포팅 메뉴얼

1. 사용 도구

- **이슈 관리**: Notion, Jira
- **형상 관리**: GitLab
- **커뮤니케이션**: MatterMost, Discord
- **디자인**: Figma
- **CI/CD**: Jenkins, Docker

2. 개발 도구

2.1. Frontend

- **프레임워크**: React (18.3.1)
- **라이브러리**:
    - Zustand
    - React Router
    - MUI
    - howler
    - lottie
    - react-dnd
    - react-icons
    - react-modal
    - react-string-replace
    - react-type-animation
    - stompjs
    - styled-components
    - sweetalert2
    - terser

2.2. Backend

- **개발 언어:** Java (21)
- **프레임워크**: Spring Boot (3.3.1)
    - **Dependencies:**
        - Spring Boot Dev Tools
        - Spring Data JPA
        - Lombok
        - MariaDB Driver
        - Spring Web
        - Spring Data Redis
        - Spring for Apache Kafka
        - Spring Security
        - WebSocket
        - Spring Reactive Web

2.3. AI

- 프레임워크 : Django
- 라이브러리 : scikit-learn(RandomForestClassifier), matplotlib, pandas
- 언어 : Python
- 세팅
    - Django requirement.txt

```
# requirements.txt

asgiref==3.8.1
certifi==2024.7.4
chardet==5.2.0
charset-normalizer==3.3.2
contourpy==1.2.1
cycler==0.12.1
Django==4.2.14
```

```
django-cors-headers==4.4.0
django-rest-framework==0.1.0
djangorestframework==3.15.2
fonttools==4.53.1
fpdf==1.7.2
idna==3.7
joblib==1.4.2
kiwisolver==1.4.5
logging==0.4.9.6
matplotlib==3.9.1
numpy==2.0.1
packaging==24.1
pandas==2.2.2
pillow==10.4.0
pyparsing==3.1.2
python-dateutil==2.9.0.post0
pytz==2024.1
reportlab==4.2.2
requests==2.32.3
scikit-learn==1.5.1
scipy==1.14.0
six==1.16.0
sqlparse==0.5.1
threadpoolctl==3.5.0
tzdata==2024.1
urllib3==2.2.2
```

- Django 가상 환경 설정

  1. 가상환경 생성 및 실행

  ```
  python -m venv venv

  source venv/bin/activate    # Windows의 경우 venv/source/act
  ```

  2. 필요한 패키지 설치

  ```
  pip install -r requirements.txt
  ```

3. properties 환경 및 Docker-Compose 파일

1. Docker-Compose

```
version: '3'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    networks:
      - network

  kafka:
    image: confluentinc/cp-kafka:latest
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    depends_on:
      - zookeeper
    networks:
      - network

  mariadb-user:
    image: mariadb:latest
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: member_db
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    networks:
      - network
    depends_on:
      - kafka
    volumes:
      - db_data_member:/var/lib/mysql

  mariadb-problem:
    image: mariadb:latest
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: problem_db
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    networks:
      - network
    volumes:
      - db_data_problem:/var/lib/mysql

  mariadb-edu:
    image: mariadb:latest
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: edu_db
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    networks:
      - network
    depends_on:
      - kafka
    volumes:
      - db_data_edu:/var/lib/mysql

  redis-user:
    image: redis:latest
    volumes:
      - ./config/redis.conf:/usr/local/etc/redis/redis.conf
      - redis_data:/data  # Redis 데이터를 저장할 볼륨
    command: [ "redis-server", "/usr/local/etc/redis/redis.co
    networks:
      - network

  user-server:
    image: zlxldgus123/user
    depends_on:
      - mariadb-user
      - redis-user
    environment:
      SPRING_DATASOURCE_URL: jdbc:mariadb://mariadb-user:3306
      SPRING_DATASOURCE_USERNAME: user
      SPRING_DATASOURCE_PASSWORD: password
      SPRING_DATA_REDIS_HOST: redis-user
      SPRING_DATA_REDIS_PORT: 6379
      JWT_SECRET: 7d1b1d6d36d8e6a8f1bda6a7f473f87b012b0345a1b
      JWT_ACCESS_TOKEN_EXPIRY: 3600000
      JWT_REFRESH_TOKEN_EXPIRY: 86400000
      LOGGING_LEVEL_ORG_HIBERNATE_SQL: debug
      LOGGING_LEVEL_COM_SCF_USER_GLOBAL_LOGINFILTER: debug
      SERVER_ERROR_INCLUDE_MESSAGE: always
      SERVER_ERROR_INCLUDE_BINDING_ERRORS: always
    networks:
      - network

  problem-server:
    image: zlxldgus123/problem
    depends_on:
      - mariadb-problem
      - redis-user
      - user-server
    environment:
      SPRING_DATASOURCE_URL: jdbc:mariadb://mariadb-problem:3
      SPRING_DATASOURCE_USERNAME: user
      SPRING_DATASOURCE_PASSWORD: password
      SPRING_DATA_REDIS_HOST: redis-user
      SPRING_DATA_REDIS_PORT: 6379
      JWT_SECRET: 0yZ6aRLZ2zXFR83xzIAtC25OQRXqsUEHtaTZYLLUsQU
      JWT_ACCESS_TOKEN_EXPIRY: 3600000
      JWT_REFRESH_TOKEN_EXPIRY: 86400000
      LOGGING_LEVEL_ORG_HIBERNATE_SQL: debug
      LOGGING_LEVEL_COM_SCF_USER_GLOBAL_LOGINFILTER: debug
      SERVER_ERROR_INCLUDE_MESSAGE: always
      SERVER_ERROR_INCLUDE_BINDING_ERRORS: always
    networks:
      - network

  multi-server:
    image: zlxldgus123/multi
    depends_on:
      - kafka
      - problem-server
    environment:
      SPRING_KAFKA_BOOTSTRAP_SERVERS: kafka:9092
      SPRING_KAFKA_PRODUCER_KEY_SERIALIZER: org.apache.kafka.
      SPRING_KAFKA_PRODUCER_VALUE_SERIALIZER: org.springframe
      SPRING_KAFKA_PRODUCER_PROPERTIES_ACKS: all
      SPRING_KAFKA_PRODUCER_PROPERTIES_RETRIES: 3
      SPRING_KAFKA_PRODUCER_PROPERTIES_COMPRESSION_TYPE: gzip
      SPRING_KAFKA_PRODUCER_PROPERTIES_BATCH_SIZE: 16384
      SPRING_KAFKA_PRODUCER_PROPERTIES_LINGER_MS: 1
      LOGGING_LEVEL_ORG_HIBERNATE_SQL: debug
      LOGGING_LEVEL_COM_SCF_USER_GLOBAL_LOGINFILTER: debug
      SERVER_ERROR_INCLUDE_MESSAGE: always
      SERVER_ERROR_INCLUDE_BINDING_ERRORS: always
      SPRING_APPLICATION_NAME: scf-multi
      PROBLEM_SERVER_URL: http://www.ssafy11s.com/problem
      USER_SERVER_URL: http://www.ssafy11s.com/user
    networks:
      - network

  rank-server:
    image: zlxldgus123/rank
    depends_on:
      - redis-user
      - kafka
      - multi-server
    environment:
      SPRING_KAFKA_BOOTSTRAP_SERVERS: kafka:9092
      SPRING_KAFKA_CONSUMER_GROUP_ID: ranking-group
      SPRING_KAFKA_CONSUMER_AUTO_OFFSET_RESET: earliest
      SPRING_KAFKA_CONSUMER_ENABLE_AUTO_COMMIT: 'false'
      SPRING_KAFKA_PRODUCER_KEY_SERIALIZER: org.apache.kafka.
      SPRING_KAFKA_PRODUCER_VALUE_SERIALIZER: org.apache.kafk
      SPRING_KAFKA_LISTENER_ACK_MODE: manual
      SPRING_APPLICATION_NAME: scf-rank
      SPRING_DATA_REDIS_PORT: 6379
      SPRING_DATA_REDIS_HOST: redis-user
      LOGGING_LEVEL_ORG_HIBERNATE_SQL: debug
      SERVER_ERROR_INCLUDE_MESSAGE: always
      SERVER_ERROR_INCLUDE_BINDING_ERRORS: always
    networks:
      - network

  battle-server:
    image: zlxldgus123/battle
    depends_on:
      - kafka
      - rank-server
    environment:
      SPRING_KAFKA_BOOTSTRAP_SERVERS: kafka:9092
      SPRING_KAFKA_PRODUCER_KEY_SERIALIZER: org.apache.kafka.
      SPRING_KAFKA_PRODUCER_VALUE_SERIALIZER: org.springframe
      SPRING_KAFKA_PRODUCER_PROPERTIES_ACKS: all
      SPRING_KAFKA_PRODUCER_PROPERTIES_RETRIES: 3
      SPRING_KAFKA_PRODUCER_PROPERTIES_COMPRESSION_TYPE: gzip
      SPRING_KAFKA_PRODUCER_PROPERTIES_BATCH_SIZE: 16384
      SPRING_KAFKA_PRODUCER_PROPERTIES_LINGER_MS: 1
      LOGGING_LEVEL_ORG_HIBERNATE_SQL: debug
      SERVER_ERROR_INCLUDE_MESSAGE: always
      SERVER_ERROR_INCLUDE_BINDING_ERRORS: always
      SPRING_APPLICATION_NAME: scf-battle
      PROBLEM_SERVER_URL: http://www.ssafy11s.com/problem
      USER_SERVER_URL: http://www.ssafy11s.com/user/public/ch
    networks:
      - network

  chat-server:
    image: zlxldgus123/chat
    depends_on:
      - rank-server
      - battle-server
    environment:
      LOGGING_LEVEL_ORG_HIBERNATE_SQL: debug
      SPRING_APPLICATION_NAME: scf-chat
      SPRING_DATASOURCE_URL: jdbc:mariadb://mariadb-user:3306
      SPRING_DATASOURCE_USERNAME: user
      SPRING_DATASOURCE_PASSWORD: password
    networks:
      - network

  single-server:
    image: zlxldgus123/single
    depends_on:
      - mariadb-edu
      - chat-server
    environment:
      SPRING_DATASOURCE_URL: jdbc:mariadb://mariadb-edu:3306/
      SPRING_DATASOURCE_USERNAME: user
      SPRING_DATASOURCE_PASSWORD: password
      LOGGING_LEVEL_ORG_HIBERNATE_SQL: debug
      LOGGING_LEVEL_COM_SCF_USER_GLOBAL_LOGINFILTER: debug
      SERVER_ERROR_INCLUDE_MESSAGE: always
      SERVER_ERROR_INCLUDE_BINDING_ERRORS: always
    networks:
      - network

networks:
  network:
    external: true
    driver: bridge

volumes:
  db_data_member:
  db_data_problem:
  db_data_edu:
  redis_data:
```

2. 배틀 서버 properties

```
spring:
  application:
    name: scf-battle

  kafka:
    bootstrap-servers: localhost:9092
    producer:
      key-serializer: org.apache.kafka.common.serialization.S
      value-serializer: org.springframework.kafka.support.ser
      properties:
        acks: all
        retries: 3
        compression-type: gzip
        batch-size: 16384
        linger.ms: 1

  web:
    encoding:
      charset: UTF-8
      enabled: true
      force: true

logging:
  level:
    org.hibernate.SQL: debug

problem:
  server:
    url: http://www.ssafy11s.com/problem

user:
  server:
    url: http://www.ssafy11s.com/user/public/charaterType

server:
  port: 8084
```

3. 채팅 서버 properties

```
spring:
  application:
    name: scf-chat

  kafka:
    bootstrap-servers: localhost:9092
    producer:
      key-serializer: org.apache.kafka.common.serialization.S
      value-serializer: org.springframework.kafka.support.ser
      properties:
        acks: all
        retries: 3
        compression-type: gzip
        batch-size: 16384
        linger.ms: 1

  web:
    encoding:
      charset: UTF-8
      enabled: true
      force: true

logging:
  level:
    org.hibernate.SQL: debug

problem:
  server:
    url: http://www.ssafy11s.com/problem

user:
  server:
    url: http://www.ssafy11s.com/user/public/charaterType

server:
  port: 8084
```

3. 멀티 서버 properties

```
spring:
  application:
    name: scf-multi
  web:
    encoding:
      charset: UTF-8
      enabled: true
      force: true

logging:
  level:
    com.scf.multi: DEBUG

problem:
  server:
    url: http://www.ssafy11s.com/problem

user:
  server:
    url: http://www.ssafy11s.com/user

server:
  port: 8082
```

4. 문제 서버 properties

```
spring:
  application:
    name: scf-problem
  datasource:
    url: ${SPRING_DATASOURCE_URL}
    username: ${SPRING_DATASOURCE_USERNAME}
    password: ${SPRING_DATASOURCE_PASSWORD}
    driver-class-name: org.mariadb.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MariaDBDialect

logging:
  level:
    org:
      hibernate:
        SQL: debug

server:
  port: 8081
```

5. 랭킹 서버 properties

```
spring:

  kafka:
    bootstrap-servers: localhost:9092
    consumer:
      group-id: ranking-group
      auto-offset-reset: earliest
      enable-auto-commit: false
    producer:
      key-serializer: org.apache.kafka.common.serialization.S
      value-serializer: org.apache.kafka.common.serialization
    listener:
      ack-mode: manual

  application:
    name: scf-rank

  data:
   redis:
    host: redis-user
    port: 6379

logging:
  level:
    org:
      hibernate:
        SQL: debug

server:
  port: 8083
```

6. 싱글 서버 properties

```
spring:
  application:
    name: scf-single

  datasource:
    url: ${SPRING_DATASOURCE_URL}
    username: ${SPRING_DATASOURCE_USERNAME}
    password: ${SPRING_DATASOURCE_PASSWORD}
    driver-class-name: org.mariadb.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true  # Shows SQL statements in the console
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MariaDBDialect

logging:
  level:
    org:
      hibernate:
        SQL: debug
```

```
server:
  port: 8086
```

7. 유저 서버 properties

```
spring:
  application:
    name: scf-user
  datasource:
    url: ${SPRING_DATASOURCE_URL}
    username: ${SPRING_DATASOURCE_USERNAME}
    password: ${SPRING_DATASOURCE_PASSWORD}
    driver-class-name: org.mariadb.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true  # Shows SQL statements in the console
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MariaDBDialect
  data:
    redis:
      host: redis-user
      port: 6379
  jwt:
    secret: ${JWT_SECRET}
    access-token-expiry: ${JWT_ACCESS_TOKEN_EXPIRY}
    refresh-token-expiry: ${JWT_REFRESH_TOKEN_EXPIRY}

logging:
  level:
    org:
      hibernate:
        SQL: debug
```

각 서버 Dockerfile

1. battle 서버 Dockerfile

```
# openjdk 21 버전의 환경을 구성
FROM openjdk:21-jdk-slim

WORKDIR /app

# COPY만 docker-compose 파일의 위치를 기반으로 작동함
COPY . .

# 개행문자 오류 해결 [unix와 window 시스템 차이]
RUN sed -i 's/\r$//' gradlew

# RUN은 현재 파일을 위치를 기반으로 작동함
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test --stacktrace

RUN mv build/libs/battle-0.0.1-SNAPSHOT.jar /app/app.jar

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

2. chat 서버 Dockerfile

```
# openjdk 21 버전의 환경을 구성
FROM openjdk:21-jdk-slim

WORKDIR /app

# COPY만 docker-compose 파일의 위치를 기반으로 작동함
COPY . .

# 개행문자 오류 해결 [unix와 window 시스템 차이]
RUN sed -i 's/\r$//' gradlew

# RUN은 현재 파일을 위치를 기반으로 작동함
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test --stacktrace
RUN mv build/libs/chat-0.0.1-SNAPSHOT.jar /app/app.jar

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

3. multi 서버 Dockerfile

```
# openjdk 21 버전의 환경을 구성
FROM openjdk:21-jdk-slim

WORKDIR /app

# COPY만 docker-compose 파일의 위치를 기반으로 작동함
COPY . .

# 개행문자 오류 해결 [unix와 window 시스템 차이]
RUN sed -i 's/\r$//' gradlew

# RUN은 현재 파일을 위치를 기반으로 작동함
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test --stacktrace

RUN mv build/libs/multi-0.0.1-SNAPSHOT.jar /app/app.jar

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

4. problem 서버 Dockerfile

```
# openjdk 21 버전의 환경을 구성
FROM openjdk:21-jdk-slim

WORKDIR /app

# COPY만 docker-compose 파일의 위치를 기반으로 작동함
COPY . .

# 개행문자 오류 해결 [unix와 window 시스템 차이]
RUN sed -i 's/\r$//' gradlew
```

```
# RUN은 현재 파일을 위치를 기반으로 작동함
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test --stacktrace

RUN mv build/libs/problem-0.0.1-SNAPSHOT.jar /app/app.jar

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

5. rank 서버 Dockerfile

```
# openjdk 21 버전의 환경을 구성
FROM openjdk:21-jdk-slim

WORKDIR /app

# COPY만 docker-compose 파일의 위치를 기반으로 작동함
COPY . .

# 개행문자 오류 해결 [unix와 window 시스템 차이]
RUN sed -i 's/\r$//' gradlew

# RUN은 현재 파일을 위치를 기반으로 작동함
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test --stacktrace

RUN mv build/libs/rank-0.0.1-SNAPSHOT.jar /app/app.jar

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

6. single 서버 Dockerfile

```
# openjdk 21 버전의 환경을 구성
FROM openjdk:21-jdk-slim

WORKDIR /app
```

```
# COPY만 docker-compose 파일의 위치를 기반으로 작동함
COPY . .

# 개행문자 오류 해결 [unix와 window 시스템 차이]
RUN sed -i 's/\r$//' gradlew

# RUN은 현재 파일을 위치를 기반으로 작동함
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test --stacktrace

RUN mv build/libs/single-0.0.1-SNAPSHOT.jar /app/app.jar

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

7. user 서버 Dockerfile

```
# openjdk 21 버전의 환경을 구성
FROM openjdk:21-jdk-slim

WORKDIR /app

# COPY만 docker-compose 파일의 위치를 기반으로 작동함
COPY . .

# 개행문자 오류 해결 [unix와 window 시스템 차이]
RUN sed -i 's/\r$//' gradlew

# RUN은 현재 파일을 위치를 기반으로 작동함
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test --stacktrace

RUN mv build/libs/user-0.0.1-SNAPSHOT.jar /app/app.jar

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

4. 포트 번호

- **Jenkins**: 7777
- **Nginx**: 80
- **Frontend**: 3000
- **Backend**: 8080(user), 8081(problem), 8082(multi), 8083(rank), 8084(battle), 8085(chat), 8086(single)
- **MySQL**: 3306, 3307
- **Kafka**: 9092
- **Zookeeper**: 2181
- **Redis:** 6379

5. 빌드 방법

5.1. Docker 설치 및 설정

1. **패키지 업데이트**

   ```
   sudo apt-get update
   ```

2. **필요한 패키지 설치**

   ```
   sudo apt-get install apt-transport-https ca-certificates
   curl
   ```

3. **GPG 키 추가**

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

4. **Docker 저장소 설정**

   ```
   echo \
   "deb [arch=$(dpkg --print-architecture) signed-by=/usr/s
   hare/keyrings/docker-archive-keyring.gpg] https://downlo
   ad.docker.com/linux/ubuntu \
   ```

```
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.l
ist.d/docker.list > /dev/null
```

5. **Docker 설치**

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

6. **Docker 실행**

```
sudo service docker start
```

7. **Docker Compose 설치**

```
sudo curl -L "https://github.com/docker/compose/release
s/download/1.29.2/docker-compose-$(uname -s)-$(uname -
m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo usermod -aG docker $USER
docker-compose --version
```

5.2. Jenkins 설치 및 설정

1. **볼륨 디렉토리 생성**

```
cd /home/ubuntu && mkdir jenkins-data
```

2. **포트 오픈 및 상태 확인**

```
sudo ufw allow 7777/tcp
sudo ufw reload
sudo ufw status
```

3. **Jenkins 컨테이너 생성 및 구동**

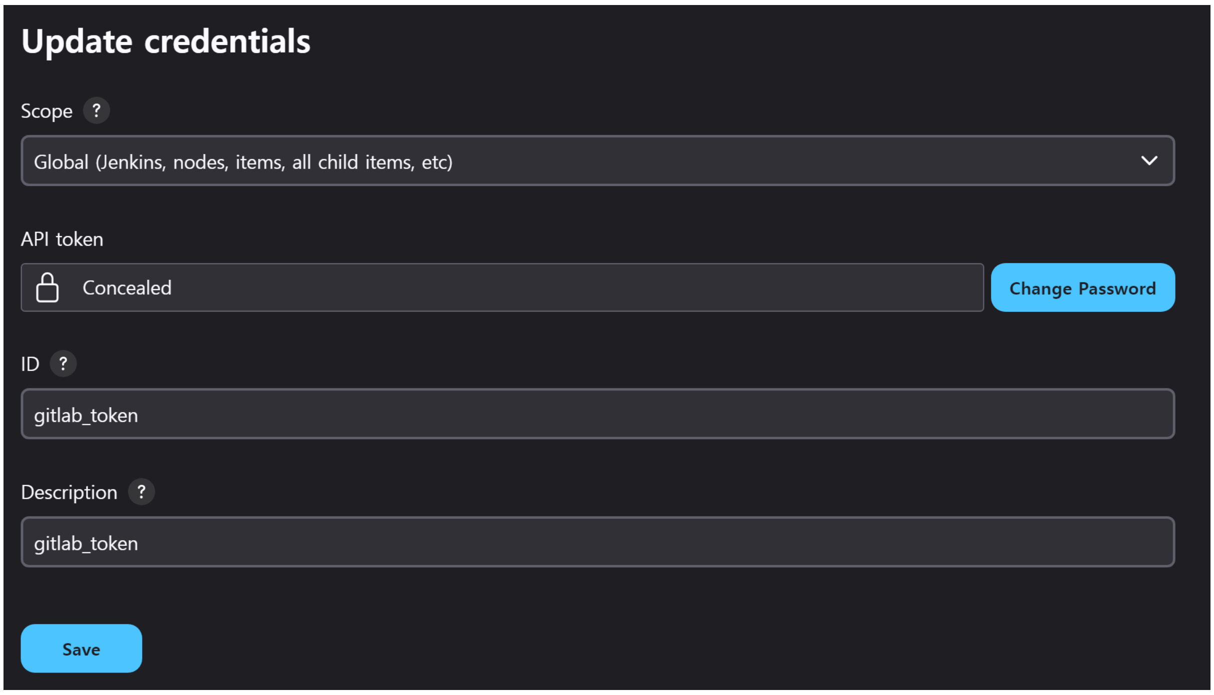
```
sudo docker run -d -p 7777:7777-v /var/run/docker.sock:/
var/run/docker.sock -v jenkins-data:/var/jenkins_home je
```

```
nkins/jenkins:lts
```

4. **로그 확인 및 초기 패스워드 확인**

```
sudo docker logs jenkins
```

5. **Jenkins 접속 및 계정 설정**
   - http://주소:7777에 접속하여 초기 비밀번호를 사용해 계정을 설정합니다.
6. **Jenkins 플러그인 설치**
   - 제안된 플러그인을 모두 설치하고, 필요한 추가 플러그인을 설치합니다.
   - 필수 플러그인:
     - Docker
     - Docker Pipeline
     - Docker API
     - Gitlab
     - Generic Webhook Trigger Plugin
7. **Jenkins Credential 설정**
   - Jenkins 관리에서 `Credential` 설정.
   - 등록해야 할 항목:
     - `gitlab_token`, `gitlab_login`, `DOCKER_REPO`, `dockerhub_credentials`, `docker-compose`
8. **Docker Compose 파일 등록**
   - **종류**: Secret file
   - **파일**: `docker-compose.yml`
   - **ID**: `docker-compose`


Update credentials

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

API token

Concealed

Change Password

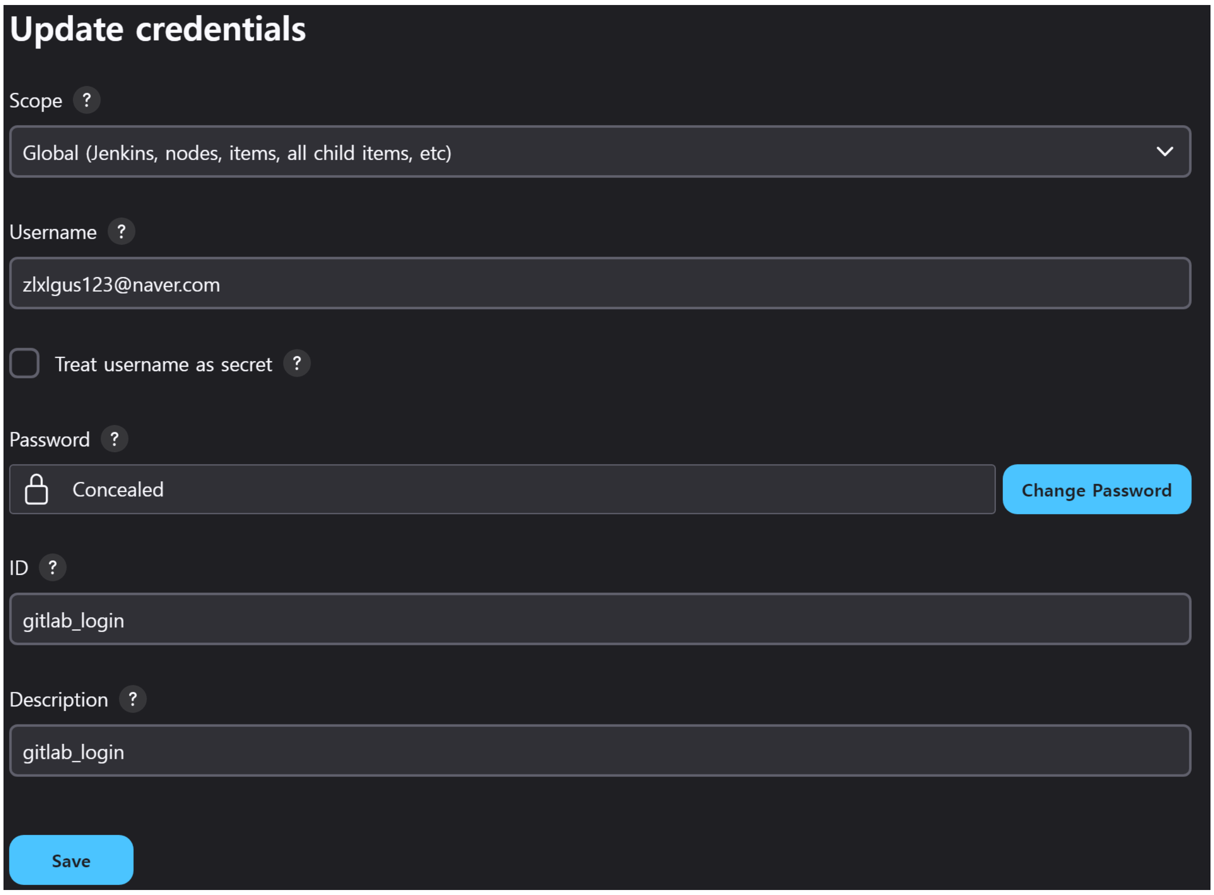
ID ?

gitlab_token

Description ?

gitlab_token

Save


Update credentials

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

zlxlgus123@naver.com

☐ Treat username as secret ?

Password ?

Concealed

Change Password

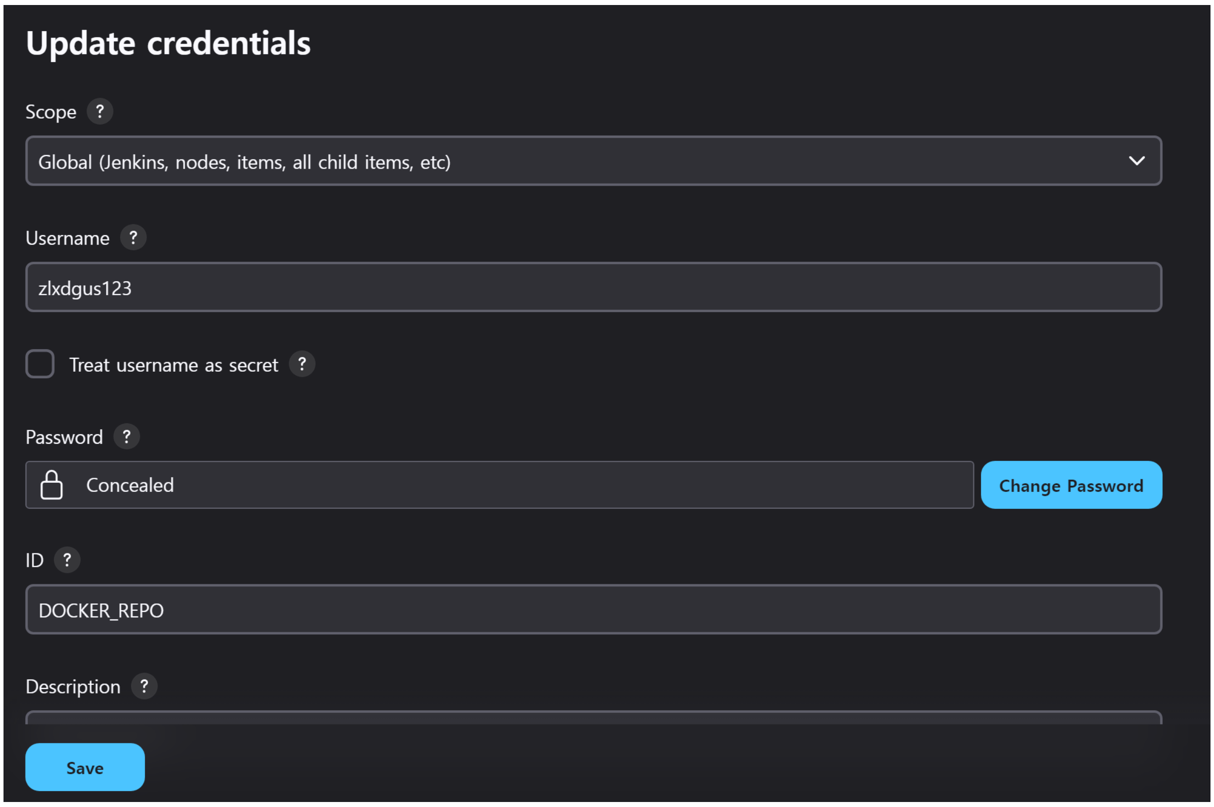
ID ?

gitlab_login

Description ?

gitlab_login

Save


Update credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

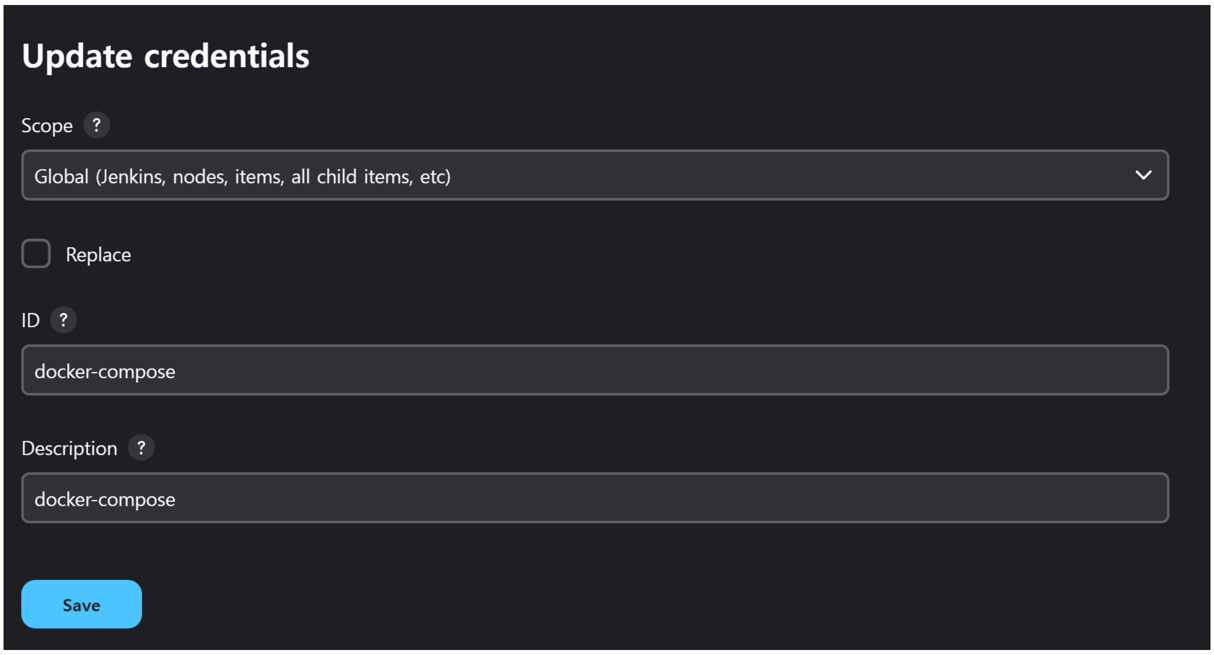
Username: zlxdgus123

☐ Treat username as secret

Password: Concealed

ID: DOCKER_REPO

Description:

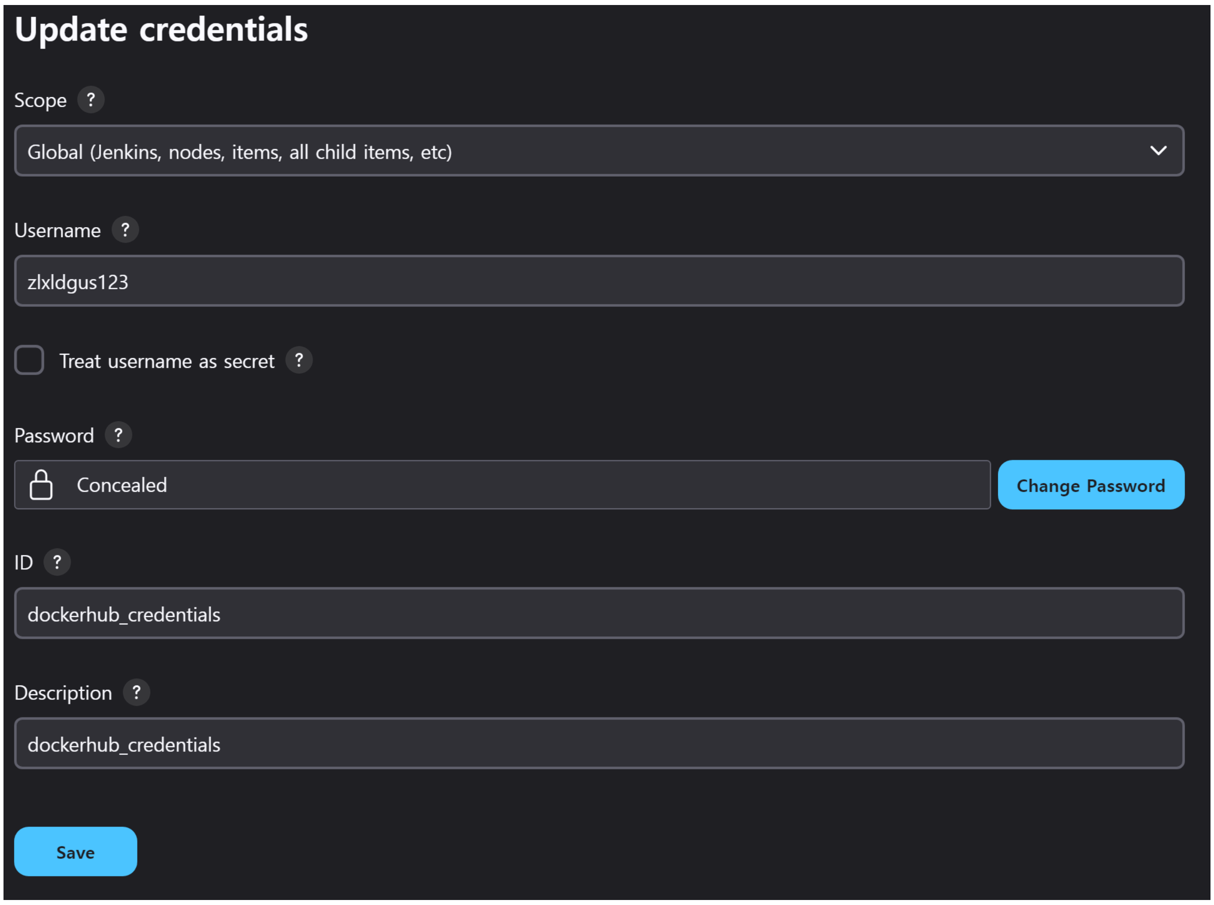

Update credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

☐ Replace

ID: docker-compose

Description: docker-compose


Update credentials

Scope

Global (Jenkins, nodes, items, all child items, etc)

Username

zlxldgus123

Treat username as secret

Password

Concealed

ID

dockerhub_credentials

Description

dockerhub_credentials

5.3. Jenkins Pipeline (Backend)

```
pipeline {
    agent any

    environment {
        DOCKER_HUB_NAMESPACE = "zlxldgus123"
        DOCKER_TAG = "latest"
        DEPLOY_DIR = "/home/ubuntu/deploy"
        GIT_BRANCH = "back/develop"
    }

    stages {
        stage('Git Clone') {
            steps {
                script {
                    git branch: "${GIT_BRANCH}", credentialsI
                }
            }
        }

        stage('Show Git Branch') {
            steps {
                script {
                    def branch = sh(script: 'git rev-parse --
                    echo "Current Git Branch: ${branch}"
                }
            }
        }

        stage('Show Directory Structure') {
            steps {
                script {
                    sh 'find .'
                }
            }
        }

        stage('Show Dockerfiles') {
            steps {
                script {
                    def services = ['battle', 'chat', 'proble
                    for (service in services) {
                        def dockerfilePath = "backend/${servi
                        def dockerfileExists = fileExists(doc

                        if (dockerfileExists) {
                            echo "Dockerfile for ${service} e
                            sh "cat ${dockerfilePath}"
                        } else {
                            echo "Dockerfile for ${service} d
                        }
                    }
                }
            }
        }
        stage('Show Docker Compose File') {
            steps {
                script {
                    def dockerComposeFilePath = "${WORKSPACE}
                    def dockerComposeFileExists = fileExists(

                    if (dockerComposeFileExists) {
                        echo "docker-compose.yml exists, disp
                        sh "cat ${dockerComposeFilePath}"
                    } else {
                        echo "docker-compose.yml does not exi
                    }
                }
            }
        }

        stage('Build Docker Images and Push') {
            steps {
                script {
                    def services = ['battle', 'chat', 'proble
                    for (service in services) {
                        def image = "${DOCKER_HUB_NAMESPACE}/
                        def dockerfilePath = "backend/${servi

                        // Check if Dockerfile exists
                        def dockerfileExists = fileExists(doc

                        if (dockerfileExists) {
                            echo "Dockerfile for ${service} e
                            withCredentials([usernamePassword
                                sh """
                                docker build -t ${image} -f $
                                docker login -u \$DOCKER_HUB_
                                docker push ${image}
                                """
                            }
                        } else {
                            echo "Dockerfile for ${service} d
                        }
                    }
                }
            }
        }

        stage('Deploy') {
    steps {
        script {
            // Docker Compose 설치 여부 확인 및 설치
            sh '''
            export PATH=$PATH:/usr/local/bin
            if ! command -v docker-compose &> /dev/null
            then
                echo "docker-compose could not be found. Inst
                curl -L "https://github.com/docker/compose/re
                chmod +x /usr/local/bin/docker-compose
            else
                echo "docker-compose is already installed."
            fi
            '''

            // DEPLOY_DIR 디렉토리가 존재하지 않으면 생성하고 파일 목
            sh '''
            echo "Current User: $(whoami)"
            if [ ! -d "${DEPLOY_DIR}" ]; then
                mkdir -p ${DEPLOY_DIR}
            fi

            cd ${DEPLOY_DIR}
            '''

            // Jenkins 크리덴셜에서 docker-compose.yml 파일 복사
            withCredentials([file(credentialsId: 'docker-comp
                sh '''
                cp "$DOCKER_COMPOSE_FILE" "${DEPLOY_DIR}/dock

                echo "Directory Contents after copying docker
                ls -al ${DEPLOY_DIR}

                if [ -f "${DEPLOY_DIR}/docker-compose.yml" ];
                    echo "docker-compose.yml exists at ${DEPL
                    cat ${DEPLOY_DIR}/docker-compose.yml
                else
                    echo "docker-compose.yml does not exist a
                    exit 1
                fi

                # 강제로 파일 시스템 동기화
                sync

                docker-compose down
                docker-compose up --build -d
                '''
            }
        }
    }
}


        stage('Docker Cleanup') {
            steps {
                script {
                    sh '''
                    echo "Cleaning up old Docker images..."
                    docker images --filter "dangling=false" -
                    '''
                }
            }
        }
    }

    post {
        always {
            echo 'Cleaning up...'
            cleanWs()
        }
        success {
            echo 'Pipeline succeeded!'
        }
        failure {
            echo 'Pipeline failed!'
        }
    }
}
```

5.4. Jenkins Pipeline (Frontend)

```
pipeline {
    agent any

    environment {
        DOCKER_HUB_NAMESPACE = "zlxldgus123"
        DOCKER_TAG = "latest"
        DEPLOY_DIR = "/home/ubuntu/deploy"
        GIT_BRANCH = "front/develop"
        PROJECT_NAME = "frontend_project"
    }

    stages {
        stage('Git Clone') {
            steps {
                script {
                    git branch: "${GIT_BRANCH}", credentialsI
                }
            }
        }

        stage('Show Git Branch') {
            steps {
                script {
                    def branch = sh(script: 'git rev-parse --
                    echo "Current Git Branch: ${branch}"
                }
            }
        }

        stage('Show Directory Structure') {
            steps {
                script {
                    sh 'find .'
                }
            }
        }

        stage('Show Dockerfile') {
            steps {
                script {
                    def dockerfilePath = "frontend/pjt-fronte
                    def dockerfileExists = fileExists(dockerf

                    if (dockerfileExists) {
                        echo "Dockerfile for frontend exists,
                        sh "cat ${dockerfilePath}"
                    } else {
                        echo "Dockerfile for frontend does no
                    }
                }
            }
        }

        stage('Show Docker Compose File') {
            steps {
                script {
                    def dockerComposeFilePath = "${WORKSPACE}
                    def dockerComposeFileExists = fileExists(

                    if (dockerComposeFileExists) {
                        echo "docker-compose-frontend.yml exi
                        sh "cat ${dockerComposeFilePath}"
                    } else {
```

```
                    echo "docker-compose-frontend.yml doe
                }
            }
        }
    }

    stage('Build Docker Image and Push') {
        steps {
            script {
                def image = "${DOCKER_HUB_NAMESPACE}/fron
                def dockerfilePath = "frontend/pjt-fronte

                // Check if Dockerfile exists
                def dockerfileExists = fileExists(dockerf

                if (dockerfileExists) {
                    echo "Dockerfile for frontend exists,
                    withCredentials([usernamePassword(cre
                        sh """
                        docker build -t ${image} -f ${doc
                        docker login -u \$DOCKER_HUB_USER
                        docker push ${image}
                        """
                    }
                } else {
                    echo "Dockerfile for frontend does no
                }
            }
        }
    }

    stage('Deploy Frontend') {
        steps {
            script {
                // Docker Compose 설치 여부 확인 및 설치
                sh """
                export PATH=\$PATH:/usr/local/bin
                if ! command -v docker-compose &> /dev/nu
```

```
                    then
                        echo "docker-compose could not be fou
                        curl -L "https://github.com/docker/co
                        chmod +x /usr/local/bin/docker-compos
                    else
                        echo "docker-compose is already insta
                    fi
                    """

                    // DEPLOY_DIR 디렉토리가 존재하지 않으면 생성하
                    sh """
                    echo "Current User: \$(whoami)"
                    if [ ! -d "${DEPLOY_DIR}" ]; then
                        mkdir -p ${DEPLOY_DIR}
                    fi

                    cd ${DEPLOY_DIR}

                    # 리포지토리에서 가져온 docker-compose-fronte
                    cp ${WORKSPACE}/docker-compose-frontend.y

                    echo "Directory Contents:"
                    ls -al

                    if [ -f "docker-compose-frontend.yml" ];
                        echo "docker-compose-frontend.yml exi
                        cat docker-compose-frontend.yml
                    else
                        echo "docker-compose-frontend.yml doe
                        exit 1
                    fi

                    docker-compose -p ${PROJECT_NAME} -f dock
                    docker-compose -p ${PROJECT_NAME} -f dock

                    """
                }
            }
```

```
        }

        stage('Docker Cleanup') {
            steps {
                script {
                    sh """
                    echo "Cleaning up old Docker images..."
                    docker images --filter "dangling=false" -
                    """
                }
            }
        }
    }

    post {
        always {
            echo 'Cleaning up...'
            cleanWs()
        }
        success {
            echo 'Pipeline succeeded!'
        }
        failure {
            echo 'Pipeline failed!'
        }
    }
}
```

5.5. Nginx 설정

1. **패키지 업데이트**

   ```
   sudo apt update
   ```

2. **Nginx 설치**

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

3. **Nginx 설정 파일 열기**

```
sudo vim /usr/local/openresty/nginx/conf/nginx.conf
```

4. **설정 파일 내용**

```
worker_processes  1;

events {
    worker_connections  1024;
}

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

    sendfile        on;
    keepalive_timeout  65;

    server {
        listen       80;
        server_name  ssafy11s.com;

        # Redirect to www.ssafy11s.com
        return 301 http://www.ssafy11s.com$request_uri;
    }

    server {
        listen       80;
        server_name  www.ssafy11s.com;
        index index.html;

        location / {
            proxy_pass http://172.19.0.2:80;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
```

```
            proxy_set_header X-Forwarded-Proto $scheme;
            try_files $uri $uri/ /index.html =404;
        }

        location /user/public {
            proxy_pass http://172.19.0.9:8080;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /user {
            access_by_lua_file /etc/nginx/lua/jwt_checke
r.lua;
            proxy_pass http://172.19.0.9:8080;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /profile {
            access_by_lua_file /etc/nginx/lua/jwt_checke
r.lua;
            proxy_pass http://172.19.0.9:8080;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /problem {
            proxy_pass http://172.19.0.10:8081;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /multi {
            access_by_lua_file /etc/nginx/lua/jwt_checke
r.lua;
            proxy_pass http://172.19.0.11:8082;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_http_version 1.1;
        }

        location /ws-multi {
            proxy_pass http://172.19.0.11:8082;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";
        }

        location /rank {
            access_by_lua_file /etc/nginx/lua/jwt_checke
r.lua;
            proxy_pass http://172.19.0.12:8083;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /battle {
            access_by_lua_file /etc/nginx/lua/jwt_checke
r.lua;
            proxy_pass http://172.19.0.13:8084;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /ws-battle {
            proxy_pass http://172.19.0.13:8084;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";
        }

        location /ws-chat {
            proxy_pass http://172.19.0.14:8085;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /single/public {
            proxy_pass http://172.19.0.15:8086;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /single {
            access_by_lua_file /etc/nginx/lua/jwt_checke
r.lua;
            proxy_pass http://172.19.0.15:8086;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        listen [::]:443 ssl ipv6only=on; # managed by Ce
rtbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/ssafy11s.c
om/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/ssafy1
1s.com/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 {
        listen 80;
        listen [::]:80;
        return 404; # managed by Certbot
    }
```

```
}
```

5.6. Lua 설정

1. **LuaJIT 설치**

```
sudo apt-get install luajit
```

2. **lua-resty-jwt 설치**

```
mkdir -p /usr/local/share/lua/5.1/resty
wget https://raw.githubusercontent.com/SkyLothar/lua-res
ty-jwt/master/lib/resty/jwt.lua -O /usr/local/share/lua/
5.1/resty/jwt.lua
```

3. **Lua Script**

```
local jwt = require "resty.jwt"

local function verify_jwt()
    local args = ngx.req.get_headers()
    local token = args["Authorization"]

    if not token then
        ngx.status = ngx.HTTP_UNAUTHORIZED
        ngx.say("Missing Authorization header")
        return ngx.exit(ngx.HTTP_UNAUTHORIZED)
    end

    token = token:match("Bearer%s+(.+)")
    if not token then
        ngx.status = ngx.HTTP_UNAUTHORIZED
        ngx.say("Invalid Authorization header")
        return ngx.exit(ngx.HTTP_UNAUTHORIZED)
    end

    local jwt_obj = jwt:verify("7d1b1d6d36d8e6a8f1bda6a7f47
3f87b012b0345a1b5f", token)
    if not jwt_obj.verified then
        ngx.status = ngx.HTTP_UNAUTHORIZED
        ngx.say("Invalid token")
        return ngx.exit(ngx.HTTP_UNAUTHORIZED)
    end

    ngx.req.set_header("X-User-ID", jwt_obj.payload.sub)
end

return verify_jwt
```