🎃

포팅 메뉴얼

프로젝트 사용도구

이슈 관리 : JIRA

형상 관리 : Gitlab, Github

커뮤니케이션 : Notion, Slack, Mattermost, Discord

디자인 : Figma

UCC : movavi

CI/CD : Jenkins

개발 : IntelliJ, VScode

개발 환경

Frontend

React	18.2.0
Node.js	18.16.0
VSCode	1.77.0
tailwind	3.3.1
npm	8.19.2
eslint	8.36.0
react-redux	8.0.5

Backend

Spring Boot	2.7.11
Java	11
IntelliJ	2022.3.1
FastAPI	0.95.0
Python	latest
MySQL	8.0.32
Redis	latest

micro-service

```
micro service 구조
  ├common
  ├aop
  ├config
  │  └exception
  ├controller
  ├dto
  │  ├request
  │  └response
  ├entity
  │  ├member
  │  │  └enums
  │  └shadowing
  ├messagequeue
  │  └dto
  │      └produce
  ├repository
  └service
```

Frontend 설정 파일

.env

```
NEXT_PUBLIC_OPEN_API=
NEXTAUTH_SECRET=
NEXT_PUBLIC_GRAMMER_API=
NEXT_PUBLIC_GOOGLE_OAUTH_ID=
NEXT_PUBLIC_GOOGLE_OAUTH_PW=
NEXT_PUBLIC_TTS_API=
NEXTAUTH_URL=
NEXT_PUBLIC_SERVER_URL=
NEXT_PUBLIC_FAST_API=
NEXT_PUBLIC_AZURE_API=
NEXT_PUBLIC_SOCKET_URL=wss://
```

Backend 설정 파일

Eureka Service

application.yml

```
server:
  port:

spring:
  application:
    name: eureka-service

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
    service-url:
      defaultZone: http://호스트:${server.port}/eureka
```

ApiGateway Service

application.yml

```
server:
  port:

eureka:
  instance:
    prefer-ip-address: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${server.port}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://호스트:eureka포트번호/eureka

spring:
  application:
    name: apigateway-service
  redis:
    host: 호스트
    port:
    password:
  jwt:
    secretKey:

  cloud:
    gateway:
      routes:
        - id: member-service
          uri: lb://MEMBER-SERVICE
          predicates:
            - Path=/member-service/auth/**
          filters:
            - AuthorizationHeaderFilter
            - RewritePath=/member-service/(?<segment>.*), /$\{segment}
        - id: member-service
          uri: lb://MEMBER-SERVICE
          predicates:
            - Path=/member-service/**
          filters:
            - RewritePath=/member-service/(?<segment>.*), /$\{segment}
        - id: challenge-service
          uri: lb://CHALLENGE-SERVICE
          predicates:
            - Path=/challenge-service/auth/**
          filters:
            - AuthorizationHeaderFilter
            - RewritePath=/challenge-service/(?<segment>.*), /$\{segment}
        - id: challenge-service
          uri: lb://CHALLENGE-SERVICE
          predicates:
            - Path=/challenge-service/watch/member-challenges/**
          filters:
            - AuthorizationChallengeFilter
            - RewritePath=/challenge-service/(?<segment>.*), /$\{segment}
        - id: challenge-service
          uri: lb://CHALLENGE-SERVICE
          predicates:
            - Path=/challenge-service/**
          filters:
            - RewritePath=/challenge-service/(?<segment>.*), /$\{segment}
        - id: shadowing-service
          uri: lb://SHADOWING-SERVICE
          predicates:
            - Path=/shadowing-service/auth/**
          filters:
            - AuthorizationHeaderFilter
            - RewritePath=/shadowing-service/(?<segment>.*), /$\{segment}
        - id: shadowing-service
          uri: lb://SHADOWING-SERVICE
          predicates:
            - Path=/shadowing-service/**
          filters:
            - RewritePath=/shadowing-service/(?<segment>.*), /$\{segment}
        - id: chatting-service
          uri: lb://CHATTING-SERVICE
          predicates:
            - Path=/chatting-service/auth/**
          filters:
            - AuthorizationHeaderFilter
            - RewritePath=/chatting-service/(?<segment>.*), /$\{segment}
        - id: chatting-service
          uri: lb://CHATTING-SERVICE
          predicates:
            - Path=/chatting-service/**
          filters:
            - RewritePath=/chatting-service/(?<segment>.*), /$\{segment}
        - id: chatting-service
          uri: lb:ws://CHATTING-SERVICE
          predicates:
            - Path=/chatting-service/**
          filters:
            - RewritePath=/chatting-service/(?<segment>.*), /$\{segment}

  rabbitmq:
    host: 호스트
    port:
    username:
    password:
  config:
    import: "optional:configserver:"

management:
  endpoints:
    web:
      exposure:
        include: busrefresh
```

bootstrap.yml

```
spring:
  cloud:
    config:
      uri: http://호스트:config-service포트번호
      name: config-service
#  profiles:
#    active: dev
```

Config Service

application.yml

```
server:
  port:
spring:
  application:
    name: config-service
  rabbitmq:
    host: 호스트
    port:
    username:
    password:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/
          username:
          password:

management:
  endpoints:
    web:
      exposure:
        include: busrefresh
```

Member Service

application.yml

```
server:
  port: 0

cloud:
  aws:
    s3:
      bucket:
    region:
      static: ap-northeast-2
    stack:
      auto: false
    credentials:
      access-key:
      secret-key:

spring:
  application:
    name: member-service
  servlet:
    multipart:
      max-file-size: 10MB # 파일 하나 당 최대 사이즈
      max-request-size: 20MB # 요청 당 최대 사이즈
  rabbitmq:
    host: 호스트
    port:
    username:
    password:
  redis:
    host: 호스트
    port:
    password:
  kafka:
    producer:
      bootstrap-servers: 호스트:포트번호

  datasource:
    url: jdbc:mysql://호스트:포트번호/opener?useSSL=false&serverTimezone=UTC
    username:
    password:
```

```
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true

    database : mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    properties:
      hibernate:
        format_sql: true

  jwt:
    secretKey:
  img:
    baseurl:

  mail:
    host: smtp.gmail.com
    port: 587
    username:
    password:
    properties:
      mail.smtp.auth: true
      mail.smtp.starttls.enable: true


eureka:
  instance:
    prefer-ip-address: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://호스트:eureka포트번호/eureka

logging:
  level:
    com.example.memberservice.client: DEBUG


management:
  endpoints:
    web:
      exposure:
        include: busrefresh
  health:
    mail:
      enabled: false
```

bootstrap.yml

```
spring:
  cloud:
    config:
      uri: http://호스트:config-service호트번호
      name: member-service
#  profiles:
#    active: dev
```

Shadowing Service

application.yml

```
server:
  port: 0

spring:
  application:
    name: shadowing-service
  rabbitmq:
    host: 호스트
    port:
    username:
    password:
  datasource:
    url: jdbc:mysql://호스트:포트번호/opener?useSSL=false&serverTimezone=UTC
    username:
    password:
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true
    defer-datasource-initialization: true

    database : mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    properties:
      hibernate:
        format_sql: true
  kafka:
    producer:
      bootstrap-servers: 호스트:포트번호
app:
  firebase-configuration-file:
  firebase-bucket :
eureka:
  instance:
    prefer-ip-address: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://호스트:eureka포트번호/eureka

logging:
  level:
    com.example.shadowingservice: DEBUG


management:
  endpoints:
    web:
      exposure:
        include: busrefresh
```

bootstrap.yml

```
spring:
  cloud:
    config:
      uri: http://호스트:config-service호트번호
      name: shadowing-service
#  profiles:
#    active: dev
```

Chatting Service

application.yml

```
server:
  port: 0

spring:
  application:
    name: chatting-service
  rabbitmq:
    host: 호스트
    port:
    username:
    password:
  redis:
    host:
```

```
    port:
    password:
  kafka:
    producer:
      bootstrap-servers: 호스트:포트번호


  datasource:
    url: jdbc:mysql://호스트:포트번호/opener?useSSL=false&serverTimezone=UTC
    username:
    password:
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true

    database : mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    properties:
      hibernate:
        format_sql: true
  img:
    baseurl:


eureka:
  instance:
    prefer-ip-address: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://호스트:eureka포트번호/eureka

logging:
  level:
    com.example.chattingservice.client: DEBUG
    org.springframework.web.socket: DEBUG


management:
  endpoints:
    web:
      exposure:
        include: busrefresh
```

bootstrap.yml

```
spring:
  cloud:
    config:
      uri: http://호스트:config-service호트번호
      name: chatting-service
#  profiles:
#    active: dev
```

Challenge Service

application.yml

```
server:
  port: 0

spring:
  application:
    name: challenge-service
  rabbitmq:
    host:
    port:
    username:
    password:
```

```
  kafka:
    producer:
      bootstrap-servers: 호스트:포트번호
  datasource:
    url: jdbc:mysql://호스트:포트번호/opener?useSSL=false&serverTimezone=UTC
    username:
    password:
    driver-class-name: com.mysql.cj.jdbc.Driver
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
      enabled: true # MultipartResolver 사용을 활성화
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true

    database : mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    properties:
      hibernate:
        format_sql: true
app:
  firebase-configuration-file:
  firebase-bucket :
eureka:
  instance:
    prefer-ip-address: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://호스트:eureka포트번호/eureka

logging:
  level:
    com.example.memberservice.client: DEBUG


management:
  endpoints:
    web:
      exposure:
        include: busrefresh
```

bootstrap.yml

```
spring:
  cloud:
    config:
      uri: http://호스트:config-service호트번호
      name: challenge-service
#  profiles:
#    active: dev
```

Fast API 설정 파일

env

```
MYSQL_USER=
MYSQL_PASSWORD=
MYSQL_HOST=호스트:포트번호
MYSQL_DB=
```

로컬 빌드

Frontend

터미널 명령어 실행

```
$ npm install
$ npm run dev
```

Backend

```
cd back/eureka-service
./gradlew bootRun

cd ../apigateway-service
./gradlew bootRun

cd ../config-service
./gradlew bootRun

cd ../member-service
./gradlew bootRun

cd ../challenge-service
./gradlew bootRun

cd ../shadowing-service
./gradlew bootRun

cd ../chatting-service
./gradlew bootRun
```

EC2

도커 설치

```
$ sudo apt-get update
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
$ echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

도커 사용자 그룹 설정

```
$ usermod -aG docker $USER

$ groups $USER

$ service docker restart
```

SSL 인증서 발급

```
$ sudo apt-get install letsencrypt
$ letsencrypt certonly --standalone -d {도메인}
```

Bridge Network 생성

```
docker network create --gateway Docker네트워크게이트웨이IP주소설정 --subnet Docker네트워크IP서브넷설정 네트워크이름
```

MYSQL DB 띄우기

```
docker run --name member_mysql -e MYSQL_ROOT_PASSWORD=비밀번호 --network 네트워크이름 -d -p 포트번호:3306 mysql:8.0.32
docker run --name challenge_mysql -e MYSQL_ROOT_PASSWORD=비밀번호 --network 네트워크이름 -d -p 포트번호:3306 mysql:8.0.32
docker run --name shadowing_mysql -e MYSQL_ROOT_PASSWORD=비밀번호  --network 네트워크이름 -d -p 포트번호:3306 mysql:8.0.32
docker run --name chatting_mysql -e MYSQL_ROOT_PASSWORD=비밀번호  --network 네트워크이름 -d -p 포트번호:3306 mysql:8.0.32
```

MYSQL 접속 후 계정과 데이터베이스 설정

```
$ docker exec -it mysql컨테이너이름 bash
$ mysql -u root -p // 루트계정으로 접속
Enter password: // 컨테이너 띄울 때 입력한 MYSQL_ROOT_PASSWORD 입력
```

```
$ create user '유저'@'%' identified by '비밀번호'; // 계정 생성
$ grant all privileges on *.* to '유저'@'%'; // 권한 부여
$ FLUSH PRIVILEGES; // 권한 반영
```

Redis 설치

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

docker exec -it redis bash
redis-cli
AUTH 비밀번호`
config set notify-keyspace-events Ex //키 만료 이벤트 구독
```

RabbitMQ 설치

```
docker run --network 네트워크이름 -d -p 포트번호:15672 -p 포트번호:5672 --name rabbitmq rabbitmq
```

Kafka 설치

Kafka connector 설정

```
//Confluent Hub 이용해 jdbc-connector 다운 후 설치
cd kafka
wget http://client.hub.confluent.io/confluent-hub-client-latest.tar.gz
tar -xvf confluent-hub-client-latest.tar.gz

//mysql connector 설치
cd ..
wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-8.0.27.tar.gz
tar -xvf mysql-connector-java-8.0.27.tar.gz

//mysql connector 넣어주기
cp mysql-connector-java-8.0.27/mysql-connector-java-8.0.27.jar /경로/confluent/component/confluentinc-kafka-connect-jdbc/lib
```

docker-compose.yml

```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.3.0
    hostname: zookeeper
    container_name: zookeeper
    ports:
      - "포트번호:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 포트번호
      ZOOKEEPER_TICK_TIME: 2000
    networks:
      - 네트워크


  broker:
    image: confluentinc/cp-kafka:7.3.0
    hostname: broker
    container_name: broker
    depends_on:
      - zookeeper
    ports:
      - "29092:29092"
      - "포트번호:9092"
      - "9101:9101"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:zookeeper포트번호'
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://broker:29092,PLAINTEXT_HOST://호스트:포트번호
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
      KAFKA_JMX_PORT: 9101
      KAFKA_JMX_HOSTNAME: localhost
      KAFKA_LOG4J_ROOT_LOGLEVEL: "WARN"
    networks:
      - opener-network


  schema-registry:
    image: confluentinc/cp-schema-registry:7.3.0
    hostname: schema-registry
    container_name: schema-registry
    depends_on:
      - broker
    ports:
      - "포트번호:8081"
    environment:
      SCHEMA_REGISTRY_HOST_NAME: schema-registry
      SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS: 'broker:29092'
      SCHEMA_REGISTRY_LISTENERS: http://0.0.0.0:8081
    networks:
      - opener-network


  connect:
    image: confluentinc/cp-kafka-connect:7.0.1
    ports:
      - 포트번호:8083
    container_name: connect
    environment:
      CONNECT_BOOTSTRAP_SERVERS: broker:29092
      CONNECT_REST_PORT: 포트번호
      CONNECT_GROUP_ID: "quickstart-avro"
      CONNECT_CONFIG_STORAGE_TOPIC: "quickstart-avro-config"
      CONNECT_OFFSET_STORAGE_TOPIC: "quickstart-avro-offsets"
      CONNECT_STATUS_STORAGE_TOPIC: "quickstart-avro-status"
      CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_STATUS_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_KEY_CONVERTER: "org.apache.kafka.connect.json.JsonConverter"
      CONNECT_VALUE_CONVERTER: "org.apache.kafka.connect.json.JsonConverter"
      CONNECT_INTERNAL_KEY_CONVERTER: "org.apache.kafka.connect.json.JsonConverter"
      CONNECT_INTERNAL_VALUE_CONVERTER: "org.apache.kafka.connect.json.JsonConverter"
      CONNECT_REST_ADVERTISED_HOST_NAME: "localhost"
      CONNECT_LOG4J_ROOT_LOGLEVEL: "WARN"
      CONNECT_PLUGIN_PATH: "/usr/share/java,/etc/kafka-connect/jars"
    volumes:
      - ./component/confluentinc-kafka-connect-jdbc/lib:/etc/kafka-connect/jars
    networks:
      - 네트워크

  rest-proxy:
    image: confluentinc/cp-kafka-rest:7.3.0
    depends_on:
      - broker
      - schema-registry
    ports:
      - 포트번호:8082
    hostname: rest-proxy
    container_name: rest-proxy
    environment:
      KAFKA_REST_HOST_NAME: rest-proxy
      KAFKA_REST_BOOTSTRAP_SERVERS: 'broker:29092'
      KAFKA_REST_LISTENERS: "http://0.0.0.0:포트번호"
      KAFKA_REST_SCHEMA_REGISTRY_URL: 'http://schema-registry:8081'
    networks:
      - 네트워크
networks:
  네트워크:
    external:
      name: 네트워크
```

실행

- `docker-compose.yml` 이 있는 경로에서

```
docker-compose up -d
```

Frontend Dockerfile 설정

```
# BUILDER
FROM node:18.16.0-alpine AS builder

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

EXPOSE 3000

CMD ["npm", "start"]
```

Backend Dockerfile 설정

eureka-service

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/eureka-service-0.0.1-SNAPSHOT.jar EurekaService.jar
ENTRYPOINT ["java", "-jar", "EurekaService.jar"]
```

apigateway-service

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/apigateway-service-0.0.1-SNAPSHOT.jar ApigatewayService.jar
ENTRYPOINT ["java", "-jar", "ApigatewayService.jar"]
```

config-service

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/config-service-0.0.1-SNAPSHOT.jar ConfigService.jar
ENTRYPOINT ["java", "-jar", "ConfigService.jar"]
```

member-service

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/member-service-0.0.1-SNAPSHOT.jar MemberService.jar
ENTRYPOINT ["java", "-jar", "MemberService.jar"]
```

shadowing-service

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/shadowing-service-0.0.1-SNAPSHOT.jar ShadowingService.jar
ENTRYPOINT ["java", "-jar", "ShadowingService.jar"]
```

chatting-service

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/chatting-service-0.0.1-SNAPSHOT.jar ChattingService.jar
ENTRYPOINT ["java", "-jar", "ChattingService.jar"]
```

challenge-service

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/challenge-service-0.0.1-SNAPSHOT.jar ChallengeService.jar
ENTRYPOINT ["java", "-jar", "ChallengeService.jar"]
```

Fast API

```
FROM python:latest

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

EXPOSE 9000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "9000"]
```

Nginx

Dockerfile

```
FROM nginx:1.23.4-alpine

COPY ./conf /etc/nginx/
```

nginx.conf

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    client_max_body_size 50M;
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*.conf;
    server_names_hash_bucket_size 64;
}
```

default.conf

```
upstream back {
    server apigateway-service:8000;
}

upstream front {
    server front:3000;
}

upstream fastapi {
    server fastapi:9000;
}

server {
    listen       80;
    server_name  k8c1041.p.ssafy.io;

    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name k8c1041.p.ssafy.io;

    location / {
        proxy_pass http://front;
        proxy_redirect off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /fast {
        proxy_pass http://fastapi/fast;
        proxy_redirect off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /member-service {
        proxy_pass http://back/member-service;
        proxy_redirect off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /challenge-service {
        proxy_pass http://back/challenge-service;
        proxy_redirect off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /shadowing-service {
        proxy_pass http://back/shadowing-service;
        proxy_redirect off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /chatting-service {
        proxy_pass http://back/chatting-service;
        proxy_redirect off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /chatting-service/user-chat {
        proxy_pass http://back/chatting-service/user-chat;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    ssl_certificate /etc/letsencrypt/live/k8c1041.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k8c1041.p.ssafy.io/privkey.pem;

}
```

Jenkins 설정

Jenkins 설치 후 시작

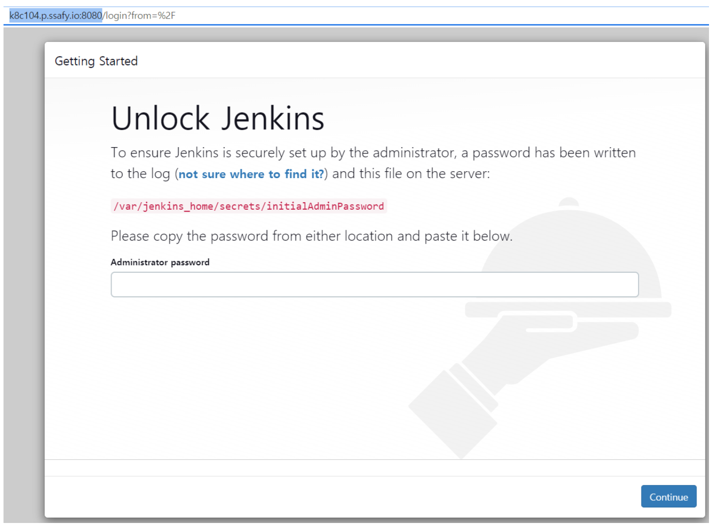
```
docker run --name jenkins -d -p 포트:8080 -v /home/ubuntu/volumes/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.soc
docker start jenkins
```

Jenkins 안에 Docker, Docker Compose 설치

```
//Jenkins 접속
docker exec -it jenkins bash

//Jenkins 안에 Docker 설치
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

//Docker Compose 설치
sudo curl -L "https://github.com/docker/compose/releases/download/{VERSION}/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/
sudo chmod +x /usr/local/bin/docker-compose
```

Jenkins 접속

- 호스트:포트



k8c104.p.ssafy.io:8080/login?from=%2F

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

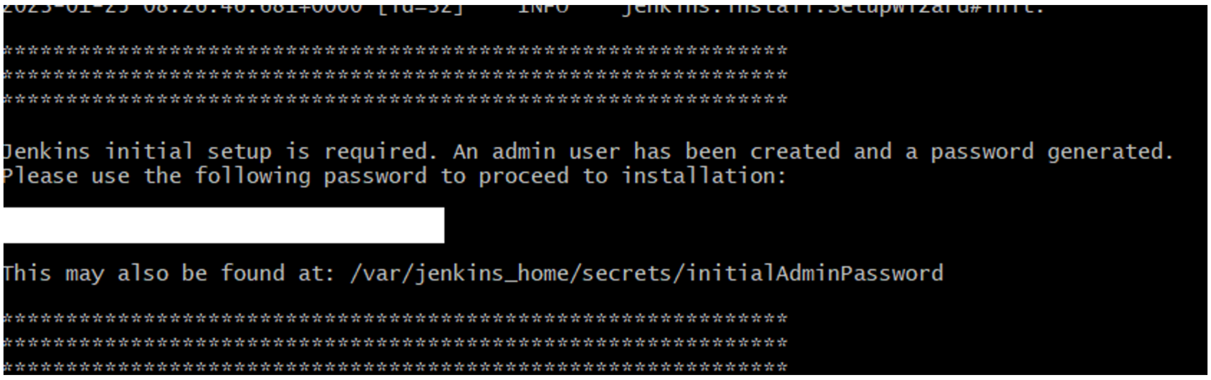
`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

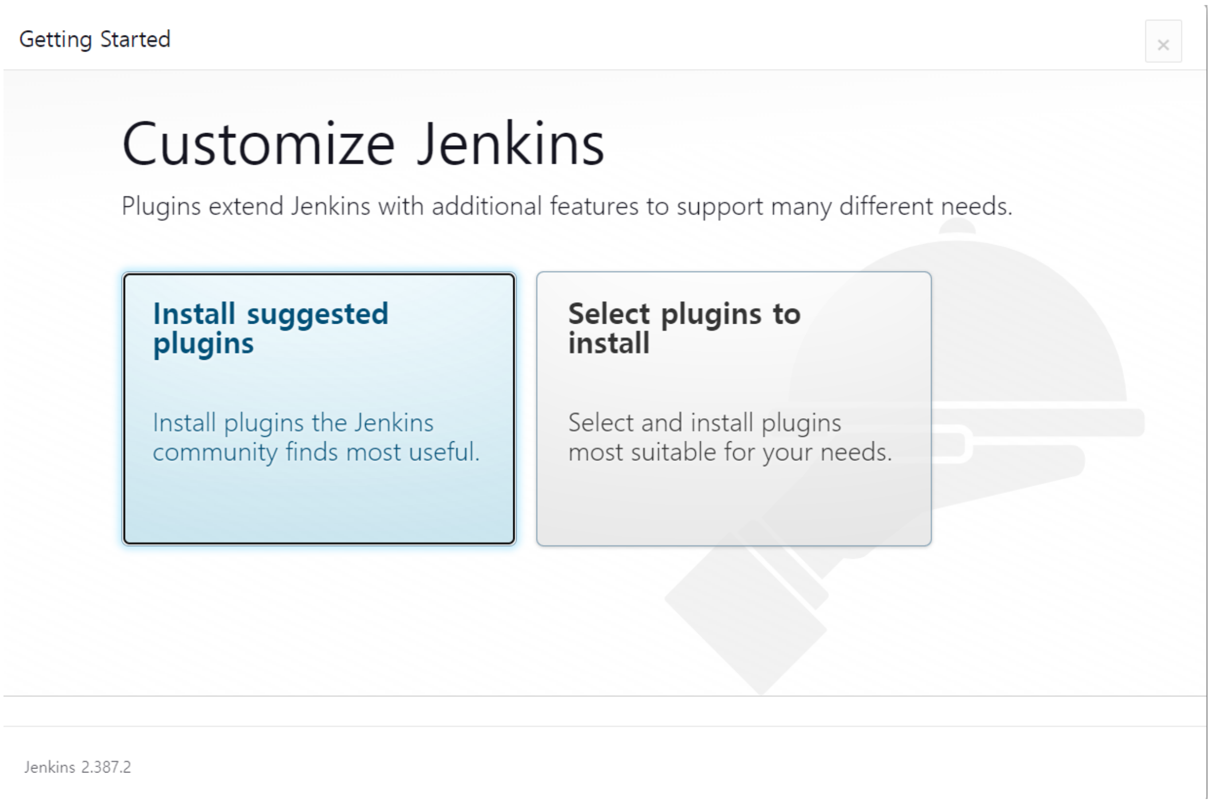
Ubuntu에서 jenkins 비밀번호 확인

```
docker logs jenkins
```



```
2023-01-25 08:26:46.081+0000 [id=32]    INFO    jenkins.install.SetupWizard#init:

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

[redacted]

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************
```

설치 진행



Getting Started

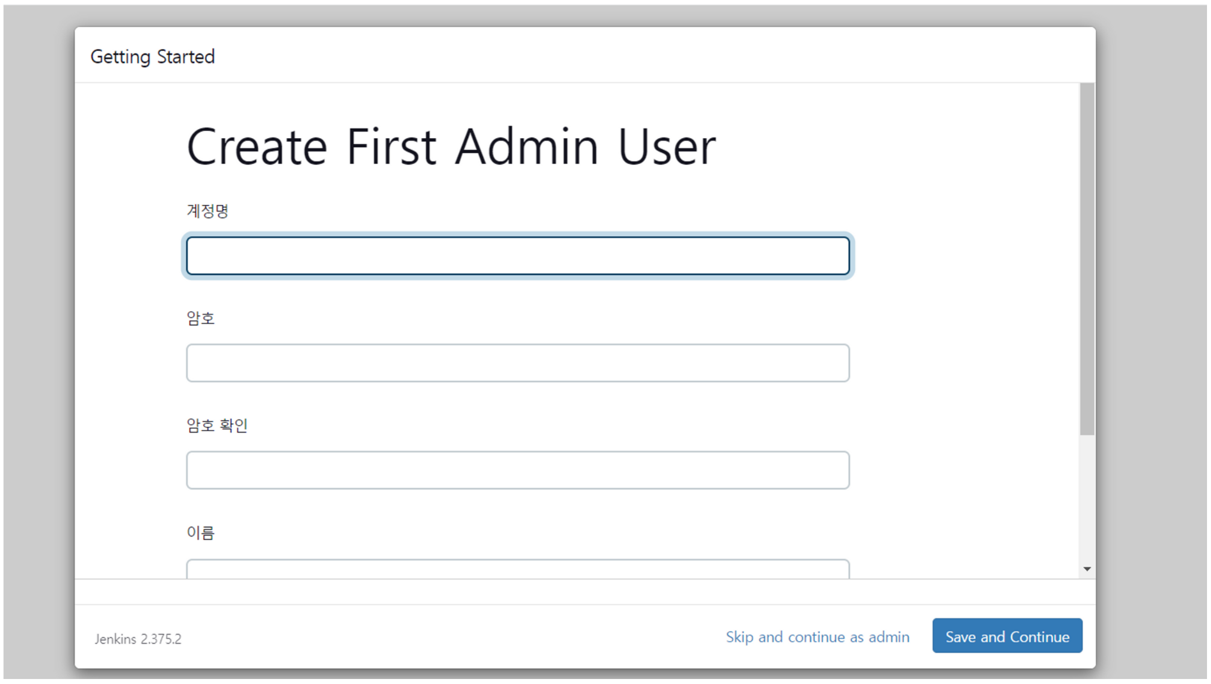
Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.387.2

Install suggested plugins 클릭

계정생성


Getting Started

Create First Admin User

계정명

암호

암호 확인

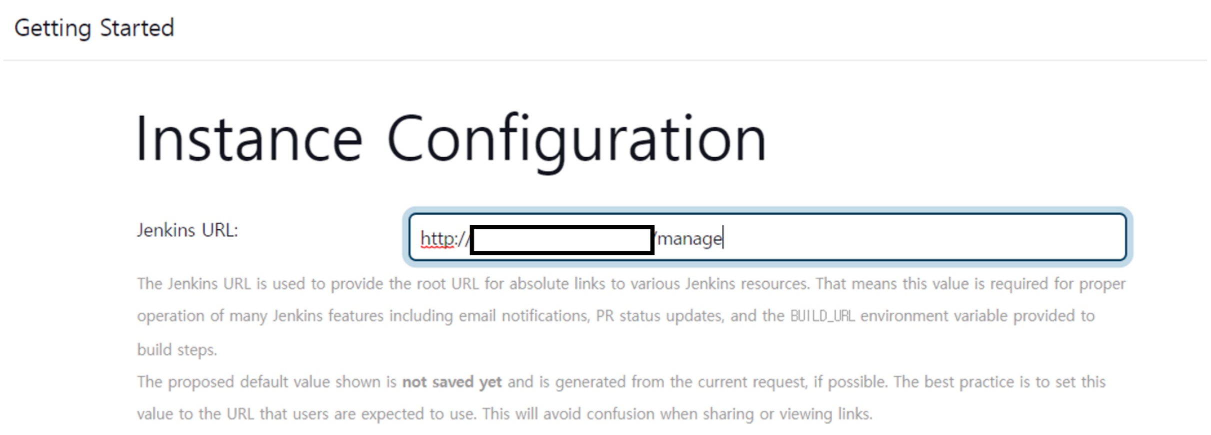
이름

Jenkins 2.375.2

Skip and continue as admin

Save and Continue

Jenkins URL 설정

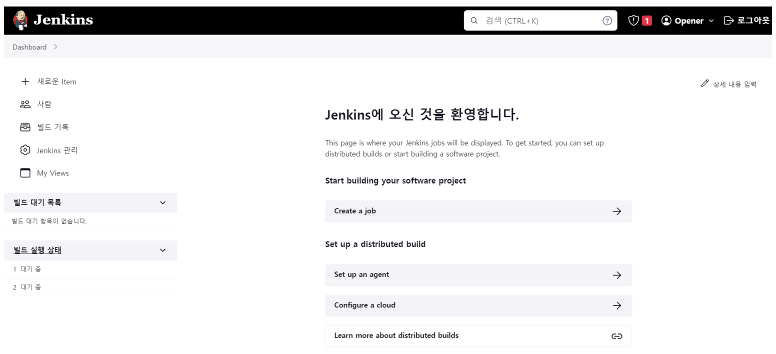

Getting Started

Instance Configuration

Jenkins URL: http://[redacted]/manage

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.



빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

Jenkins에 오신 것을 환영합니다.

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

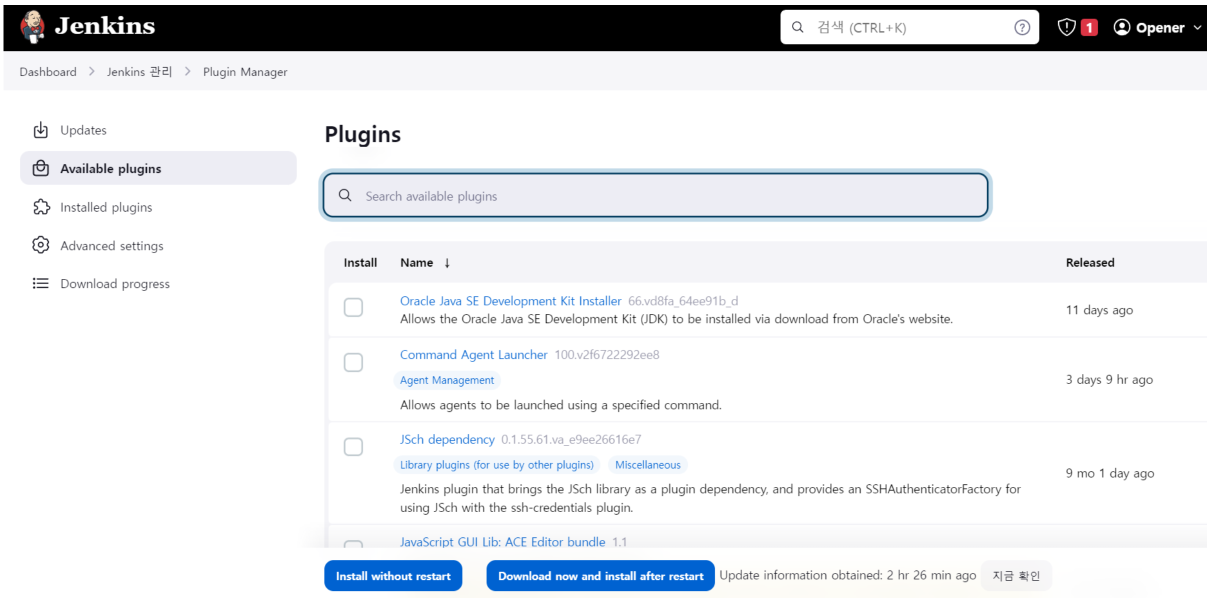
Set up an agent

Configure a cloud

Learn more about distributed builds

플러그인 설치

- Dashboard > Jenkins 관리 > 플러그인 관리 > Available plugins



Install	Name ↓	Released
☐	Oracle Java SE Development Kit Installer 66.vd8fa_64ee91b_d Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	11 days ago
☐	Command Agent Launcher 100.v2f6722292ee8 Agent Management Allows agents to be launched using a specified command.	3 days 9 hr ago
☐	JSch dependency 0.1.55.61.va_e9ee26616e7 Library plugins (for use by other plugins) Miscellaneous Jenkins plugin that brings the JSch library as a plugin dependency, and provides an SSHAuthenticatorFactory for using JSch with the ssh-credentials plugin.	9 mo 1 day ago
☐	JavaScript GUI Lib: ACE Editor bundle 1.1	

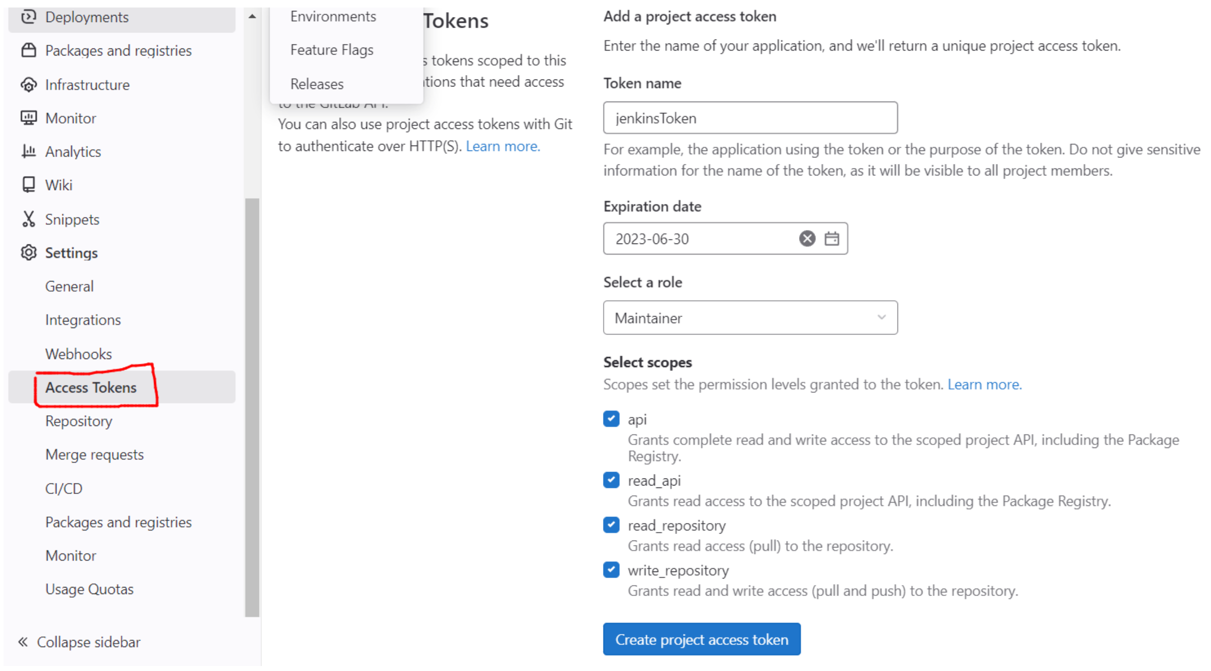
Update information obtained: 2 hr 26 min ago

- Gitlab 검색해 나온 플러그인 전부, Publish Over SSH, Mattermost Notification 설치

```
$ docker start jenkins
```

GitLab

Gitlab Project Access Token발급



s tokens scoped to this
tions that need access
to the GitLab API.
You can also use project access tokens with Git to authenticate over HTTP(S). Learn more.

Add a project access token

Enter the name of your application, and we'll return a unique project access token.

Token name

jenkinsToken

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

Expiration date

2023-06-30

Select a role

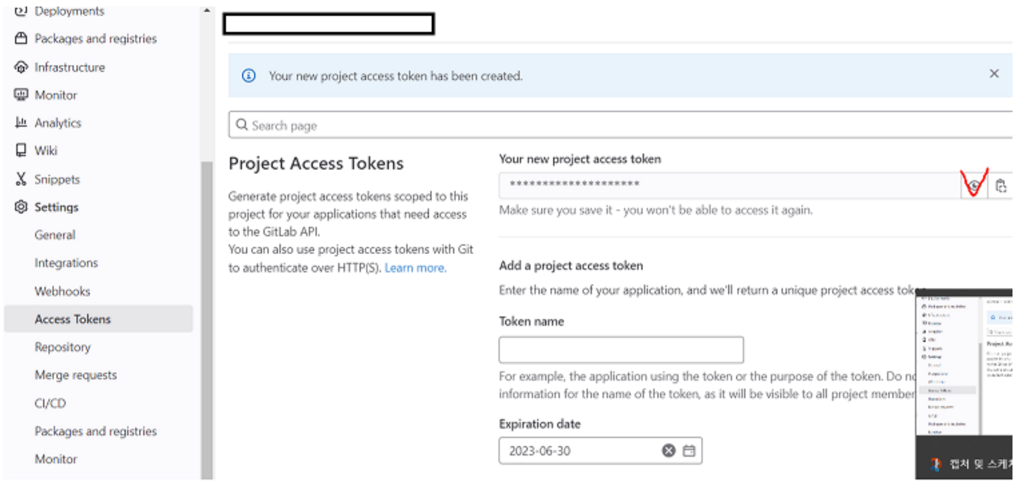
Maintainer

Select scopes

Scopes set the permission levels granted to the token. Learn more.

- [✓] api
  Grants complete read and write access to the scoped project API, including the Package Registry.
- [✓] read_api
  Grants read access to the scoped project API, including the Package Registry.
- [✓] read_repository
  Grants read access (pull) to the repository.
- [✓] write_repository
  Grants read and write access (pull and push) to the repository.

- 해당 repository에서 Project Access Token 발급



[redacted]

Your new project access token has been created.

Search page

Project Access Tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API.
You can also use project access tokens with Git to authenticate over HTTP(S). Learn more.

Your new project access token
****
Make sure you save it - you won't be able to access it again.

Add a project access token
Enter the name of your application, and we'll return a unique project access tok

Token name

For example, the application using the token or the purpose of the token. Do no information for the name of the token, as it will be visible to all project member

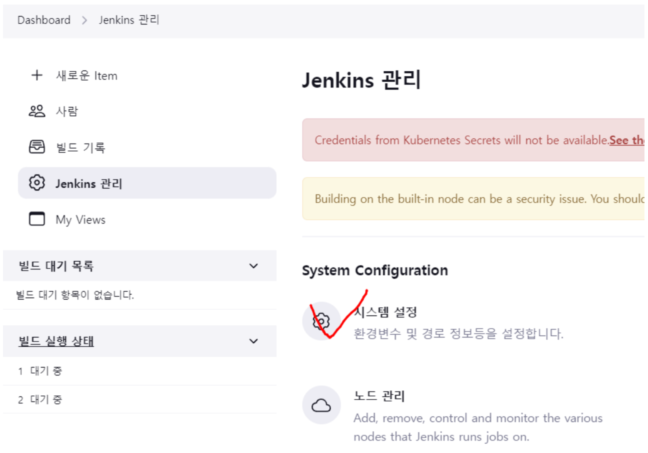
Expiration date
2023-06-30

- 토큰 확인

Jenkins

- Jenkins에 발급받은 토큰을 등록

Jenkins 관리 페이지 - 시스템 설정



빌드 대기 항목이 없습니다.

1 대기 중
2 대기 중

Jenkins 관리

Credentials from Kubernetes Secrets will not be available.See th

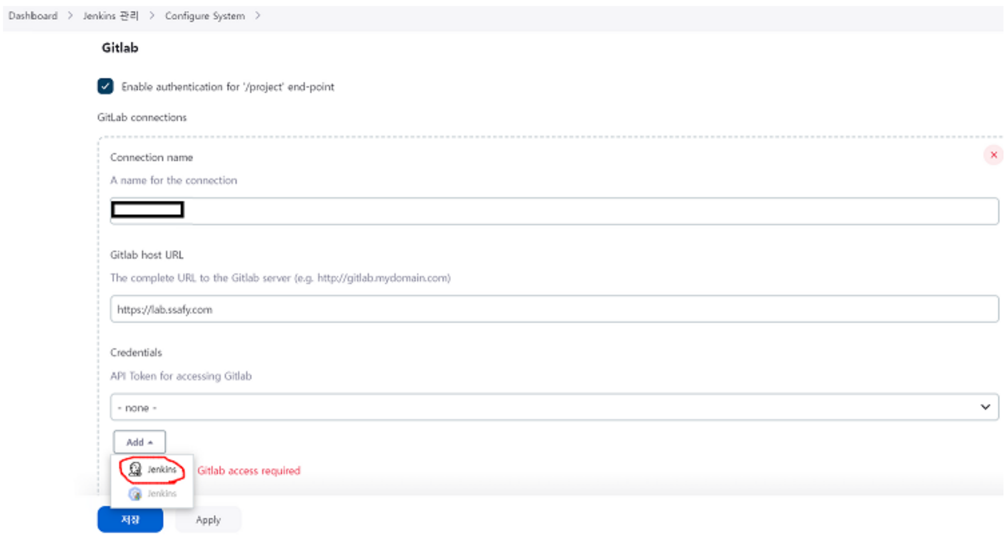
Building on the built-in node can be a security issue. You shoulc

System Configuration

시스템 설정
환경변수 및 경로 정보등을 설정합니다.

노드 관리
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

- 설정 - Gitlab



Dashboard > Jenkins 관리 > Configure System >

Gitlab

[✓] Enable authentication for '/project' end-point

GitLab connections

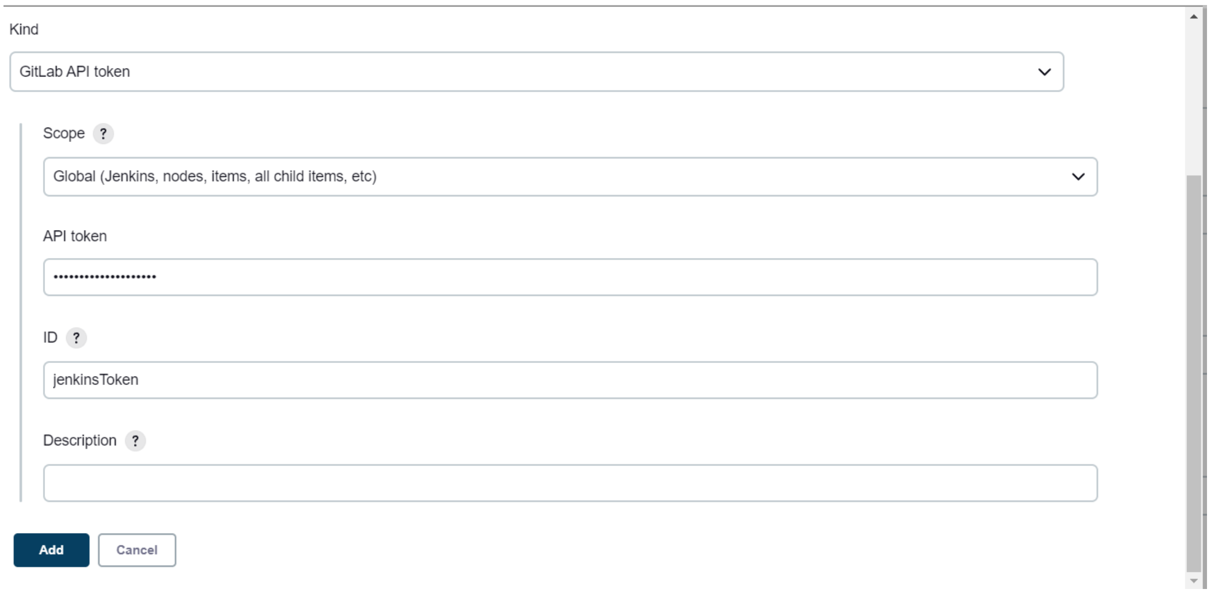
Connection name
A name for the connection
[redacted]

Gitlab host URL
The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)
https://lab.ssafy.com

Credentials
API Token for accessing Gitlab
- none -

Gitlab access required

- 연결할 Repository의 이름과, URL주소를 입력
- Add 버튼 - GitLab API Token을 입력



Kind
GitLab API token

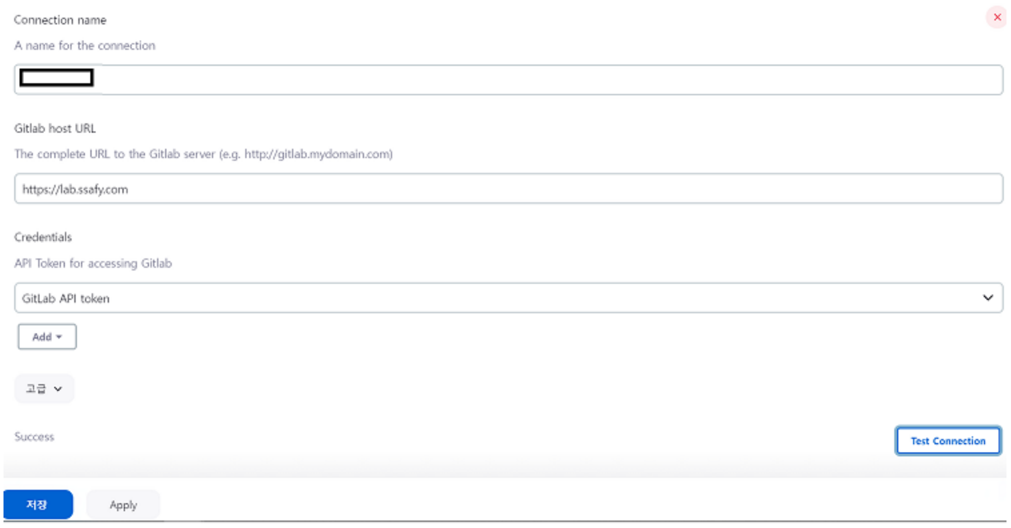
Scope
Global (Jenkins, nodes, items, all child items, etc)

API token
••••••••••••••••••••

ID
jenkinsToken

Description

- `API token` : Gitlab에서 발급한 API token 값
- `ID` : 이 보안설정값의 이름
- `Description` : 설명란


Connection name

A name for the connection

[redacted]

Gitlab host URL

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

https://lab.ssafy.com

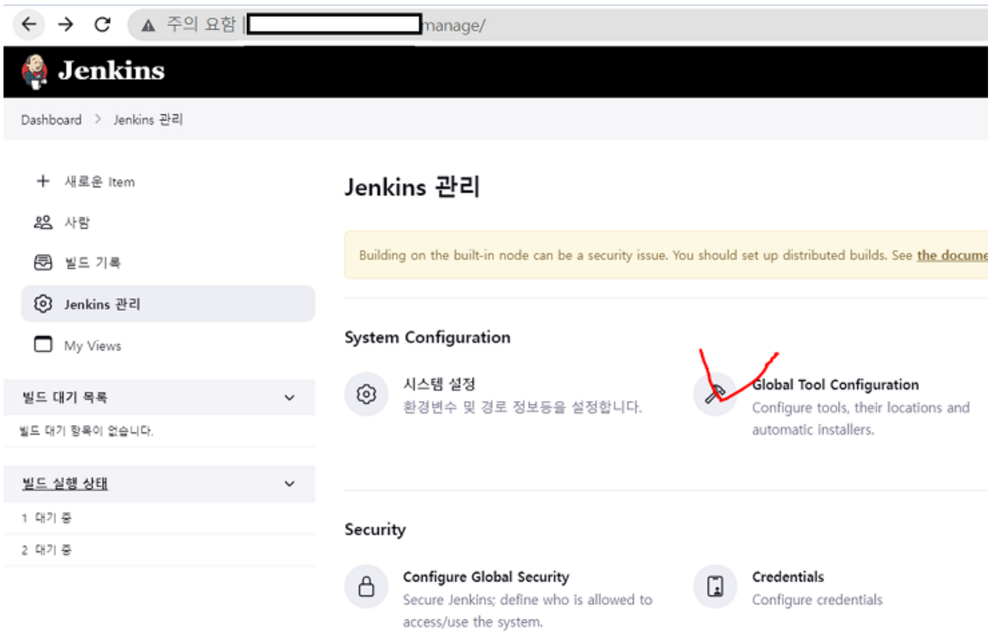
Credentials

API Token for accessing Gitlab

GitLab API token

- 해당 Token을 입력

Jenkins 안에 Gradle 설치



Jenkins 관리

Building on the built-in node can be a security issue. You should set up distributed builds. See the docume

System Configuration

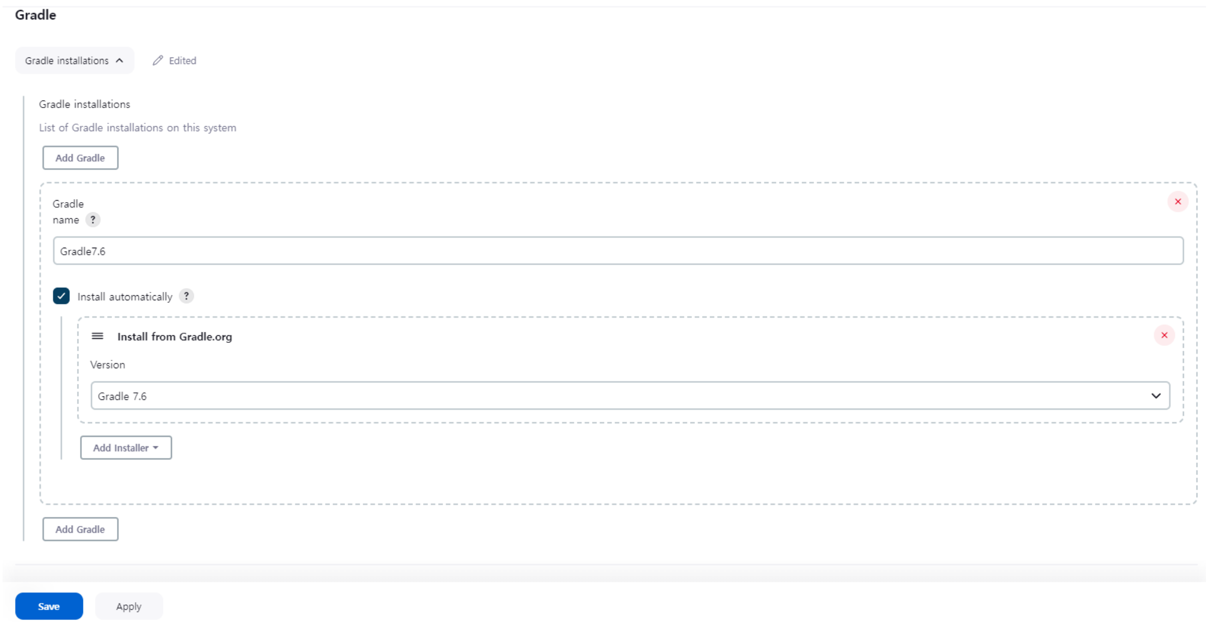
시스템 설정
환경변수 및 경로 정보등을 설정합니다.

Global Tool Configuration
Configure tools, their locations and automatic installers.

Security

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Credentials
Configure credentials


Gradle

Gradle installations

Gradle installations
List of Gradle installations on this system

Gradle
name: Gradle7.6

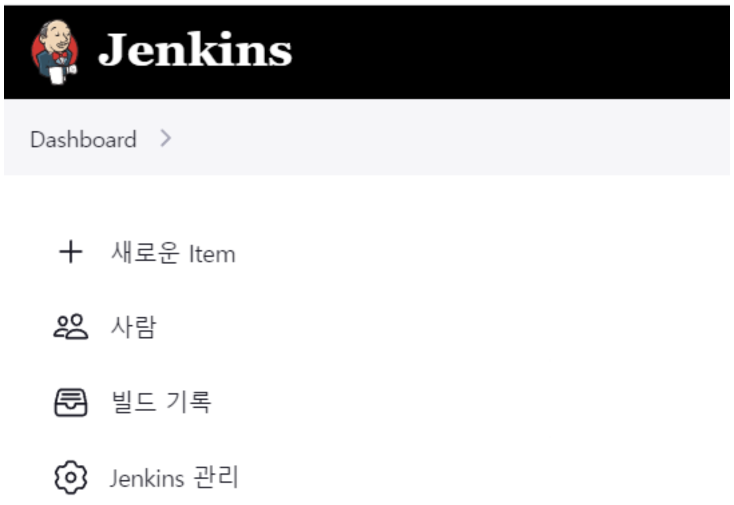
[✓] Install automatically

Install from Gradle.org
Version: Gradle 7.6

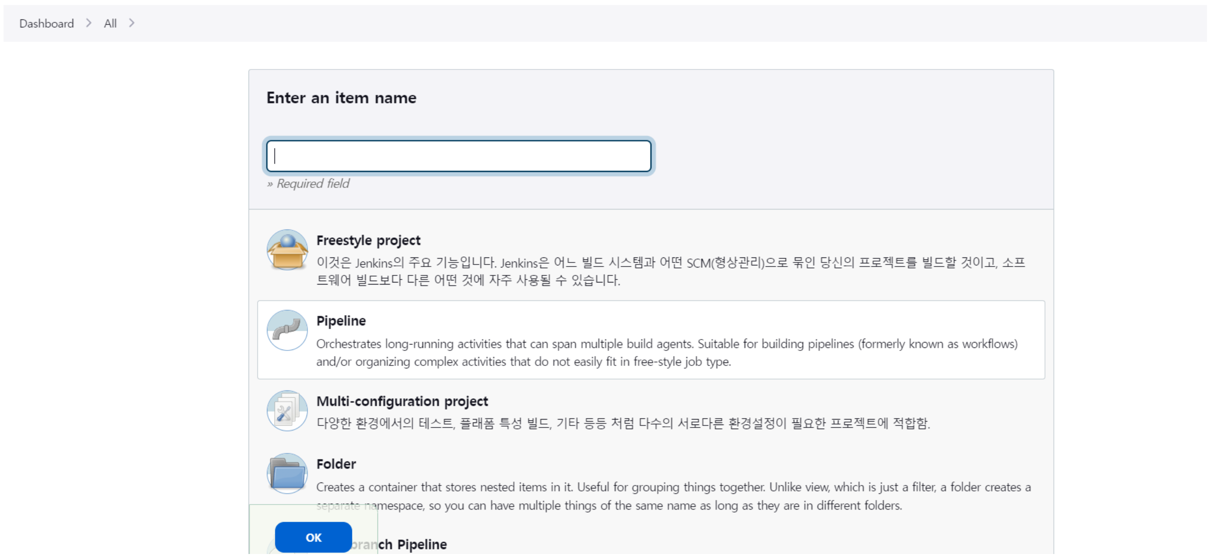
- 설정 후 Save

파이프라인 생성

- `새로운 Item` 클릭



- item 이름 입력 후 `Pipeline` 선택 후 OK


Dashboard > All >

Enter an item name

» Required field

Freestyle project
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

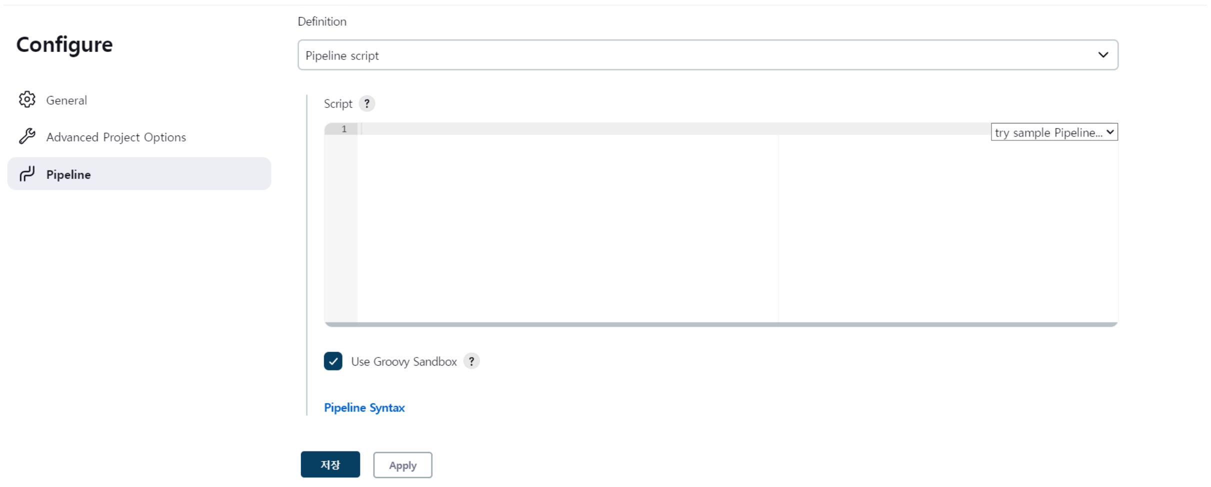
Multi-configuration project
다양한 환경에서의 테스트, 플래폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

ranch Pipeline

pipeline script 작성


Configure

General
Advanced Project Options
Pipeline

Definition: Pipeline script

Script ?

try sample Pipeline...

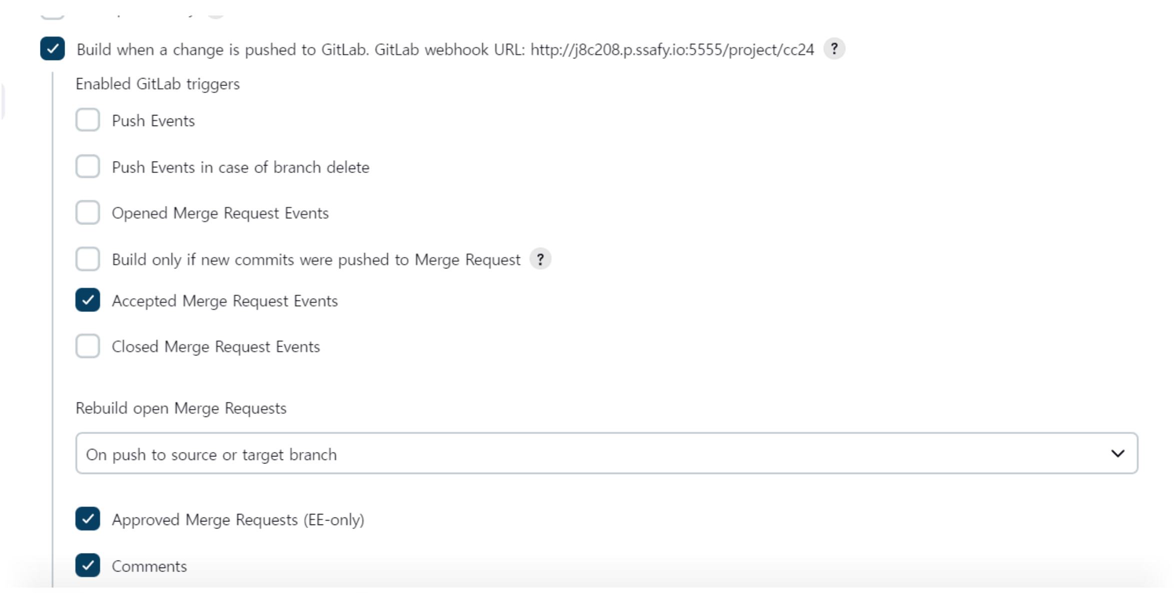
☑ Use Groovy Sandbox ?

Pipeline Syntax

저장 Apply

Dashboard > item이름 > 구성

빌드 트리거 등록



- [✓] Build when a change is pushed to GitLab. GitLab webhook URL: http://j8c208.p.ssafy.io:5555/project/cc24 ?

Enabled GitLab triggers

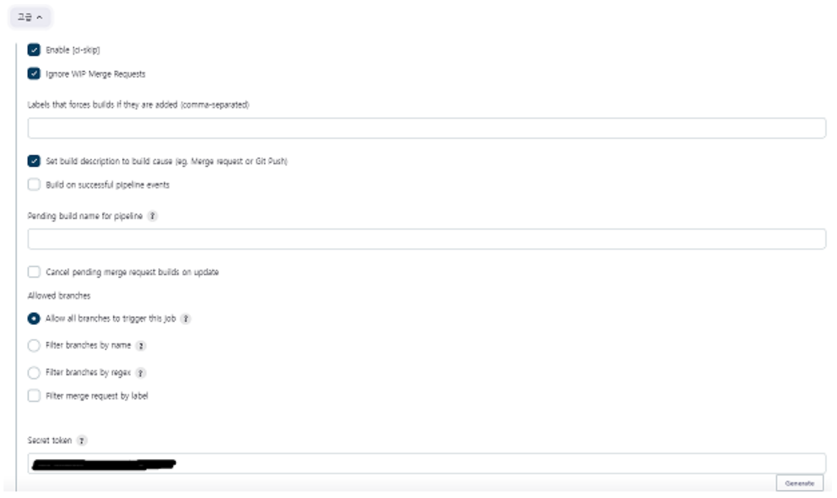
- [ ] Push Events
- [ ] Push Events in case of branch delete
- [ ] Opened Merge Request Events
- [ ] Build only if new commits were pushed to Merge Request ?
- [✓] Accepted Merge Request Events
- [ ] Closed Merge Request Events

Rebuild open Merge Requests: On push to source or target branch

- [✓] Approved Merge Requests (EE-only)
- [✓] Comments

Merge 이벤트에 트리거 이벤트 등록

- 고급 > Secret Token > Generate



고급 ^

- [✓] Enable [ci-skip]
- [✓] Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

- [✓] Set build description to build cause (eg. Merge request or Git Push)
- [ ] Build on successful pipeline events

Pending build name for pipeline ?

- [ ] Cancel pending merge request builds on update

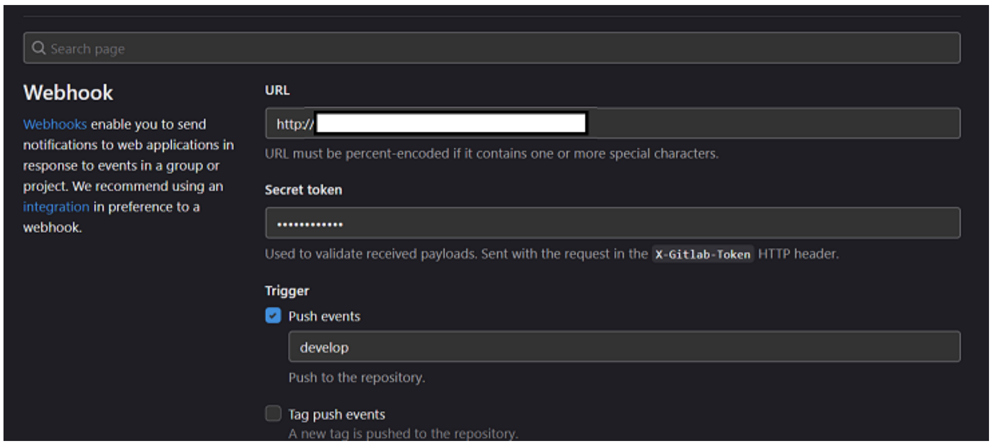
Allowed branches

- ● Allow all branches to trigger this job ?
- ○ Filter branches by name ?
- ○ Filter branches by regex ?
- [ ] Filter merge request by label

Secret token ?

[redacted]

GitLab 프로젝트 WebHook 설정



Webhook

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.

URL

http://[redacted]

URL must be percent-encoded if it contains one or more special characters.

Secret token

••••••••••••

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

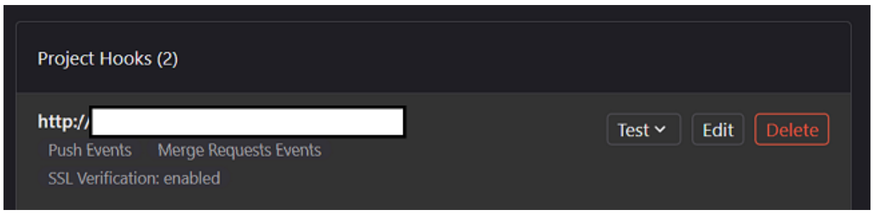
- [✓] Push events

  develop

  Push to the repository.
- [ ] Tag push events

  A new tag is pushed to the repository.

- 위에서 생성한 토큰 입력


Project Hooks (2)

http:// [redacted]

Push Events Merge Requests Events

SSL Verification: enabled

pipeline

```
pipeline {
    agent any

        tools {
           gradle 'Gradle7.6'
      }


    environment {
      GRANT_GRADLE = 'chmod +x ./gradlew'
      BUILD_COMMAND = './gradlew clean build -x test'
      MEMBER_PROJECT='member-service'
      CHALLENGE_PROJECT='challenge-service'
      SHADOWING_PROJECT='shadowing-service'
      CHATTING_PROJECT='chatting-service'
      EUREKA_PROJECT='eureka-service'
      GATEWAY_PROJECT='apigateway-service'
      CONFIG_PROJECT='config-service'
      FASTAPI='fastapi'
      FRONT='front'
      NGINX='nginx'
    }

    stages {
        stage('github clone') {
            steps {
                git branch: 'develop',
                    credentialsId: '토큰',
                    url: 'https://lab.ssafy.com/'
            }
        }

        stage('init config files') {
            steps {
                dir('front') {
                    sh 'cp /var/jenkins_home/initfile/front/.env.production ./.env.production'
                }
                dir('back/member-service/src/main/resources') {
                    sh 'cp /var/jenkins_home/initfile/member-service/bootstrap.yml ./bootstrap.yml'
                }
                dir('back/challenge-service/src/main/resources') {
                    sh 'cp /var/jenkins_home/initfile/challenge-service/bootstrap.yml ./bootstrap.yml'
                    sh 'cp /var/jenkins_home/initfile/challenge-service/serviceAccountKey.json ./serviceAccountKey.json'
                }
                dir('back/shadowing-service/src/main/resources') {
                    sh 'cp /var/jenkins_home/initfile/shadowing-service/bootstrap.yml ./bootstrap.yml'
                }
                dir('back/chatting-service/src/main/resources') {
                    sh 'cp /var/jenkins_home/initfile/chatting-service/bootstrap.yml ./bootstrap.yml'
                }
                dir('back/eureka-service/src/main/resources') {
                    sh 'cp /var/jenkins_home/initfile/eureka-service/application.yml ./application.yml'
                }
                dir('back/apigateway-service/src/main/resources') {
                    sh 'cp /var/jenkins_home/initfile/apigateway-service/bootstrap.yml ./bootstrap.yml'
                    sh 'cp /var/jenkins_home/initfile/apigateway-service/application.yml ./application.yml'
                }
                dir('back/config-service/src/main/resources') {
                    sh 'cp /var/jenkins_home/initfile/config-service/application.yml ./application.yml'
                }
            }
        }
        stage('Build') {
            parallel{
                stage('build-member-service'){
                    when {
                        changeset "back/member-service/**"
                    }
                    steps{
                        dir('back/member-service') {
                            sh 'pwd'
                            sh "$GRANT_GRADLE"
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-challenge-service'){
                    when {
                        changeset "back/challenge-service/**"
                    }
                    steps{
                        dir('back/challenge-service') {
                            sh "$GRANT_GRADLE"
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-shadowing-service'){
                    when {
                        changeset "back/shadowing-service/**"
                    }
                    steps{
                        dir('back/shadowing-service') {
                            sh "$GRANT_GRADLE"
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-chatting-service'){
                    when {
                        changeset "back/chatting-service/**"
                    }
                    steps{
                        dir('back/chatting-service') {
                            sh "$GRANT_GRADLE"
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-eureka-service'){
                    when {
                        changeset "back/eureka-service/**"
                    }
                    steps{
                        dir('back/eureka-service') {
                            sh "$GRANT_GRADLE"
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-apigateway-service'){
                    when {
                        changeset "back/apigateway-service/**"
                    }
                    steps{
                        dir('back/apigateway-service') {
                            sh "$GRANT_GRADLE"
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('config-service'){
                    when {
                        changeset "back/config-service/**"
                    }
                    steps{
                        dir('back/config-service') {
                            sh "$GRANT_GRADLE"
                            sh "$BUILD_COMMAND"
                        }
                    }
                }

            }
        }

        stage('Backup & Copy'){
            parallel{
                stage('backup-copy-member-service'){
                    when{
                        changeset "back/member-service/**"
                    }
                    steps{
                        dir('back/member-service') {
                            sh 'docker build -t ssafyc104/${MEMBER_PROJECT} .'
                            sh 'docker push ssafyc104/${MEMBER_PROJECT}'
                        }
                    }
                }
                stage('backup-copy-challenge-service'){
                    when{
                        changeset "back/challenge-service/**"
                    }
                    steps{
                        dir('back/challenge-service') {
                            sh 'docker build -t ssafyc104/${CHALLENGE_PROJECT} .'
                            sh 'docker push ssafyc104/${CHALLENGE_PROJECT}'
                        }
                    }
                }
                stage('backup-copy-shadowing-service'){
                    when{
                        changeset "back/shadowing-service/**"
                    }
                    steps{
                        dir('back/shadowing-service') {
                            sh 'docker build -t ssafyc104/${SHADOWING_PROJECT} .'
                            sh 'docker push ssafyc104/${SHADOWING_PROJECT}'
                        }
                    }
                }
                stage('backup-copy-chatting-service'){
                    when{
                        changeset "back/chatting-service/**"
                    }
                    steps{
                        dir('back/chatting-service') {
                            sh 'docker build -t ssafyc104/${CHATTING_PROJECT} .'
                            sh 'docker push ssafyc104/${CHATTING_PROJECT}'
                        }
                    }
                }
                stage('backup-copy-eureka-service'){
                    when{
                        changeset "back/eureka-service/**"
                    }
                    steps{
                        dir('back/eureka-service') {
                            sh 'docker build -t ssafyc104/${EUREKA_PROJECT} .'
                            sh 'docker push ssafyc104/${EUREKA_PROJECT}'
                        }
                    }
                }
                stage('backup-copy-apigateway-service'){
                    when{
                        changeset "back/apigateway-service/**"
                    }
                    steps{
                        dir('back/apigateway-service') {
                            sh 'docker build -t ssafyc104/${GATEWAY_PROJECT} .'
                            sh 'docker push ssafyc104/${GATEWAY_PROJECT}'
                        }
                    }
                }
                stage('backup-copy-config-service'){
                    when{
                        changeset "back/config-service/**"
                    }
                    steps{
                        dir('back/config-service') {
                            sh 'docker build -t ssafyc104/${CONFIG_PROJECT} .'
                            sh 'docker push ssafyc104/${CONFIG_PROJECT}'
                        }
                    }
                }
                stage('backup-copy-fastapi'){
                    when{
                        changeset "fastapi/**"
                    }
                    steps{
                        dir('fastapi') {
                            sh 'docker build -t ssafyc104/${FASTAPI} .'
                            sh 'docker push ssafyc104/${FASTAPI}'
                        }
                    }
                }
                stage('backup-copy-front'){
                    when{
                        changeset "front/**"
                    }
                    steps{
                        dir('front') {
                            sh 'docker build -t ssafyc104/${FRONT} .'
                            sh 'docker push ssafyc104/${FRONT}'
                        }
                    }
                }
                stage('backup-copy-nginx'){
                    when{
                        changeset "nginx/**"
                    }
                    steps{
                        dir('nginx') {
                            sh 'docker build -t ssafyc104/${NGINX} .'
                            sh 'docker push ssafyc104/${NGINX}'
                        }
                    }
                }
            }
        }

        stage('Deploy'){
            parallel{
                stage('deploy-member-service'){
                    when{
                        changeset "back/member-service/**"
                    }
                    steps{
                        sh 'docker stop ${MEMBER_PROJECT} || true && docker rm ${MEMBER_PROJECT} || true'
                        sh 'docker run -d --network opener-network --name ${MEMBER_PROJECT} -e "eureka.client.serviceUrl.defaultZone=h
                    }
                }
                stage('deploy-challenge-service'){
                    when{
                        changeset "back/challenge-service/**"
                    }
                    steps{
                        sh 'docker stop ${CHALLENGE_PROJECT} || true && docker rm ${CHALLENGE_PROJECT} || true'
                        sh 'docker run -d --network opener-network --name ${CHALLENGE_PROJECT} -e "eureka.client.serviceUrl.defaultZon
                    }
                }
                stage('deploy-shadowing-service'){
                    when{
                        changeset "back/shadowing-service/**"
                    }
                    steps{
                        sh 'docker stop ${SHADOWING_PROJECT} || true && docker rm ${SHADOWING_PROJECT} || true'
                        sh 'docker run -d --network opener-network --name ${SHADOWING_PROJECT} -e "eureka.client.serviceUrl.defaultZon
                    }
                }
                stage('deploy-chatting-service'){
                    when{
                        changeset "back/chatting-service/**"
                    }
                    steps{
                        sh 'docker stop ${CHATTING_PROJECT} || true && docker rm ${CHATTING_PROJECT} || true'
                        sh 'docker run -d --network opener-network --name ${CHATTING_PROJECT} -e "eureka.client.serviceUrl.defaultZone
                    }
                }
                stage('deploy-eureka-service'){
                    when{
                        changeset "back/eureka-service/**"
                    }
                    steps{
                        sh 'docker stop ${EUREKA_PROJECT} || true && docker rm ${EUREKA_PROJECT} || true'
                        sh 'docker run -d -p 8761:8761 --network opener-network --name ${EUREKA_PROJECT} ssafyc104/${EUREKA_PROJECT}'
                    }
                }
                stage('deploy-apigateway-service'){
                    when{
                        changeset "back/apigateway-service/**"
                    }
                    steps{
                        sh 'docker stop ${GATEWAY_PROJECT} || true && docker rm ${GATEWAY_PROJECT} || true'
                        sh 'docker run -d -p 포트번호:포트번호 --network opener-network --name ${GATEWAY_PROJECT} -e "eureka.client.servic
                    }
                }
                stage('deploy-config-service'){
                    when{
                        changeset "back/config-service/**"
                    }
                    steps{
```

```
                    sh 'docker stop ${CONFIG_PROJECT} || true && docker rm ${CONFIG_PROJECT} || true'
                    sh 'docker run -d -p 포트번호:포트번호 --network opener-network --name ${CONFIG_PROJECT} -e "eureka.client.service
                }
            }
            stage('deploy-fastapi'){
                when{
                    changeset "fastapi/**"
                }
                steps{
                    sh 'docker stop ${FASTAPI} || true && docker rm ${FASTAPI} || true'
                    sh 'docker run -d -p 포트번호:포트번호 --network opener-network --name ${FASTAPI} ssafyc104/${FASTAPI}'
                }
            }
            stage('deploy-front'){
                when{
                    changeset "front/**"
                }
                steps{
                    sh 'docker stop ${FRONT} || true && docker rm ${FRONT} || true'
                    sh 'docker run -d --name ${FRONT} -p 포트번호:포트번호 --network opener-network ssafyc104/${FRONT}'
                }
            }
            stage('deploy-nginx'){
                when{
                    changeset "nginx/**"
                }
                steps{
                    sh 'docker stop ${NGINX} || true && docker rm ${NGINX} || true'
                    sh 'docker run -d --name ${NGINX} -v /etc/letsencrypt/:/etc/letsencrypt/ -p 80:80 -p 443:443 --network opener-
                }
            }
        }
    }
    stage('End') {
        steps {
            mattermostSend color: '#32a852', message: "Open'ur Deploy End! (${env.JOB_NAME}) #(${env.BUILD_NUMBER}) (<${env.BUILD_
        }
    }
  }
}
```