# 포팅 메뉴얼

## 1. Stacks

### 1-1. Development Environment

**[Front-End]**

- Node.js: 20.11.1
- vite: ^5.0.11
- socket.io-client: ^2.5.0
- vue: ^3.4.15
- vuetify: ^3.5.8
- vite-plugin-pwa: ^0.19.2

**[Back-End]**

- Node.js:  20.11.1
- express: ^4.18.3
- socket.io: ^2.5.0
- mysql2: ^3.9.2

**[Ros]**

- socket.io : ^2.5.0

- ROS : eloquent (20200124 release)
- python : ^3.7.5
- openssl : ^1.0.2u
- choco : ^0.10.15
- rti : ^5.3.1
- opensplice : ^6.9.190403

## 1-2. Main Ribrary

**[Front-End]**

- @tosspayments/payment-widget-sdk: ^0.10.0
- axios: ^1.6.7
- pinia: ^2.1.7
- pinia-plugin-persistedstate: ^3.2.1
- vue-router: ^4.2.5

**[Back-End]**

- cors: ^2.8.5
- dotenv: ^16.4.5
- nodemon: ^3.1.0

## 1-3. Deploy Management

- AWS EC2
- Ubuntu 20.04.6 LTS
- Jenkins: ^2.444
- nginx ^1.18.0 (Ubuntu)
- Docker ^25.0.4

## 1-4. SCM(Software Configure Management)

- Git
- GitLab
- Git bash
- Gerrit

### 1-5. Community

- Mattermost
- Notion

### 1-6. Issue Management

- Jira
- Gerrit

## 2. Build & Distribute

### 2-1. Jenkins

- **Plugin 설치**
    - Docker API Plugin
    - Docker Commons Plugin
    - Docker Pipeline
    - Docker plugin
    - Gitlab API Plugin
    - GitLab Authentication plugin
    - GitLab Branch Source Plugin
    - Gitlab Merge Request Builder
    - Generic Webhook Trigger Plugin

- **Gitlab Webhook 연결**
    - url

    ```
    http://j10c109.p.ssafy.io:8080/project/mulja-pipeline
    ```

    - push event
      - wildcard pattern: release
    - commit event

- **Jenkinsfile Pipeline**
  **-**
  path: /Jenkinsfile

  ```
  pipeline {
      agent any
  ```

```
    environment{
        BACK_DOCKER_IMAGE_NAME='backend/nodejs'
        BACK_CONTAINER_NAME = 'nodejs-server'

        FRONT_DOCKER_IMAGE_NAME='frontend/vuejs'
        FRONT_CONTAINER_NAME='vuejs-client'
    }


    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage('Copy env file'){
            steps{
                script{
                    sh'''
                        cp /var/jenkins_home/settingsFiles/.env ./backend/
                    '''
                }
            }
        }

        stage('Parallel Build Docker Image'){
            parallel{
                stage('frontend Build Docker Image'){
                    steps{
                        script{
                            sh'''
                                cd ./frontend/kind-mulja
                                docker build -t ${FRONT_DOCKER_IMAGE_NAME}
                            '''
                        }
                    }
                }

                stage('backend Build Docker Image'){
                    steps{
                        script{
                            sh'''
                                cd ./backend
                                docker build -t ${BACK_DOCKER_IMAGE_NAME}
                            '''
                        }
                    }
                }
            }
        }

        stage('Parallel Delete Previous Docker Container'){
            parallel{
                stage('Delete Previous Front Docker Container'){
                    steps {
                        script {
                            sh" docker stop  ${FRONT_CONTAINER_NAME} ||tru
                            sh "docker rm ${FRONT_CONTAINER_NAME} || true"
                        }
                    }
                }
                stage('Delete Previous Back Docker Container'){
                    steps {
                        script {
                            sh" docker stop  ${BACK_CONTAINER_NAME} || tru
                            sh "docker rm ${BACK_CONTAINER_NAME} || true"
                        }
                    }
                }
            }
        }


        stage('Parallel Run Docker Container'){
            parallel{
                stage('Run Front Docker Container'){
                    steps{
                        script{
                            sh "docker run -d --name ${FRONT_CONTAINER_NAM
                        }
                    }
                }
                stage('Run Back Docker Container'){
                    steps{
                        script{
                            sh "docker run -d --name ${BACK_CONTAINER_NAME
                        }
                    }
                }
            }
        }

    }
}
```

## 2-2. Docker file

- Vuejs : /frontend/kind-mulja/dockerfile

```
FROM node:20.11.1

WORKDIR /app

COPY package.json .
COPY package-lock.json .

# 의존성 모듈 삭제후
RUN rm -rf node_modules
# npm install 로 의존성 설치 후 구동
RUN npm i

COPY . .

## EXPOSE [Port you mentioned in the vite.config file]

## vite 환경과 동일하게 포트 설정
EXPOSE 5173/tcp

CMD ["npm", "run", "dev"]
```

- Nodejs: /backend/dockerfile

```
FROM node:20

WORKDIR /usr/src/app

COPY package.json package-lock.json ./

RUN npm install
RUN npm install --global pm2

COPY . .

EXPOSE 3000

CMD ["node", "app.js"]
```

# 3. MariaDB HidiSQL Connection

### 3-1. 세션 연결 방법

```
- 네트워크 유형: MariaDB or MySQL(TCP\IP)
- 호스트 명/ IP: j10c109.p.ssafy.io\
- 사용자: root
- 암호: mulja109
- 포트: 3306
```

## 4. Nginx default

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

server {
    #listen 80;
        server_name j10c109.p.ssafy.io;
                #url은 자신의 aws 주소를 입력한다

        access_log /var/log/nginx/reverse-access.log;
        error_log /var/log/nginx/reverse-error.log;

        location / {
                    proxy_pass http://127.0.0.1:5173;
        #포트번호는 서버를 개방하기 위해 설정한 포트번호를 입력한다
    }
        location /api/ {
                    proxy_pass http://127.0.0.1:3000/;
        }

        location /socket {
        #       proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $host;

                proxy_pass http://127.0.0.1:12002;
        # 포트번호를 소켓을 배포한 서버의 포트로 작성합니다.

                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection $connection_upgrade;
        }


        location /lift/ {
        #       proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
                proxy_set_header Host $host;

                proxy_pass http://127.0.0.1:12001/socket.io/;
                # 포트번호를 소켓을 배포한 서버의 포트로 작성합니다.

                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection $connection_upgrade;
        }



    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j10c109.p.ssafy.io/fullchain.pem; #
    ssl_certificate_key /etc/letsencrypt/live/j10c109.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = j10c109.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80;
        listen [::]:80;
        server_name j10c109.p.ssafy.io;
    return 404; # managed by Certbot


}
```

## 5. EC2 Setting

### 5-1. Jenkins

```
# jenkins docker 설치 및 실행

sudo docker run -itd \
-p 8080:8080 \
-p 50000:50000 \
-v /home/ubuntu/jenkins-data:/var/jenkins_home \
-v /$(which docker):/usr/bin/docker \
-v /var/run/docker.sock:/var/run/docker.sock \
--name jenkins jenkins/jenkins:2.444
```

### 5-2. Nginx

- reverse-proxy.conf 설정
- conf 파일 연결(sites-availabe, sites-enabled)
- ssl 설정 - Certbot

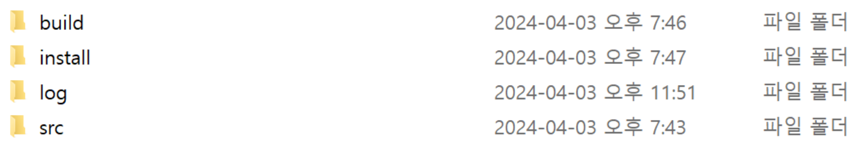
## 6. Files ignored

### 6-1 .gitingore

- /backend/.end

```
DB_HOST=호스트 명
DB_PORT=3306
DB_USER=root
DB_PASSWORD=비밀번호
DB_DATABASE=s10p22c109
```

- ros build 후 생성되는 폴더
  - /ros2_kind_mulja/build
  - /ros2_kind_mulja/intstall
  - /ros2_kind_mulja/log



| | | |
|---|---|---|
| build | 2024-04-03 오후 7:46 | 파일 폴더 |
| install | 2024-04-03 오후 7:47 | 파일 폴더 |
| log | 2024-04-03 오후 11:51 | 파일 폴더 |
| src | 2024-04-03 오후 7:43 | 파일 폴더 |

## 7. Etc