screenshot shows a file explorer interface with the following directory structure:

```

ENGSE207-HANA
  engse207-dev
    microservices
      docker
      docker-compose
      docker-image
      dockerfile
    term-project
      backend
      database
        mongodb
          Dockerfile
          wallboarddb-init.js
        mssql
      frontend
  
```

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

```
engse207-dev > microservices > term-project > database > mongodb > JS wallboarddb-init.js > ...
1  // Switch to wallboarddb database and create wallboarduser
2  db = db.getSiblingDB('wallboarddb');
3  db.createUser({
4      user: "wallboarduser",
5      pwd: "WB1qazxsw2",
6      roles: [{ role: "readWrite", db: "wallboarddb" }]
7  });
8
9  // Switch to admin database and create wallboardadmin
10 db = db.getSiblingDB('admin');
11 db.createUser({
12     user: "wallboardadmin",
13     pwd: "WB1qazxsw2",
14     roles: [
15         { role: "userAdminAnyDatabase", db: "admin" },
16         "readWriteAnyDatabase"
17     ]
18 });
19 |
```

สร้างไฟล์ Dockerfile สำหรับ build image MongoDB

```
engse207-dev > microservices > term-project > database > mongodb > Dockerfile > ...
1  # Use MongoDB 4.4 official image
2  FROM mongo:4.4
3
4  # Set environment variables for root user creation
5  ENV MONGO_INITDB_ROOT_USERNAME=admin
6  ENV MONGO_INITDB_ROOT_PASSWORD=@se2024
7  ENV MONGO_INITDB_DATABASE=wallboarddb
8
9  # Copy the custom-init.js script to the init directory
10 COPY wallboarddb-init.js /docker-entrypoint-initdb.d/wallboarddb-init.js
11
12 # Expose MongoDB port
13 EXPOSE 27017
14 |
```

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: 41B
=> 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 .dockerrcignore
=> => 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      2dd45245a71f  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 ที่ 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 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
```

```
PROBLEMS    OUTPUT    TERMINAL    PORTS    COMMENTS    DEBUG CONSOLE

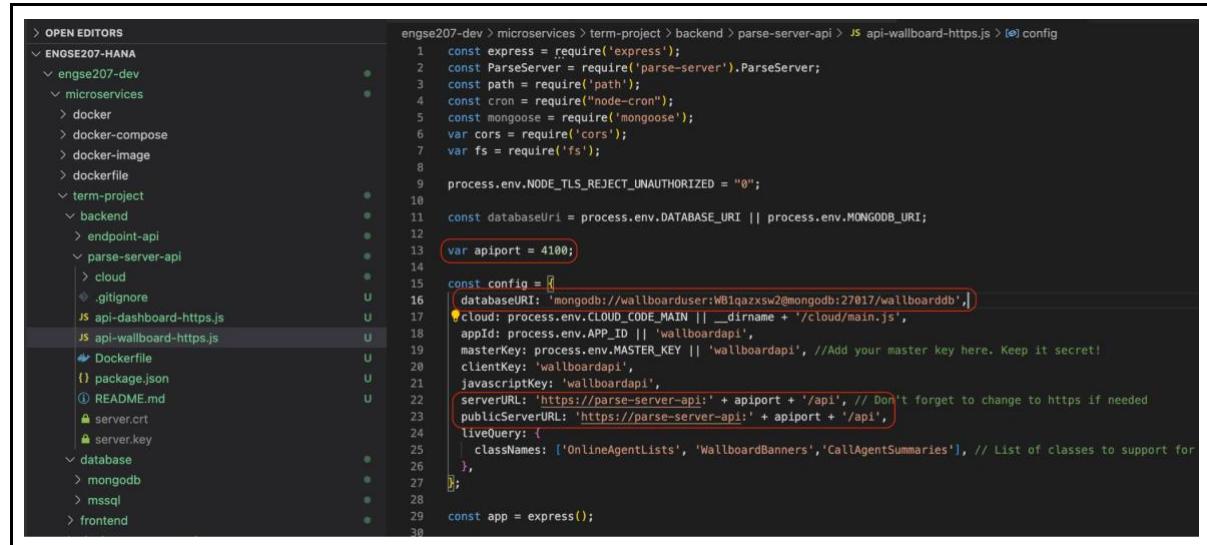
root@0e9fcc5091b7:/#
root@0e9fcc5091b7:/# mongo -u wallboardadmin -p --authenticationDatabase admin
MongoDB shell version v4.4.29
Enter password:
connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("68701368-8309-4962-9085-498f37106084") }
MongoDB server version: 4.4.29
> use admin
switched to db admin
> show users
{
  "_id" : "admin.admin",
  "userId" : UUID("47066894-4afc-4407-b650-667f35e0d77a"),
  "user" : "admin",
  "db" : "admin",
  "roles" : [
    {
      "role" : "root",
      "db" : "admin"
    }
  ],
  "mechanisms" : [
    "SCRAM-SHA-1",
    "SCRAM-SHA-256"
  ]
}
{
  "_id" : "admin.wallboardadmin",
  "userId" : UUID("d49ddf03-1a23-49eb-a705-e85b57e17763"),
  "user" : "wallboardadmin",
  "db" : "admin",
  "roles" : [
    {
      "role" : "userAdminAnyDatabase",
      "db" : "admin"
    },
    {
      "role" : "readWriteAnyDatabase",
      "db" : "admin"
    }
  ],
  "mechanisms" : [
    "SCRAM-SHA-1",
    "SCRAM-SHA-256"
  ]
}
> []
```

Backend → parse-server-api

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

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

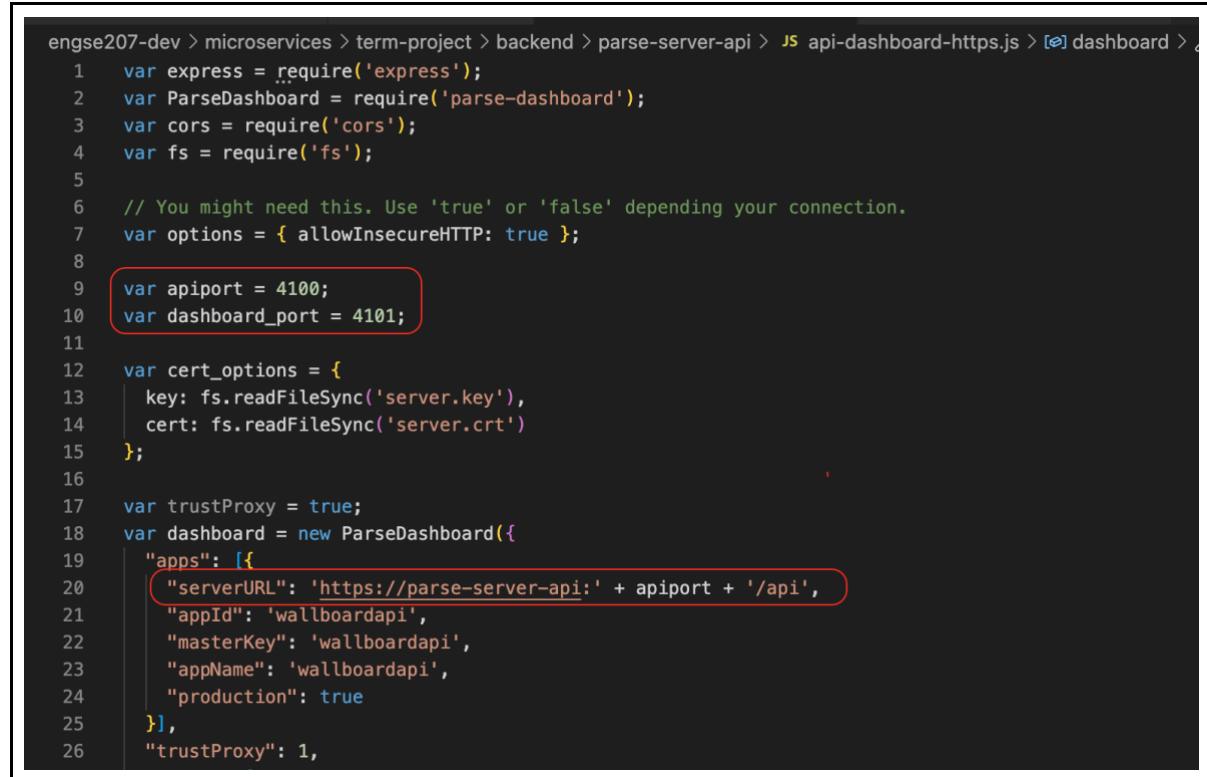
และ mongodb connection



```
engse207-dev > microservices > term-project > backend > parse-server-api > JS api-wallboard-https.js > config
1 const express = require('express');
2 const ParseServer = require('parse-server').ParseServer;
3 const path = require('path');
4 const cron = require("node-cron");
5 const mongoose = require('mongoose');
6 var cors = require('cors');
7 var fs = require('fs');
8
9 process.env.NODE_TLS_REJECT_UNAUTHORIZED = "0";
10
11 const databaseUri = process.env.DATABASE_URI || process.env.MONGODB_URI;
12
13 var apiport = 4100;
14
15 const config = [
16   {
17     databaseURI: 'mongodb://wallboarduser:WBlqazxsw2@mongodb:27017/wallboarddb',
18     cloud: process.env.CLOUD_CODE_MAIN || __dirname + '/cloud/main.js',
19     appId: process.env.APP_ID || 'wallboardapi',
20     masterKey: process.env.MASTER_KEY || 'wallboardapi', //Add your master key here. Keep it secret!
21     clientKey: 'wallboardapi',
22     javascriptKey: 'wallboardapi',
23     serverURL: 'https://parse-server-api:' + apiport + '/api', // Don't forget to change to https if needed
24     publicServerURL: 'https://parse-server-api:' + apiport + '/api',
25     liveQuery: {
26       classNames: ['OnlineAgentLists', 'WallboardBanners', 'CallAgentSummaries'], // List of classes to support for live query.
27     },
28   },
29 ];
30
31 const app = express();
```

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

Port 4101 และ Server URL



```
engse207-dev > microservices > term-project > backend > parse-server-api > JS api-dashboard-https.js > dashboard > config
1 var express = require('express');
2 var ParseDashboard = require('parse-dashboard');
3 var cors = require('cors');
4 var fs = require('fs');
5
6 // You might need this. Use 'true' or 'false' depending your connection.
7 var options = { allowInsecureHTTP: true };
8
9 var apiport = 4100;
10 var dashboard_port = 4101;
11
12 var cert_options = {
13   key: fs.readFileSync('server.key'),
14   cert: fs.readFileSync('server.crt')
15 };
16
17 var trustProxy = true;
18 var dashboard = new ParseDashboard({
19   "apps": [
20     {
21       "serverURL": 'https://parse-server-api:' + apiport + '/api',
22       "appId": 'wallboardapi',
23       "masterKey": 'wallboardapi',
24       "appName": 'wallboardapi',
25       "production": true
26     },
27     "trustProxy": 1,
28   }
29 };
```

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

```
engse207-dev > microservices > term-project > backend > parse-server-api > $ setup-classes.sh
1  #!/bin/sh
2
3  echo "Waiting for Parse Server to start..."
4  sleep 30s
5
6  # Check if the setup has already been run
7  if [ -f "/app/class-setup-done" ]; then
8      echo "Classes have already been created. Skipping setup."
9      exit 0
10 fi
11
12 # Define Parse Server URL and credentials
13 PARSE_SERVER_URL="https://parse-server-api:4100/api/schemas"
14 APP_ID="wallboardapi"
15 MASTER_KEY="wallboardapi"
16
17 # Create each class if it hasn't been created before
18 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
19 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
20 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
21
22 # Create the status file to prevent re-running
23 touch /app/class-setup-done
24 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 > ...
1  # Dockerfile for parse-server-api (Node.js)
2  FROM node:18
3
4  # Set working directory
5  WORKDIR /app
6
7  # Copy package files and install dependencies
8  COPY package*.json .
9  RUN npm install
10
11 # Copy application source code
12 COPY .
13
14 # Copy SSL certificates to the container (if necessary for HTTPS)
15 COPY server.crt /app/server.crt
16 COPY server.key /app/server.key
17
18 # Copy the setup script into the container
19 COPY setup-classes.sh /app/setup-classes.sh
20
21 # Ensure setup script has executable permissions
22 RUN chmod +x /app/setup-classes.sh
23
24 # Expose ports for the backend
25 EXPOSE 4100
26 EXPOSE 4101
27
28 # Run the setup script on the first run and start the applications
29 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

=> [internal] load metadata for docker.io/library/node:18
=> [internal] load .dockercignore
=> => 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->4100-4101/tcp   parse-server-api
a0cd4a6506aa        mongod: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/binperm..."   38 minutes ago   Up 38 minutes   0.0.0.0:1433->1433/tcp   mssql
70c00747853e        coreui:0.1           "/docker-entrypoint.s..."   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 ได้ถูกสร้างแล้วหรือปัง on Parse Dashboard <https://parse-server-api:4101/>

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

The screenshot shows the Parse Dashboard for the 'wallboardapi' application. On the left, there's a sidebar with 'Core' and 'Push' sections. Under 'Core', 'Browser' is selected, showing a list of classes: 'CallAgentSummaries' (selected and highlighted with a red box), 'OnlineAgentLists', and 'WallboardBanners'. The 'CallAgentSummaries' class details are shown on the right: it has 0 objects, and the columns are 'objectId' (String), 'updatedat' (Date), and 'createdat' (Date). The 'ACL' and 'ACL' buttons are visible. Below this, a large circular icon with a document symbol contains the text 'No data to display'. A small note says 'Add a row to store an object in this class.' with a 'Add a row' button.

จากนั้นใช้ Postman test API connection

The screenshot shows a Postman request for the 'parse-server-hello' collection. The method is 'POST' and the URL is 'https://parse-server-api:4100/api/functions/hello'. The 'Headers' tab is selected, showing the following configuration:

Key	Value	Description	Bulk Edit	Presets
Content-Type	application/json			
X-Parse-Application-Id	wallboardapi			
X-Parse-Master-Key	wallboardapi			
Key	Value	Description		

Below the headers, the 'Body' tab is selected, showing the response:

200 OK • 69 ms • 706 B • ⓘ • 000

Pretty Raw Preview Visualize JSON ↻ ⌂ ⌂

```
1 "result": "Hi from Parse Server"
```

HTTP 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

```
1 "result": 9
```

OnlineAgentLists - Parse Dashboard

Not Secure https://parse-server-api:4101/apps/wallboardapi/browser/OnlineAgentLists

PARSE DASHBOARD 5.4.0

Core

Browser

- _Role
- _User
- CallAgentSummaries
- OnlineAgentLists
- WallboardBanners

Webhooks

Jobs

CLASS OnlineAgentLists 1 object • Public Read and Write enabled

objectId	updatedAt	createdAt	ACL	AgentStatusCode	AgentName	Queue	AgentCode
XbtMILZQEj	29 Oct 2024 at 09:21	29 Oct 2024 at 09:21	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

```
docker-compose.yml U ×
engse207-dev > microservices > term-project > docker-compose.yml

1 version: "3.8"
2
3 services:
4   coreui:
5     image: coreui:0.1
6     container_name: coreui
7     ports:
8       - "443:443"
9     networks:
10    |   - internal_network
11
12   endpoint-api:
13     image: endpoint-api:0.1
14     container_name: endpoint-api
15     ports:
16       - "5000:5000"
17       - "5101:5101"
18     networks:
19    |   - internal_network
20
21   mssql:
22     image: mssql:0.1
23     container_name: mssql
24     environment:
25       ACCEPT_EULA: "y"
26       MSSQL_SA_PASSWORD: "Sag2024W"
27     ports:
28       - "1433:1433"
29     networks:
30    |   - internal_network
31
32   mongodb:
33     image: mongodb:0.1
34     container_name: mongodb
35     ports:
36       - "27017:27017"
37     networks:
38    |   - internal_network
39
40   parse-server-api:
41     image: parse-server-api:0.1
42     container_name: parse-server-api
43     ports:
44       - "1000:1000"
45       - "4001:4001"
46     networks:
47    |   - internal_network
48
49 networks:
50   internal_network:
51     driver: bridge

PROBLEMS OUTPUT TERMINAL PORTS COMMENTS DEBUG CONSOLE
● prasert@MacBook term-project % docker-compose up -d
WARN[0000] /Users/prasert/engse207-hana/engse207-dev/microservices/term-project/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
✓ Network term-project_internal_network
✓ Container parse-server-api
✓ Container endpoint-api
✓ Container mssql
✓ Container coreui
! Container coreui requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
○ prasert@MacBook term-project %
prasert@MacBook term-project %
```

Container list

The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. The sidebar on the left includes links for Containers, Images, Volumes, Builds, Docker Scout, and Extensions. The main area displays container usage statistics: Container CPU usage (2.67% / 400%) and Container memory usage (1.48GB / 3.74GB). A search bar and a filter option 'Only show running containers' are present. A chart titled 'Show charts' is also visible. The table lists the following containers:

Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
term-project		Running (5/5)		2.67%	20 minutes ago	[Actions]
endpoint-api	ae71a2024c2d	Running	5000:5000, 5101:5101	0%	20 minutes ago	[Actions]
mssql	453f360e5fac	Running	1433:1433	2.37%	20 minutes ago	[Actions]
coreui	845ff97141e11	Running	443:443	0%	20 minutes ago	[Actions]
mongodb	d27fc729411d	Running	27017:27017	0.3%	20 minutes ago	[Actions]
parse-server-api	88956bb5c34c	Running	4100:4100, 4101:4101	0%	20 minutes ago	[Actions]

Frontend → agent_notification Electron.js

แก้ไฟล์ index.html ส่วนของ API Server และ WebSocketServer ในชื่มมาที่ 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 WebSocketServer = "ws://192.168.64.15:3071";
41
42       //--Staging Server--
43       //var APIServer = "https://lab-api.se-rmutil.net/engse207/api/v1";
44       //var WebSocketServer = "ws://lab-ws.se-rmutil.net/engse207";
45
46       var APIServer = "https://endpoint-api:5000/api/v1";
47       var WebSocketServer = "ws://endpoint-api:5101";
48
49       var APIServer2 = "https://parse-server-api:4100/api/";
50       var WebSocketServer2 = "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;
```

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" คือ Wrap และ "IsLogin": "1"

The screenshot displays three main components:

- Wallboard:** A real-time call center dashboard showing statistics: Offer Call (0), Abandon Call (0), Agents (1), Total Queue (0). It also shows Available Agent, Active Agent, Wrap Agent, and Not Ready Agent buttons.
- API Network:** A tool for managing API endpoints. One endpoint is highlighted: `GET https://endpoint-api:5000/api/v1/getOnlineAgentByAgentCode`. The request parameters are `agentcode: 1234`. The response body is shown in JSON format:

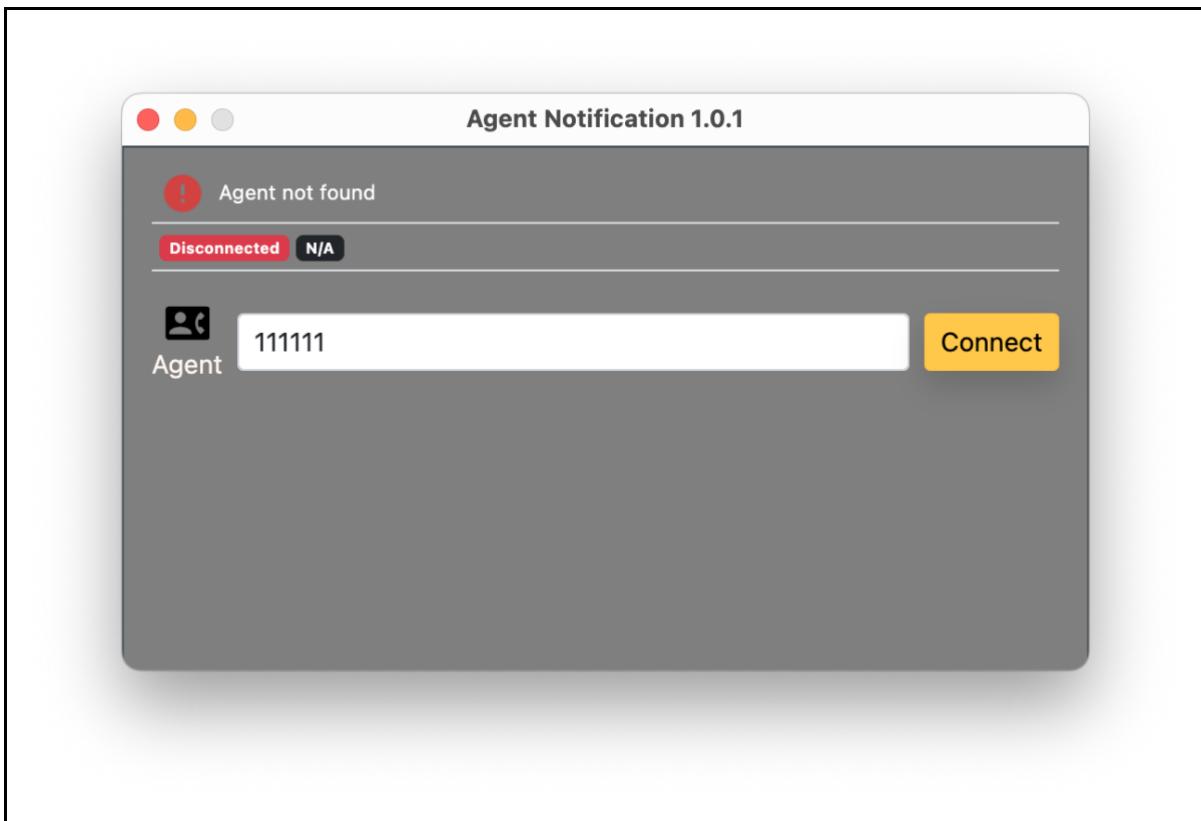
```
[{"id": 1, "agentcode": "1234", "statuscode": 200, "data": {"agent_id": 1234, "agent_name": "Test Agent1234", "agent_status": "Wrap", "is_login": 1, "startonline": "2024-10-29T11:22:49.143Z", "endonline": "2024-10-29T11:22:49.143Z", "agentstatus": "Test Agent1234", "logstatus": "Connected", "logagent": "1234", "logstatus": "Logout"}, "error": null}
```
- Agent Notification:** A small window showing a green "Connected" button and a "Disconnect" button. It also has fields for "Agent" and "Type message here".

และเมื่อกด disconnect agent_notification ที่ Wallboard agentcode 1234 จะหายไป

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

The screenshot shows the same three components as before, but the "Wrap Agent" button on the Wallboard has been clicked, changing the status to "Connected". The API Network and Agent Notification windows remain the same as in the previous screenshot.

ทดสอบพยายาม connect กับ AgenCode ที่ไม่มีอยู่ในระบบ ตัว agent_notification จะวิ่งไปเช็คที่ endpoint-api



ทดสอบส่งข้อความ

A screenshot of a web-based call center management system. At the top, it shows "AGENT 07:39" and "STATUS 30/10/2024". Below this, there are four colored boxes: green ("Offer Call 0"), pink ("Abandon Call 0"), blue ("Agents 2"), and red ("Total Queue 0"). Further down, there are four status buttons: "Available Agent" (green), "Active Agent" (blue), "Wrap Agent" (yellow), and "Not Ready Agent" (red). Below these buttons are two small boxes showing "Test Agent5678[Team5] 00:00:06" and "Test Agent1234[Team5] 00:00:00". To the right of the main interface, there is a smaller window titled "Agent Notification 1.0.1" which also shows the "Agent not found" message and the "Connected" status.

Link VDO ทดสอบการใช้งาน: [LAB5 - Docker implementation.mov](#)