📖

포팅 메뉴얼

형상 관리

- Gitlab

이슈 관리

- Jira

커뮤니케이션

- Mattermost
- Notion
- Discord

IDE

- Intellij `2023.3.2`
- Visual Studio Code `1.85.1`

Server

- AWS Lightsail
  - Ubuntu `20.04.6 LTS`
  - CPU : `쿼드코어`
  - RAM : `16GB`
  - SSD : `320GB`
  - Traffic : `6TB/month`
- Docker `25.0.0`
- Nginx `1.18.0 (Ubuntu)`

Frontend

- Vue `3.3.11`
- NodeJS `20.11.0`
- @stomp/stompjs `7.0.0`

Backend

- Java OpenJDK `17`
- Spring Boot `3.2.1`
- Gradle `8.5`
- Spring Data JPA
- Lombok
- Hibernate

Database

- MariaDB `10.3.39`

Infra

- Jenkins `2.426.2`

SSH 연결(MobaXterm)

```
- MobaXterm을 실행한 뒤 User Sessions 우클릭 -> New session
- SSH 접속정보 입력
1. SSH 타입 선택
2. 서버 주소 입력
3. Specify username 체크
4. 계정명 입력 (ubuntu)
5. Use private key 체크
6. 디렉토리에서 *.pem 키 선택
7. OK 클릭
8. Accept 클릭
```

패키지 업데이트

```
sudo apt-get -y update && sudo apt-get -y upgrade
```

우분투 서버 시간 변경

```
sudo timedatectl set-timezone Asia/Seoul
```

Ufw 설정

```
sudo ufw enable

sudo ufw allow 22
sudo ufw allow 80
sudo ufw allow 443
sudo ufw allow 3000
sudo ufw allow 3306
sudo ufw allow 6379
sudo ufw allow 8081
sudo ufw allow 8082
```

Docker 설치

- Docker 설치 전 필요한 패키지 설치

```
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

- Docker에 대한 GPC Key 인증 진행

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- Docker 레포지토리 등록

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

- Docker 패키지 설치

```
sudo apt-get -y install docker-ce docker-ce-cli containerd.io
```

- Docker 일반 유저에게 권한 부여

```
sudo usermod -aG docker ubuntu
```

- Docker 서비스 재시작

```
sudo service docker restart
```

Nginx

- 설치

```
sudo apt-get -y install nginx
```

- SSL 설정 (CertBot)

```
sudo snap install --classic certbot
```

- SSL 인증서 발급

```
sudo apt-get -y install python3-certbot-nginx
sudo certbot --nginx -d i10a808.p.ssafy.io
```

- Nginx 설정

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;

        server_name _;

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

server {

        client_max_body_size 20M;

        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;
    server_name i10a808.p.ssafy.io; # managed by Certbot


        location / {
                proxy_pass http://i10a808.p.ssafy.io:3000;
        }

        location /api {
                proxy_pass http://i10a808.p.ssafy.io:8082;
        }

        location /ws {
                proxy_pass http://i10a808.p.ssafy.io:8082;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;

                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i10a808.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i10a808.p.ssafy.io/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = i10a808.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name i10a808.p.ssafy.io;
    return 404; # managed by Certbot
```

```
}
```

MariaDB

- 설치

```
sudo apt-get -y install mariadb-server
```

- MariaDB 외부접속 설정

```
sudo sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/mariadb.conf.d/50-server.cnf
```

- MariaDB root 비밀번호 변경

```
sudo mysql -u root -e "set password for 'root'@'localhost' = password('비밀번호'); flush privileges;"
```

- MariaDB root 계정 외부 접속 설정

```
sudo mysql -uroot -p비밀번호 -e "grant all privileges on *.* to 'root'@'%' identified by '비밀번호'; flush privileges;"
```

- MariaDB 서버 시간을 한국 표준시(UTC+9)로 변경

```
sudo sed -i '$s/$/\n\n[mysqld]\ndefault-time-zone='+9:00'/g' /etc/mysql/mariadb.conf.d/50-server.cnf
```

- MariaDB 서비스 재시작

```
sudo systemctl restart mariadb
```

Frontend Dockerfile

```
# nginx 이미지 사용
FROM nginx:latest

# root에 /app 폴더 생성
RUN mkdir /app

# work dir 고정
WORKDIR /app

# work dir에 dist 폴더 생성
RUN mkdir ./dist

# host pc의 현재 경로의 dist 폴더를 work dir의 dist 폴더로 복사
ADD ./dist ./dist

# nginx의 default.conf 삭제
RUN rm /etc/nginx/conf.d/default.conf

# host pc의 nginx.conf를 아래 경로에 복사
COPY ./nginx.conf /etc/nginx/conf.d

# 80 포트 개방
EXPOSE 80

# container 실행 시 자동으로 실행할 command. nginx 시작함
CMD ["nginx", "-g", "daemon off;"]
```

Nginx.conf

```
server {
    listen 80;
    location / {
        root    /app/dist;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

Backend Dockerfile

```
FROM openjdk:17-jdk
WORKDIR /app
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Jenkins

- 설치

```
docker pull jenkins/jenkins:lts
```

- 실행

```
docker run -d --env JENKINS_OPTS=--httpPort=8080 -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul -p 8080:8080
 -v /jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v /usr/local/bin/docker-compose:/usr/local/bin/docker-compose
 --name jenkins -u root jenkins/jenkins:lts
```

- 컨테이너 접속

```
docker exec -it jenkins /bin/bash
```

- Docker Repository 등록 및 docker-ce 패키지 설치

```
apt-get update && apt-get -y install apt-transport-https ca-certificates curl gnupg2 software-properties-common
&& curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey
```

```
&& add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") $(lsb_release -cs) stable"
&& apt-get update && apt-get -y install docker-ce
```

- Docker Jenkins에서 Host Docker 접근권한 부여

```
groupadd -f docker
usermod -aG docker jenkins
chown root:docker /var/run/docker.sock
```

- Docker Compose 다운로드

```
curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

- /usr/local/bin/docker-compose 권한 변경

```
chmod +x /usr/local/bin/docker-compose
```

- 플러그인 다운로드
  - Docker plugin, Docker Pipeline
  - Generic Webhook Trigger
  - GitHub Authentication plugin
  - Gitlab Plugin
  - Gradle
  - NodeJS Plugin
  - SSH Agent Plugin
- Frontend
  - GitLab Connection 선택
  - Build Triggers
    - Build when a change is pushed to GitLab 체크
    - 고급
      - Enable [ci-skip]
      - Ignore WIP Merge Requests
      - Set build description to build cause (eg. Merge request or Git Push)
      - Filter branches by name(include - fe)
      - Secret token
  - Pipeline script

```
pipeline {
    agent any
    tools {nodejs "nodejs"}

    environment {
        imageName = "wooseobee/a808-frontend"
        registryCredential = 'docker'
        dockerImage = ''

        releaseServerAccount = 'ubuntu'
        releaseServerUri = 'i10a808.p.ssafy.io'
        releasePort = '80'
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'fe',
                    credentialsId: 'gitlab',
                    url: 'https://lab.ssafy.com/s10-webmobile2-sub2/S10P12A808.git'
            }
        }
        stage('.env download') {
            steps {
                withCredentials([file(credentialsId: 'env', variable: 'env')]) {
                    script {
                        sh 'cp $env frontend/.env'
                    }
                }
            }
        }
        stage('Node Build') {
            steps {
                dir ('frontend') {
                    sh 'npm install'
                    sh 'npm run build'
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                dir('frontend') {
                    script {
                        docker.withRegistry('', registryCredential) {
                            sh "docker buildx create --use --name mybuilder"
                            sh "docker buildx build --platform linux/amd64,linux/arm64 -t $imageName:$BUILD_NUMBER --push ."
                            sh "docker buildx build --platform linux/amd64,linux/arm64 -t $imageName:latest --push ."
                        }
                    }
                }
            }
        }
        stage('Before Service Stop') {
            steps {
                sshagent(credentials: ['ubuntu-a808']) {
                    sh '''
                    if test "`ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker ps -aq --filter ancestor=$imageName:latest"`"; then
```

```
                ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker stop $(docker ps -aq --filter ancestor=$imageName:latest)"
                ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker rm -f $(docker ps -aq --filter ancestor=$imageName:latest)"
                ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker rmi $imageName:latest"
                fi
                '''
            }
        }
    }
    stage('DockerHub Pull') {
        steps {
            sshagent(credentials: ['ubuntu-a808']) {
                sh "ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri 'sudo docker pull $imageName:latest'"
            }
        }
    }
    stage('Service Start') {
        steps {
            sshagent(credentials: ['ubuntu-a808']) {
                sh '''
                    ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "sudo docker run -i -e TZ=Asia/Seoul
                                        --name frontend -p 3000:$releasePort -v /etc/letsencrypt:/etc/letsencrypt -d $imageName:latest"
                '''
            }
        }
    }
    stage('Service Check') {
        steps {
            sshagent(credentials: ['ubuntu-a808']) {
                sh '''
                    #!/bin/bash

                    for retry_count in \$(seq 20)
                    do
                      if curl -s "https://i10a808.p.ssafy.io" > /dev/null
                      then
                          curl -d '{"text":"# [BOT] :jerry: Front 자동 CI/CD 배포 성공"}' -H "Content-Type: application/json"
                                                          -X POST https://meeting.ssafy.com/hooks/hz6ikiege7n7mq56iby8trbafc
                          break
                      fi

                      if [ $retry_count -eq 20 ]
                      then
                        curl -d '{"text":"# [BOT] :soragae2: Front 자동 CI/CD 배포 실패"}' -H "Content-Type: application/json"
                                                      -X POST https://meeting.ssafy.com/hooks/hz6ikiege7n7mq56iby8trbafc
                        exit 1
                      fi

                      echo "The server is not alive yet. Retry health check in 5 seconds..."
                      sleep 5
                    done
                '''
            }
        }
    }
}
```

- Backend Pipeline
  - GitLab Connection 선택
  - Build Triggers
    - Build when a change is pushed to GitLab 체크
    - 고급
      - Enable [ci-skip]
      - Ignore WIP Merge Requests
      - Set build description to build cause (eg. Merge request or Git Push)
      - Filter branches by name(include - be)
      - Secret token
  - Pipeline script

```
pipeline {
    agent any

    environment {
        imageName = "wooseobee/a808-backend"
        registryCredential = 'docker'
        dockerImage = ''

        releaseServerAccount = 'ubuntu'
        releaseServerUri = 'i10a808.p.ssafy.io'
        releasePort = '8082'
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'be',
                    credentialsId: 'gitlab',
                    url: 'https://lab.ssafy.com/s10-webmobile2-sub2/S10P12A808.git'
            }
        }
        stage('yml download') {
            steps {
                sh 'mkdir -p backend/src/main/resources'

                withCredentials([file(credentialsId: 'application.yml', variable: 'yamlFile')]) {
                    script {
                        sh 'cp $yamlFile backend/src/main/resources/application.yml'
                    }
```

```
                }
            }
        }
        stage('Jar Build') {
            steps {
                dir ('backend') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean bootJar'
                    // sh './gradlew build'
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                dir('backend') {
                    script {
                        docker.withRegistry('', registryCredential) {
                            sh "docker buildx create --use --name mybuilder"
                            sh "docker buildx build --platform linux/amd64,linux/arm64 -t $imageName:$BUILD_NUMBER --push ."
                            sh "docker buildx build --platform linux/amd64,linux/arm64 -t $imageName:latest --push ."
                        }
                    }
                }
            }
        }
        stage('Before Service Stop') {
            steps {
                sshagent(credentials: ['ubuntu-a808']) {
                    sh '''
                    if test "`ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker ps -aq --filter ancestor=$imageName:latest"`"; then
                    ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker stop $(docker ps -aq --filter ancestor=$imageName:latest)"
                    ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker rm -f $(docker ps -aq --filter ancestor=$imageName:latest)"
                    ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "docker rmi $imageName:latest"
                    fi
                    '''
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: ['ubuntu-a808']) {
                    sh "ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri 'sudo docker pull $imageName:latest'"
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: ['ubuntu-a808']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri "sudo docker run -i -e TZ=Asia/Seoul
                                                -e "SPRING_PROFILES_ACTIVE=prod" --name backend -p $releasePort:$releasePort -d $imageName:latest"
                    '''
                }
            }
        }
        stage('Service Check') {
            steps {
                sshagent(credentials: ['ubuntu-a808']) {
                    sh '''
                        #!/bin/bash

                        for retry_count in \$(seq 20)
                        do
                          if curl -s "https://i10a808.p.ssafy.io/actuator/health" > /dev/null
                          then
                              curl -d '{"text":"# [BOT] :jerry: Server 자동 CI/CD 배포 성공"}' -H "Content-Type: application/json" -X POST {알림 url}
                              break
                          fi

                          if [ $retry_count -eq 20 ]
                          then
                            curl -d '{"text":"# [BOT] :soragae2: Server 자동 CI/CD 배포 실패"}' -H "Content-Type: application/json" -X POST {알림 url}
                            exit 1
                          fi

                          echo "The server is not alive yet. Retry health check in 5 seconds..."
                          sleep 5
                        done
                    '''
                }
            }
        }
    }
}
```

- Jenkins 관리
  - System
    - Global properties
      - Environment variables
        - 이름 : CI
        - 값 : false
    - GitLab
      - Check Enable authentication for '/project' end-point

        ```
        Connection name : a808
        GitLab host URL : https://lab.ssafy.com
        Credentials : GitLab API token
        ```
  - Tools

- Gradle installations
  - Add Gradle

    ```
    {Gradle version} 8.5
    ```
- NodeJS installations
  - Add NodeJS

    ```
    {NodeJS version} 20.11.0
    ```

- Credentials
  - GitLab - Username with password
  - GitLab API Token
  - Docker Hub Token - Username with password
  - Ubuntu Credential - SSH Username with private key
  - env - Secret file
  - application.yml - Secret file