🚀

포팅 메뉴얼

💊SSAFY 8기 자율프로젝트_B302 어바웃 타임캡슐 포팅 매뉴얼💊

✏️목차✏️

개발 환경

🐸IDE

- Android Studio: `Flamingo 2022.2.1`
- Intellij: `2022.3.1`

🐹프론트엔드

- Kotlin: `1.8.0`
- JAVA: `11.0.17`

🐷백엔드

- JAVA: `11.0.18`
- MySQL: `8.0.32`
- SpringBoot: `2.7.11`

🐳인프라

- Docker: `23.0.5`
- Jenkins: `2.60.3`

프로젝트 사용 도구

- 이슈 관리: JIRA
- 형상 관리: GitLab
- 커뮤니케이션: Notion, Mattermost, Webex
- 디자인: Figma
- UCC: Movavi
- CI/CD: Jenkins

외부 서비스

- AWS S3

```
cloud:
  aws:
    credentials:
      access-key: ${S3_ACCESS_KEY}
      secret-key: ${S3_SECRET_KEY}
    region:
      static: ap-northeast-2
    s3:
      bucket: ${S3_BUCKET}
    stack:
      auto: false
```

- Kakao Local API

```
kakao:
  localMap:
    key: ${KAKAO_LOCAL_KEY}
```

- Redis

```
redis:
  host: ${HOST}
  port: ${PORT}
  password: ${PASSWORD}
```

- FCM

```
ServiceAccountKey.json 에 해당 내용 있음
```

- MySQL

```
datasource:
  driver-cass-name: com.mysql.cj.jdbc.Driver
  url: ${URL}
  username: ${USERNAME}
  password: ${PASSWORD}
```

- Security OAuth

```
spring:
  security:
    oauth2:
      client:
        registration:
          naver:
            client-id: ${NAVER_CLIENT_ID}
            client-secret: ${NAVER_SECRET_KEY}
            authorization-grant-type: authorization_code
            redirect-uri: "${BASE_URL}/api/oauth/naver/callback"
            scope:
              - email
              - profile_image
            client-name: naver
          kakao:
            client-id: ${KAKAO_CLIENT_ID}
            client-secret : ${KAKAO_SECRET_KEY}
            redirect-uri: "${BASE_URL}/api/oauth/kakao/callback"
            client-authentication-method: POST
```

```
            authorization-grant-type: authorization_code
            scope:
              - profile_image
              - account_email
            client-name: kakao
        provider:
          naver:
            authorization-uri: https://nid.naver.com/oauth2.0/authorize
            token-uri: https://nid.naver.com/oauth2.0/token
            user-info-uri: https://openapi.naver.com/v1/nid/me
            user-name-attribute:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id
```

- JWT

```
jwt:
  secret-key: ${SECRET_KEY}
  access-expiration-time: 43200000
  refresh-expiration-time: 1209600000
```

빌드 및 배포

🚪 AWS EC2 접속

```
ssh -i K8B302T.pem ubuntu@k8b302.p.ssafy.io
```

🐳 AWS에 Docker 설치

- apt 패키지 업데이트 및 설치

```
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg
```

- Docker GPG key 추가

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

- Repository 셋업

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- Docker Engine 설치

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

🐬 MySQL 컨테이너 생성

- MySQL Docker 이미지 다운로드

```
sudo docker pull mysql:8.0.32
```

- MySQL Docker 컨테이너 생성 및 실행

```
sudo docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=b302capsule -d -p 3306:3306 mysql:8.0.32
```

- MySQL Docker 컨테이너 접속

```
sudo docker exec -it mysql-container bash
mysql -u root -p
Enter password: #비밀번호 입력
```

🎲 Redis 컨테이너 생성

- Redis Docker 이미지 다운로드

```
docker pull redis
```

- Redis Docker 컨테이너 생성 및 실행

```
docker run -p 6379:6379 --name redis -d redis:latest --requirepass "비밀번호"
```

- Redis Docker 컨테이너 접속

```
docker exec -it redis redis-cli
docker exec -it redis redis-cli -a #비밀번호
```

🤵 Jenkins 설치 및 설정

Jenkins 컨테이너 설치

- 젠킨스 설치시 아래 옵션을 추가해 도커의 소켓 파일 마운트

```
sudo mkdir /home/opendocs/jenkins
sudo docker run \
--name jenkins \
-d \
-p 8080:8080 \
-p 50000:50000 \
-v /home/opendocs/jenkins:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-u root \
jenkins/jenkins:lts
```

- Jenkins 컨테이너 접속

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

- Jenkins 안 Docker 설치

```
# Docker 설치
## - Old Version Remove
apt-get remove docker docker-engine docker.io containerd runc
## - Setup Repo
apt-get update
apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
## - Install Docker Engine
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

🐳 Dockerfile

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
ARG JAR_FILE=build/libs/*jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

🤵 Jenkinsfile

- 각 마이크로 서비스 별로 Jenkinsfile 생성

```
pipeline {
    agent any
    environment {
        DOCKER = 'sudo docker'
    }
```

```
    stages {
        stage('Clone Repository') {
            steps {
                checkout scm
                echo 'Checkout Scm'
            }
        }

        stage('Build image') {
            steps {
                sh 'ls -al'
                dir('BE/capsule-service') {
                    sh 'ls -al'
                    sh 'chmod +x ./gradlew'
                    sh './gradlew build'
                    sh 'docker build -t rink645/timecapsule-capsule .'
                }
                echo 'Build timecapsule-capsule image...'
            }
        }

        stage('Remove Previous image') {
            steps {
                script {
                    try {
                        h 'docker stop timecapsule-capsule'
                        sh 'docker rm timecapsule-capsule'
                    } catch (e) {
                        echo 'fail to stop and remove container'
                    }
                }
            }
        }
        stage('Run New image') {
            steps {
                sh 'docker run --name timecapsule-capsule -d -p 9003:9003 rink645/timecapsule-capsule'
                echo 'Run New member image'
            }
        }
    }
}
```