포팅 매뉴얼 - 한솥밥

프로젝트 사용도구

```
이슈관리 : Jira
형상관리 : Gitlab
디자인 : Figma
서버관리 : Termius
CI/CD : Jenkins
개발 툴 : Visual Studio Code,  Android Studio, IntelliJ
커뮤니케이션 : Notion, MatterMost
```

버전 정보

- Frontend

```
VSCode version : 1.86
Android Studio : 1.1.28
Flutter : 3.86.0-stable
Dart : 3.86.0
```

- Backend

```
Intellij : 2023.3.4
Java : 17
Spring Boot : 3.2.3
Gradle : 8.5
Python : 3.11.8
Torch : 2.2.1+cu121
Fastapi : 0.110.0
Hadoop : 3.3.6
Spark : 3.5.1
MySql : 8.0.35
Redis : 7.2.4
```

- Service

```
Ubuntu : 20.04
NginX : 1.18.0
Docker : 25.0.4
Jenkins : 2.448
Sonarqube : 4.4.1.3373
```

외부 서비스

```
- Kakao SDK for Flutter ⇒ .env 파일에 정의
- Google Maps Platform ⇒ .env 파일에 정의
- Firebase FCM ⇒ firebase_service_key.json 파일에 정의
- AWS S3 ⇒ application.properties 파일에 정의
```

환경 변수

Frontend

- .env

```
MAP_KEY=map_key
NATIVE_KEY=native_key
JSAPP_KEY=jsapp_key
GOOGLE_KEY=google_key
```

Backend

- application.properties

```
# Host
host.server.base-url=http://j10b209.p.ssafy.io:8081
host.server.domain=j10b209.p.ssafy.io:8081
#host.server.name=j10b209.p.ssafy.io
host.server.name=j10b209.p.ssafy.io

#port
server.port=8081

# Spring MVC Configuration
spring.mvc.pathmatch.matching-strategy=ant_path_matcher

# Database (MySQL) Configurations
spring.jpa.database=mysql
spring.jpa.hibernate.ddl-auto=update
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jackson.serialization.fail-on-empty-beans=false

# Database Access Information
spring.datasource.url=jdbc:mysql://${host.server.name}:3306/hansotbab?charact
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=

# Springdoc Swagger Configuration
springdoc.api-docs.path=/api-docsspringdoc.swagger-ui.disable-swagger-default
#springdoc.swagger-ui.oauth2-redirect-url=/api/login/oauth2/code/{registratio
springdoc.packages-to-scan=com.b209.hansotbab

# Kakao OAuth2
spring.security.oauth2.client.provider.kakao.authorization-uri=
spring.security.oauth2.client.provider.kakao.token-uri=
spring.security.oauth2.client.provider.kakao.user-name-attribute=
spring.security.oauth2.client.provider.kakao.user-info-authentication-method=
spring.security.oauth2.client.registration.kakao.client-id=
spring.security.oauth2.client.registration.kakao.client-secret=
spring.security.oauth2.client.registration.kakao.client-authentication-method
spring.security.oauth2.client.registration.kakao.redirect-uri=
spring.security.oauth2.client.registration.kakao.authorization-grant-type=
spring.security.oauth2.client.registration.kakao.client-name=
spring.security.oauth2.client.registration.kakao.scope=

# JWT
```

```
jwt.secret.key=Si13pG5Y9mx0wjS12GCh0x17dl28Q4zsH3f289er8s9wLm3Vr93Rp0q
#jwt.secret=dkfj2987dkfbknera9s8d7f01923mvlxcbjswidufopssdkj9872384gh2k4bj34n
# Https
#server.ssl.enabled=true
#server.ssl.key-store=key.pem
spring.data.redis.host=${host.server.name}
spring.data.redis.port=6379
spring.data.redis.password=

spring.elasticsearch.uris=j10b209.p.ssafy.io:9200

cloud.aws.s3.bucket:ssafyjoblog
cloud.aws.stack.auto: false
cloud.aws.region.static: ap-northeast-2
cloud.aws.credentials.accessKey:
cloud.aws.credentials.secretKey:

spring.servlet.multipart.maxFileSize=50MB
spring.servlet.multipart.maxRequestSize=50MB
```

- firebase_service_key.json

```
{
  "type": "service_account",
  "project_id": ,
  "private_key_id": ,
  "private_key": ,
  "client_email": "firebase-adminsdk-bepdp@hansotbab-alarm.iam.gserviceaccoun
  "client_id": ,
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs"
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/
  "universe_domain": "googleapis.com"
}
```

EC2 서버 세팅

포트 설정

```
# 포트확인
#---------------------------------- netstat --------------------------------
# net-tools 다운로드
$ sudo apt install net-tools

# 모든 포트 확인
```

```
$ netstat -tnlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN

# 열려있는 특정 포트 번호만 확인
$ netstat -tnlp : grep 포트번호

#-------------------------------------- ufw ----------------------------------
ubuntu@ip-:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
8989                       ALLOW       Anywhere
443                        ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
8989 (v6)                  ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)


# 방화벽 허용
# jenkins
$ sudo ufw allow 8080
# mysql
$ sudo ufw allow 3306
# springboot(back)
$ sudo ufw allow 8081
# redis
$ sudo ufw allow 6379

# 방화벽 리로드
$ sudo ufw reload
Firewall reloaded

# 포트 상태 확인
$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
8989                       ALLOW       Anywhere
443                        ALLOW       Anywhere
3306                       ALLOW       Anywhere
```

```
Nginx Full                 ALLOW       Anywhere
9200                       ALLOW       Anywhere
5044                       ALLOW       Anywhere
5601                       ALLOW       Anywhere
6379                       DENY        Anywhere
80                         ALLOW       Anywhere
8080                       ALLOW       Anywhere
8081                       ALLOW       Anywhere
9000                       ALLOW       Anywhere
Anywhere                   DENY        14.50.47.222
22 (v6)                    ALLOW       Anywhere (v6)
8989 (v6)                  ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
3306 (v6)                  ALLOW       Anywhere (v6)
Nginx Full (v6)            ALLOW       Anywhere (v6)
9200 (v6)                  ALLOW       Anywhere (v6)
5044 (v6)                  ALLOW       Anywhere (v6)
5601 (v6)                  ALLOW       Anywhere (v6)
6379 (v6)                  DENY        Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
8080 (v6)                  ALLOW       Anywhere (v6)
8081 (v6)                  ALLOW       Anywhere (v6)
9000 (v6)                  ALLOW       Anywhere (v6)
```

필요 패키지 설치

```
# java 17 jdk 설치
$ sudo apt install openjdk-17-jdk

# 자바 버전 확인
$ java -version
openjdk 17.0.10 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-120.04.1)
OpenJDK 64-Bit Server VM (build 17.0.10+7-Ubuntu-120.04.1, mixed mode, sharin

....
```

디렉터리 구조

- front

  ```
  ## Front
  ```

```
📦flutter/hansotbab
 ┣ 📂android
 ┣ 📂assets
 ┣ 📂build
 ┣ 📂lib
 ┃ ┣ 📂class
 ┃ ┣ 📂firebase
 ┃ ┃ ┣ 📜fcmSetting.dart
 ┃ ┃ ┗ 📜firebase_options.dart
 ┃ ┣ 📂pages
 ┃ ┣ 📂providers
 ┃ ┃ ┣ 📜api.dart
 ┃ ┃ ┣ 📜loginState.dart
 ┃ ┃ ┗ 📜userpreferences.dart
 ┃ ┣ 📂widget
 ┃ ┃ ┣ 📜bottomappbar.dart
 ┃ ┃ ┣ 📜button.dart
 ┃ ┃ ┣ 📜carousel.dart
 ┃ ┃ ┣ 📜imagepicker.dart
 ┃ ┃ ┣ 📜nearst.dart
 ┃ ┃ ┗ 📜navermap.dart
 ┣ 📜.gitignore
 ┣ 📜analysis_options.yaml
 ┣ 📜devtools_options.yaml
 ┣ 📜flutter_jank_metrics_01.json
 ┣ 📜flutter_native_splash.yaml
 ┣ 📜pubspec.yaml
 ┗ 📜README.md
```

- back

```
## Back

📦server/hansotbab
 ┣ 📂.gradle
 ┣ 📂.idea
 ┣ 📂gradle
 ┣ 📂src
 ┃ ┣ 📂main
 ┃ ┃ ┣ 📂java
 ┃ ┃ ┃ ┗ 📂com
 ┃ ┃ ┃ ┃ ┗ 📂b209
 ┃ ┃ ┃ ┃ ┃ ┗ 📂hansotbab
 ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂alarm
 ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂config
 ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜FCMConfig.java
 ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂controller
 ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜AlarmController.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂dto
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂request
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜AlarmRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜AlarmUserRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂entity
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜FcmToken.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂repository
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜FcmTokenRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂service
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜AlarmService.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜AlarmServiceImpl.java
┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂food
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂controller
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜FoodController.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂dto
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜FoodRequestDto.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂entity
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ElasticSearchItems.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂repository
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ElasticSearchItemsRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂service
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜FoodService.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜JsonDataUtils.java
┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂fridge
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂controller
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜FridgeController.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂dto
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂request
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ProductCreateRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ProductUpdateRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂response
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜FridgeListResponseDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ProductDetailResponseDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ProductListResponseDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂entity
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜Fridge.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜FridgeLike.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜Product.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ProductBring.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ProductCategory.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ProductCondition.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂repository
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜FridgeLikeRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜FridgeRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ProductBringRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ProductRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂service
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜FridgeService.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ProductService.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜S3Service.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜WebClientService.java
┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂global
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂advice
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜AuthControllerAdvice.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂config
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜AuditConfig.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ElasticSearchConfig.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜S3Config.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜SwaggerConfig.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂entity
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜BaseEntity.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜BaseTimeEntity.java
┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂review
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂controller
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ReviewController.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂dto
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂request
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ReviewLikeRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ReviewRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂response
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ReviewResponseDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂entity
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜Review.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ReviewLike.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂repository
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ReviewRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂service
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜ReviewService.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜ReviewServiceImpl.java
┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂user
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂config
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜RedisConfig.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜RestTemplateConfig.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜SecurityConfig.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜WebConfig.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂controller
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜UserController.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂dto
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂request
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜AdminReqestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜LoginRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜NicknameDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜RegisterRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜TokenReissueRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂response
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜LikedFridgeDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜TokenDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜UserProfileDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜UserResponseDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂entity
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜RefreshToken.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜RoleType.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜User.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜UserPrincipal.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂exception
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜AlreadyAuthenticatedException.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜InvalidTokenException.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜NotAuthenticactedException.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜NoUserExistsException.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜RegisterFailedException.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜UnauthorizedException.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂handler
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜CustomAccessDeniedHandler.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜CustomAuthenticationEntryPoint.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜CustomLogoutHandler.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜CustomLogoutSuccessHandler.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂jwt
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜JWTAuthenticationFilter.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜LoginAuthenticationToken.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜TokenProvider.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂repository
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜TokenRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜UserRepository.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂service
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜RefreshTokenService.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜UserPrincipalService.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜UserService.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂util
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜AuthUtils.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜CookieUtils.java
┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂wishlist
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂controller
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜WishlistController.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂dto
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂request
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜WishlistLikeRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜WishlistRequestDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📂response
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜WishlistResponseDTO.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂entity
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜Wishlist.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┗ 📜WishlistLike.java
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📂repository
┃ ┃ ┃ ┃ ┃ ┃ ┃ ┃ ┣ 📜WishlistLikeRepository.java
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜WishlistRepository.java
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📂service
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┃  ┣ 📜WishlistService.java
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜WishlistServiceImpl.java
┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜HansotbabApplication.java
┃  ┃  ┗ 📂resources
┃  ┃  ┃  ┣ 📂files
┃  ┃  ┃  ┃  ┣ 📜food_info.json
┃  ┃  ┃  ┃  ┗ 📜food_result.json
┃  ┃  ┃  ┣ 📂firebase
┃  ┃  ┃  ┃  ┣ 📜firebase.json
┃  ┃  ┃  ┃  ┗ 📜firebase_service_key.json
┃  ┃  ┃  ┣ 📂static
┃  ┃  ┃  ┃  ┗ 📜index.html
┃  ┃  ┃  ┗ 📜log4j2.xml
┃  ┗ 📂test
┃  ┃  ┗ 📂java
┃  ┃  ┃  ┗ 📂com
┃  ┃  ┃  ┃  ┗ 📂b209
┃  ┃  ┃  ┃  ┃  ┗ 📂hansotbab
┃  ┃  ┃  ┃  ┃  ┃  ┣ 📂fridge
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┣ 📂repository
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜FridgeRepositoryTest.java
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📂service
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜WebClientServiceTest.java
┃  ┃  ┃  ┃  ┃  ┃  ┣ 📂user
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📂repository
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜UserRepositoryTest.java
┃  ┃  ┃  ┃  ┃  ┃  ┣ 📂wishlist
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┣ 📂repository
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜WishlistRepositoryTest.java
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📂service
┃  ┃  ┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜WishlistServiceTest.java
┃  ┃  ┃  ┃  ┃  ┃  ┗ 📜HansotbabApplicationTests.java
┣ 📜.gitignore
┣ 📜build.gradle
┣ 📜compose.yaml
┣ 📜gradlew
┣ 📜gradlew.bat
┗ 📜settings.gradle
```

- Server

```
/home/ubuntu
📦 /home/ubuntu
 ┣ 📂deploy
 ┃  ┣ 📂back
 ┃  ┃  ┣ 📜Dockerfile
```

```
|   |   ├ 📜docker-compose.yaml
|   |   └ 📜hansotbab-0.0.1-SNAPSHOT.jar
|   └ 📜deploy.sh
├ 📂env
|   ├ 📜application.properties
|   └ 📜firebase_service_key.json
└ 📂workspace
    └ 📂docker
        ├ 📜docker-compose.yaml
        └ 📂sonarqube
            └ 📜docker-compose.yaml
```

서비스 이용을 위한 빌드 및 배포 (CI/CD)

Dockerfile

- back

```
FROM openjdk:17-jdk
LABEL maintainer="yeong"
ARG JAR_FILE=hansotbab-0.0.1-SNAPSHOT.jar
ADD ${JAR_FILE} docker-springboot.jar
ENTRYPOINT ["java","-jar","docker-springboot.jar"]
```

docker-compose

- back

```
version: '3'

services:
  app:
    container_name: b209-back
    image: docker-springboot:0.1
    ports:
       - 8081:8081
    build:
      context: . # Dockerfile locate
      dockerfile: Dockerfile
```

- jenkins, mysql, redis

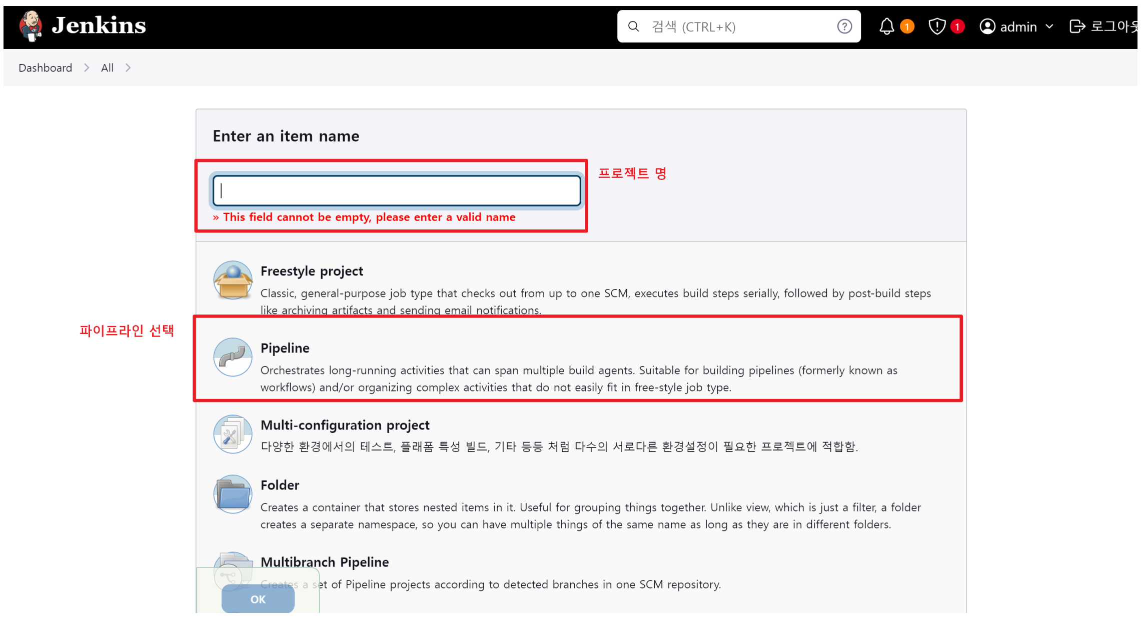
```
version: '3'

services:
  jenkins:
    container_name: jenkins
    image: jenkins/jenkins:latest
    user: root
    ports:
      - "${JENKINS_BINDING_PORT}:${JENKINS_PORT}"
    environment:
      - JENKINS_OPTS=--prefix=/jenkins
    restart: on-failure
    volumes:
      - ${JEKINS_DATA_PATH}:/var/jenkins_home
  db:
    image: mysql:8.0.35
    container_name: mysql-server
    ports:
      - "${MYSQL_BINDING_PORT}:${MYSQL_PORT}"
    environment:
        MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
        MYSQL_DATABASE: ${MYSQL_DATABASE}
        MYSQL_USER: ${MYSQL_USER}
        MYSQL_PASSWORD: ${MYSQL_PASSWORD}
        TZ: Asia/Seoul
    command: # 명령어 실행
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci

  redis:
    # 사용할 이미지
    image: redis:latest
    # 컨테이너명
    container_name: redis
    # 접근 포트 설정(컨테이너 외부:컨테이너 내부)
    ports:
      - "${REDIS_BINDING_PORT}:${REDIS_PORT}"
    # 스토리지 마운트(볼륨) 설정
    volumes:
      - ${REDIS_DATA_PATH}:/data
      - ${REDIS_DEFAULT_CONFIG_FILE}:/usr/local/conf/redis.conf
    # 컨테이너 종료시 재시작 여부 설정
    restart: always
    # command: redis-server /usr/local/conf/redis.conf
    command: redis-server --requirepass ${REDIS_PASSWORD} --port 6379
```

- sonarqube

```
version: "3"
services:
  sonarqube:
    image: sonarqube:community
    hostname: sonarqube
    container_name: sonarqube
    depends_on:
      - qube-db
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://qube-db:5432/sonar
      SONAR_JDBC_USERNAME: ${SONAR_USERNAME}
      SONAR_JDBC_PASSWORD: ${SONAR_PASSWORD}
    ports:
      - "${SONAR_BINDING_PORT}:${SONAR_PORT}"
  qube-db:
    image: postgres:13
    hostname: postgresql
    container_name: postgresql
    ports:
      - "${POSTGRES_BINDING_PORT}:${POSTGRES_PORT}"
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
```

Jenkins 초기 설정 및 Webhook 연결 (CI)

1. item 생성



Enter an item name

» This field cannot be empty, please enter a valid name

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

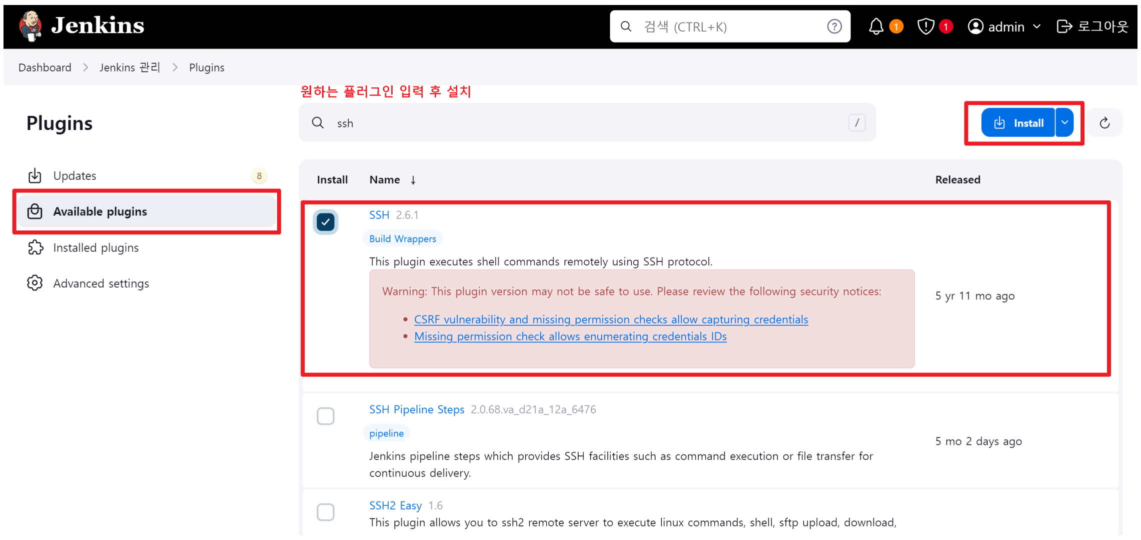
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

2. ssh 플러그인 설치



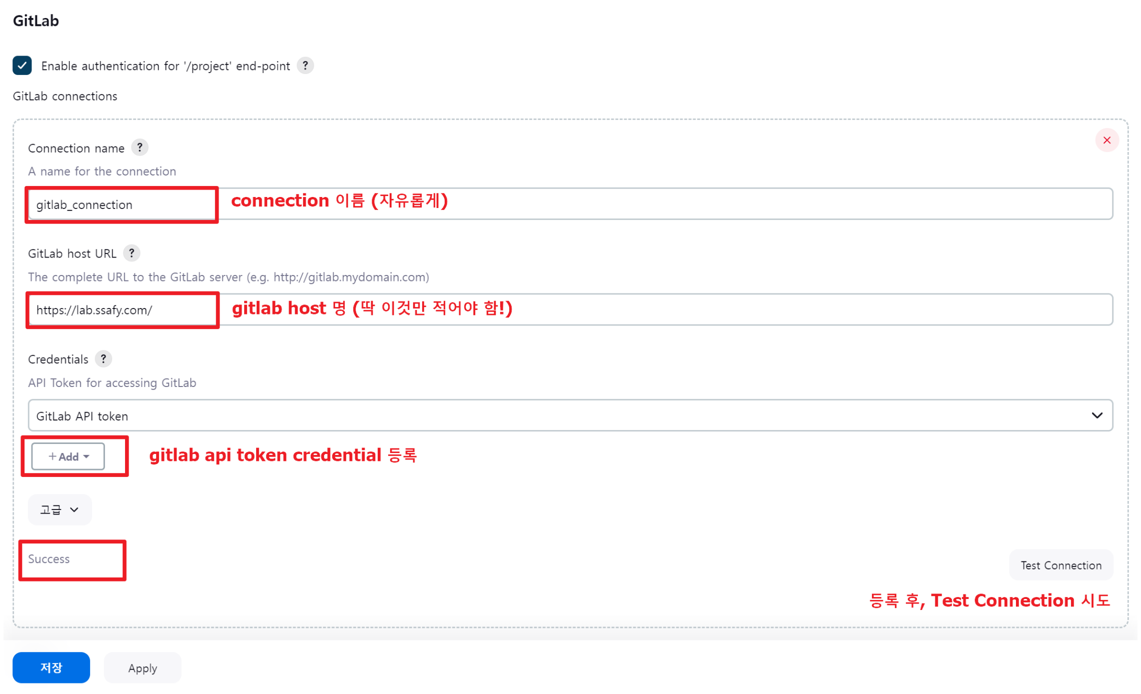
원하는 플러그인 입력 후 설치

ssh

Install	Name ↓	Released
✓	SSH 2.6.1 Build Wrappers This plugin executes shell commands remotely using SSH protocol. Warning: This plugin version may not be safe to use. Please review the following security notices: • CSRF vulnerability and missing permission checks allow capturing credentials • Missing permission check allows enumerating credentials IDs	5 yr 11 mo ago
	SSH Pipeline Steps 2.0.68.va_d21a_12a_6476 pipeline Jenkins pipeline steps which provides SSH facilities such as command execution or file transfer for continuous delivery.	5 mo 2 days ago
	SSH2 Easy 1.6 This plugin allows you to ssh2 remote server to execute linux commands, shell, sftp upload, download,	

```
SSH Agent
Docker
Docker Commons
Docker Pipeline
Docker API
Generic Webhook Trigger
GitLab
GitLab API
GitLab Authentication
```

3. gitlab API Token 및 credential 등록


GitLab

✓ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?
A name for the connection
gitlab_connection
connection 이름 (자유롭게)

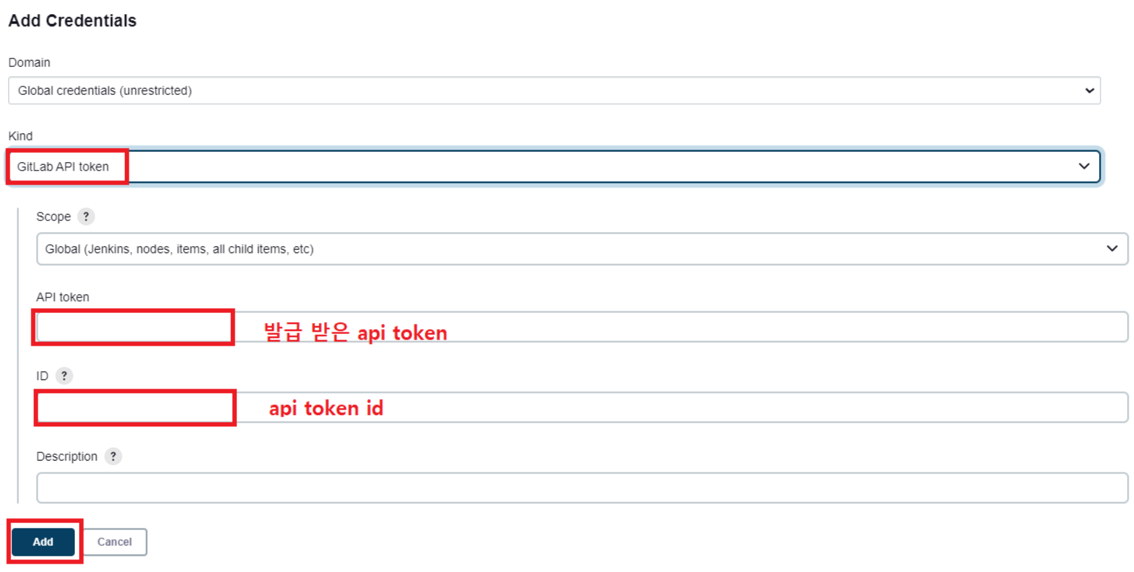
GitLab host URL ?
The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)
https://lab.ssafy.com/
gitlab host 명 (딱 이것만 적어야 함!)

Credentials ?
API Token for accessing GitLab
GitLab API token

gitlab api token credential 등록

Success

등록 후, Test Connection 시도


Add Credentials

Domain
Global credentials (unrestricted)

Kind
GitLab API token

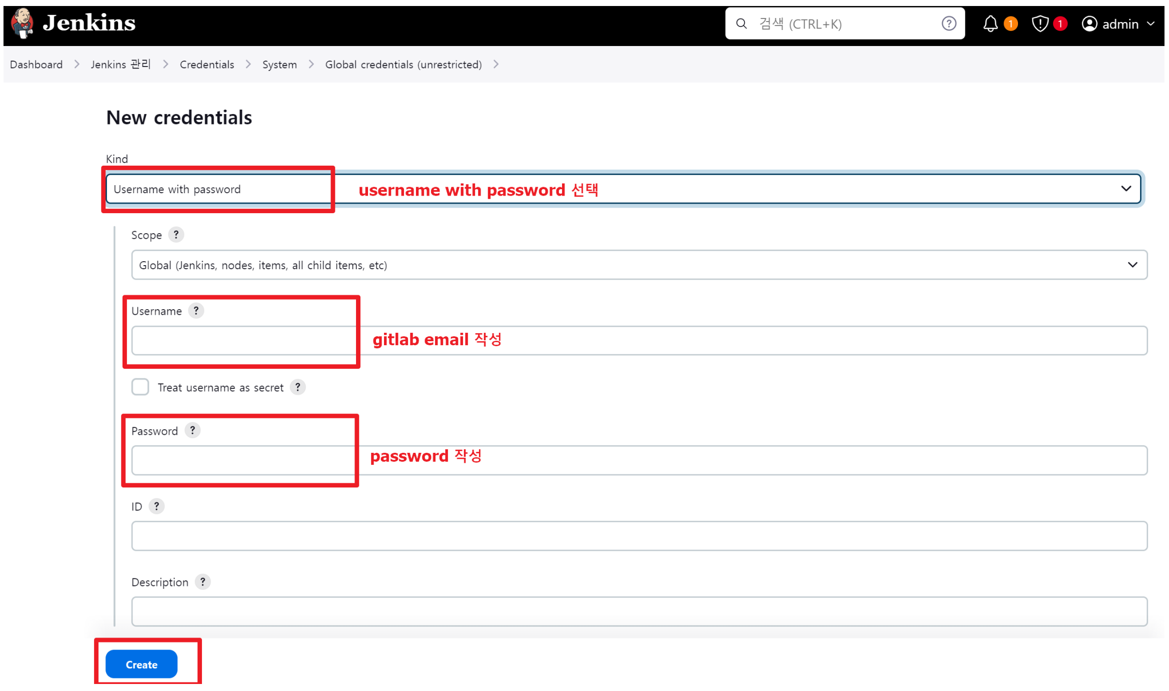
Scope
Global (Jenkins, nodes, items, all child items, etc)

API token
발급 받은 api token

ID
api token id

Description

gitlab user 등록



Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind
Username with password
username with password 선택

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

Username
gitlab email 작성

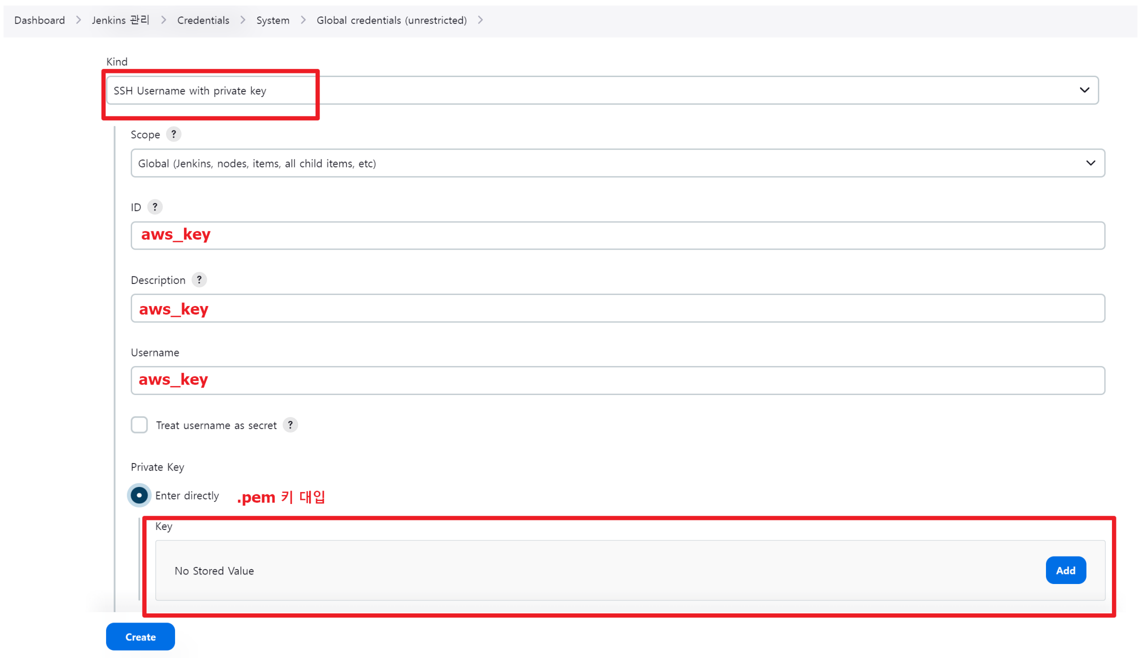
Treat username as secret

Password
password 작성

ID

Description

ssh 등록


Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >

Kind: SSH Username with private key

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

ID: aws_key

Description: aws_key

Username: aws_key

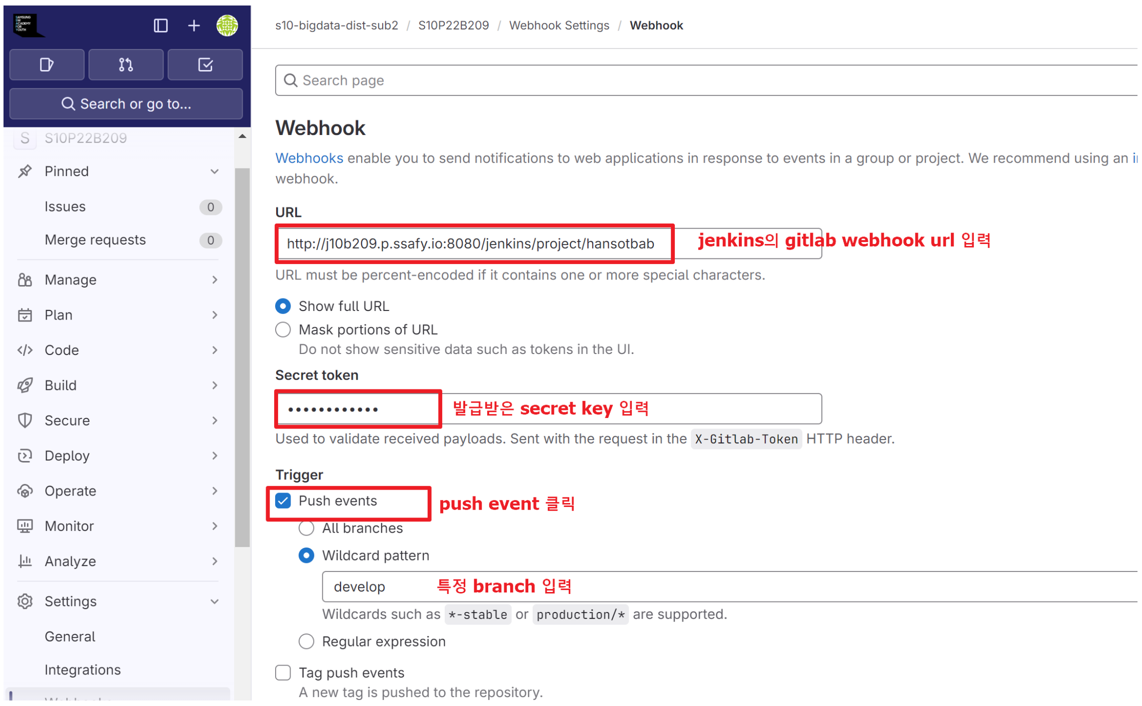
Treat username as secret

Private Key

Enter directly .pem 키 대입

Key: No Stored Value

4. gitlab webhook 등록



s10-bigdata-dist-sub2 / S10P22B209 / Webhook Settings / Webhook

Webhook

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

URL: http://j10b209.p.ssafy.io:8080/jenkins/project/hansotbab jenkins의 gitlab webhook url 입력

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

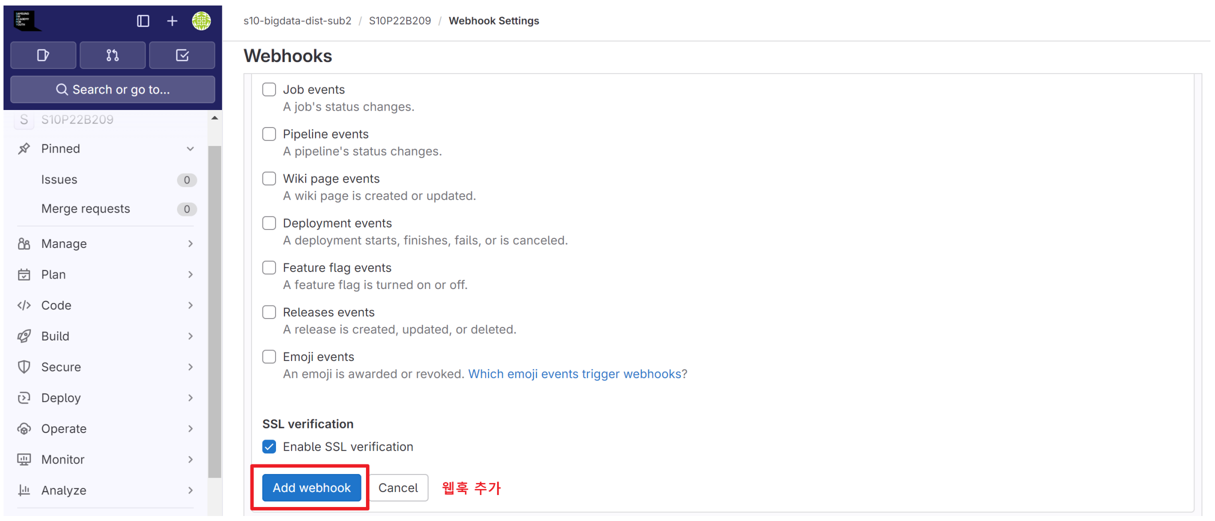
- Show full URL [✓]
- Mask portions of URL
  Do not show sensitive data such as tokens in the UI.

Secret token: •••••••••••• 발급받은 secret key 입력

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

Trigger

- [✓] Push events push event 클릭
  - All branches
  - Wildcard pattern [✓]: develop 특정 branch 입력
    Wildcards such as `*-stable` or `production/*` are supported.
  - Regular expression
- [ ] Tag push events
  A new tag is pushed to the repository.

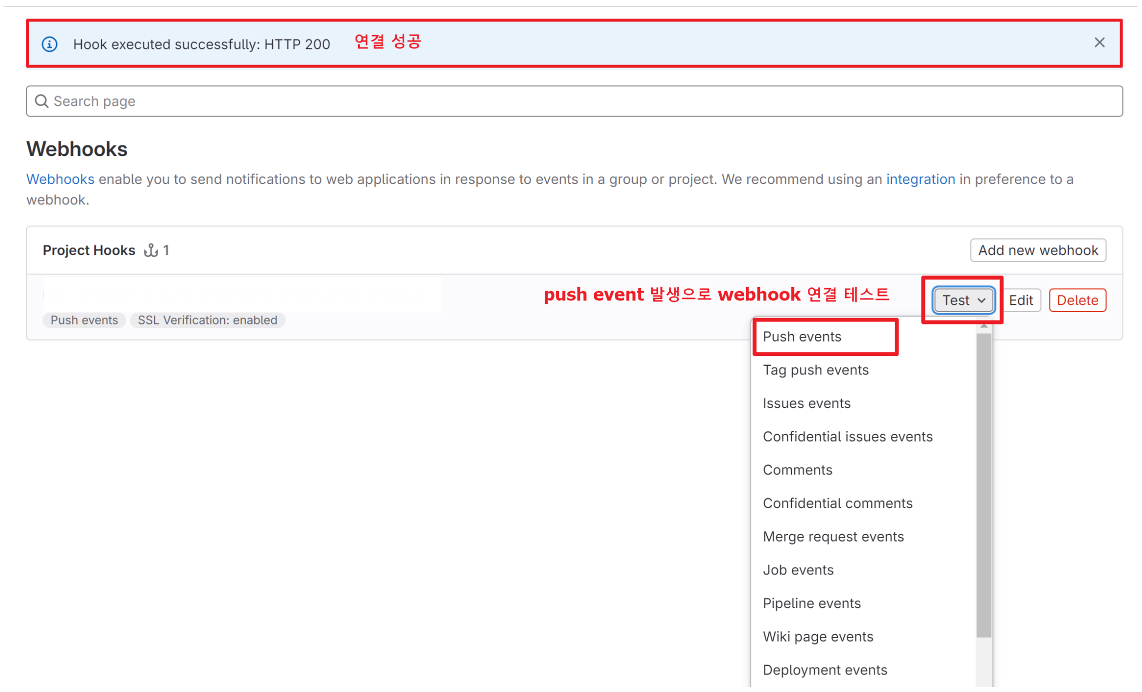


s10-bigdata-dist-sub2 / S10P22B209 / Webhook Settings

Webhooks

- [ ] Job events
  A job's status changes.
- [ ] Pipeline events
  A pipeline's status changes.
- [ ] Wiki page events
  A wiki page is created or updated.
- [ ] Deployment events
  A deployment starts, finishes, fails, or is canceled.
- [ ] Feature flag events
  A feature flag is turned on or off.
- [ ] Releases events
  A release is created, updated, or deleted.
- [ ] Emoji events
  An emoji is awarded or revoked. Which emoji events trigger webhooks?

SSL verification
- [x] Enable SSL verification



Hook executed successfully: HTTP 200 연결 성공

Search page

Webhooks

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.

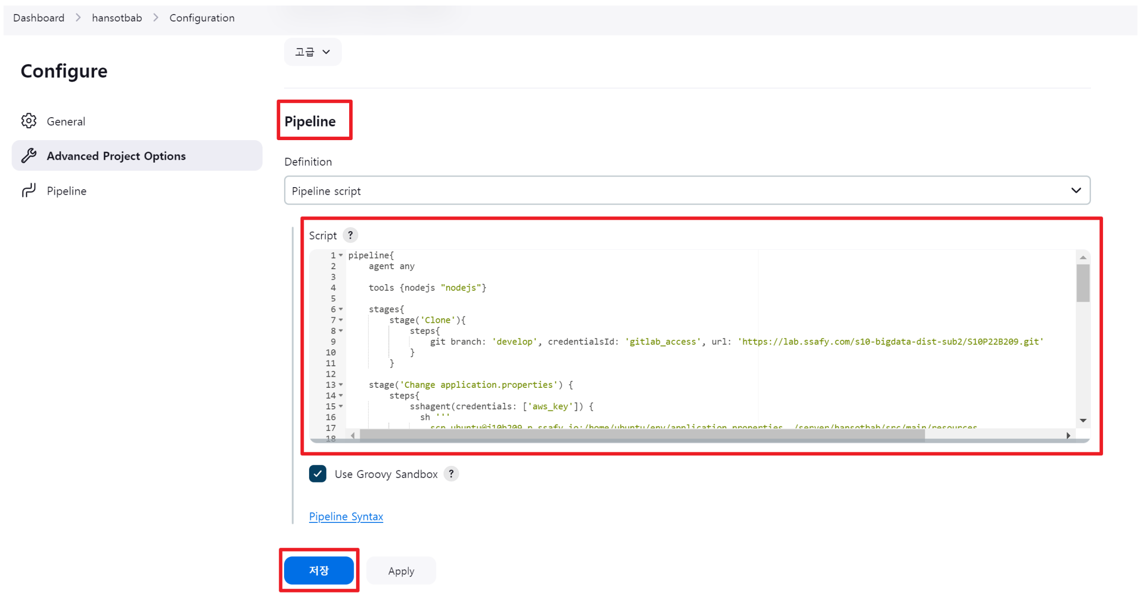
Project Hooks 1

[redacted]

Push events SSL Verification: enabled

push event 발생으로 webhook 연결 테스트

Jenkins 파이프라인 작성 및 배포 (CD)



```
pipeline{
    agent any

    tools {nodejs "nodejs"}

    stages{
        stage('Clone'){
            steps{
                git branch: 'develop', credentialsId: 'gitlab_access', url: 'https://lab.ssafy.com/s10-bigdata-dist-sub2/S10P22B209.git'
            }
        }

    stage('Change application.properties') {
        steps{
            sshagent(credentials: ['aws_key']) {
              sh '''
                scp ubuntu@j10b209.p.ssafy.io:/home/ubuntu/env/application.properties ./server/hansotbab/src/main/resources
```

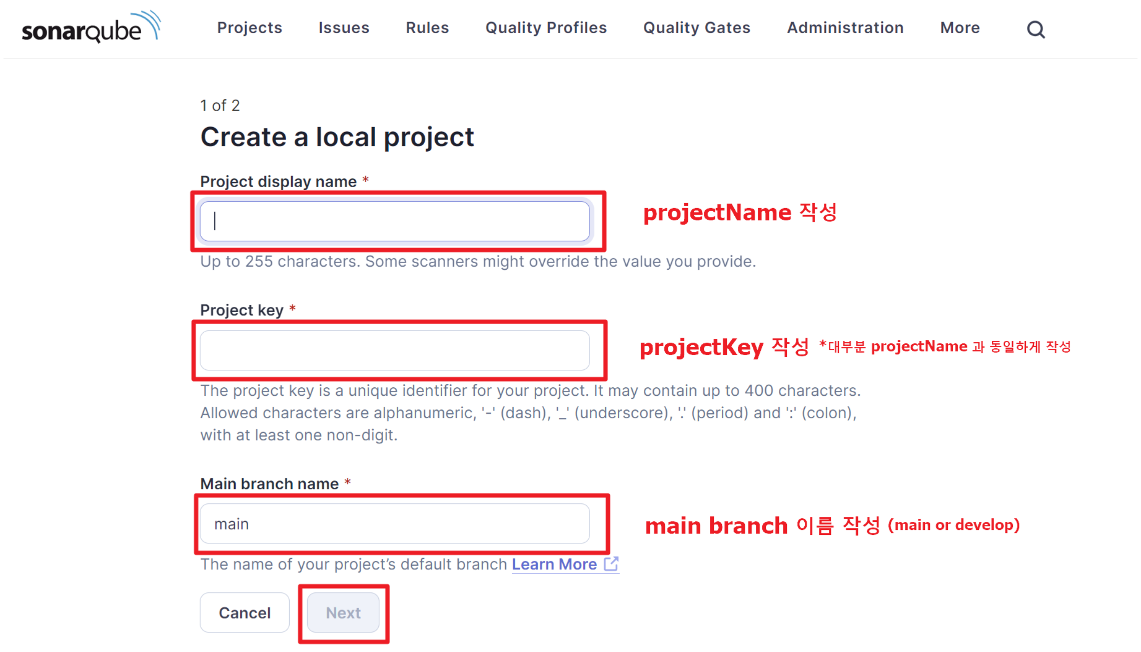
```
pipeline{
    agent any

    stages{
        stage('Clone'){
            steps{
                git branch: 'develop', credentialsId: 'gitlab_access', url: '
            }
        }

        stage('Change application.properties') {
            steps{
                sshagent(credentials: ['aws_key']) {
                    sh '''
                    ls -al
                    scp ubuntu@j10b209.p.ssafy.io:/home/ubuntu/env/applicatio
                    scp ubuntu@j10b209.p.ssafy.io:/home/ubuntu/env/firebase_s

                    '''
                }
            }
        }

       /*
        stage('Gradle Junit Test') {
            steps{
                dir('server') {
                    dir('hansotbab') {
                        sh '''
                        ls -al
                        chmod +x ./gradlew
                        ./gradlew test
                    '''
                }

            }
        }
    }
    */


    stage('Gradle Build'){
        steps{
            dir('server') {
                dir('hansotbab') {
                    sh '''
                        ls -al
                        chmod +x ./gradlew
                        ./gradlew build --exclude-task test
                    '''
                }

            }
        }
    }

    stage('SonarQube') {
        steps {
            withSonarQubeEnv(credentialsId:"SONAR_TOKEN",installationName
                dir('server'){
                    dir('hansotbab'){
                        sh '''
                            ls -al
                            chmod +x ./gradlew
                            ./gradlew sonar
                        '''

                    }
                }
            }
        }
    }


   // stage('Publish test results') {
   //     steps{
//             junit '**/build/test-results/test/*.xml'
```

```
        //      }
    //      }



        stage('Deploy') {
            steps {
                sshagent(credentials: ['aws_key']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@j10b209.p.ssaf
                        scp -C /var/jenkins_home/workspace/hansotbab/server/h
                        ssh -o StrictHostKeyChecking=no ubuntu@j10b209.p.ssaf
                        ssh -o StrictHostKeyChecking=no ubuntu@j10b209.p.ssaf
                    '''
                }
            }
        }

    }
}
```

Sonarqube 설정

1. sonarqube 초기 세팅

   manually or local 프로젝트 생성



1 of 2

Create a local project

Project display name *

Up to 255 characters. Some scanners might override the value you provide.

Project key *

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

main

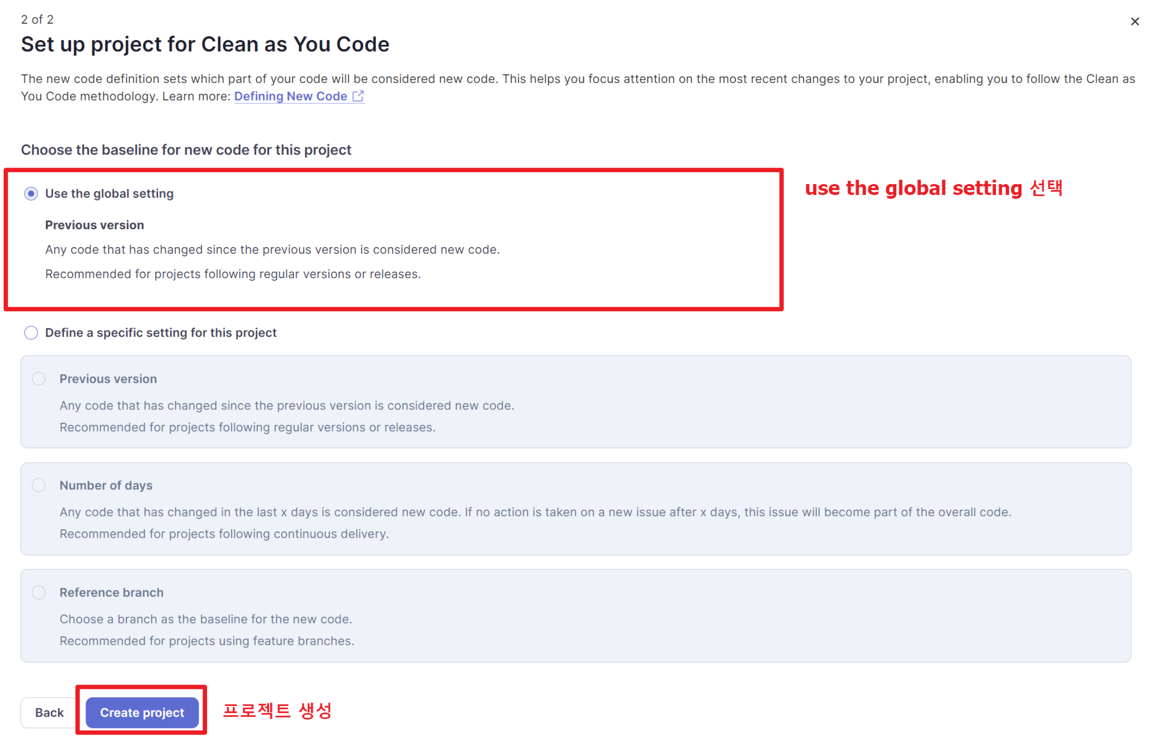
The name of your project's default branch Learn More

Cancel Next

projectName 작성

projectKey 작성 *대부분 projectName 과 동일하게 작성

main branch 이름 작성 (main or develop)



2 of 2

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: Defining New Code

Choose the baseline for new code for this project

(●) Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

( ) Define a specific setting for this project

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

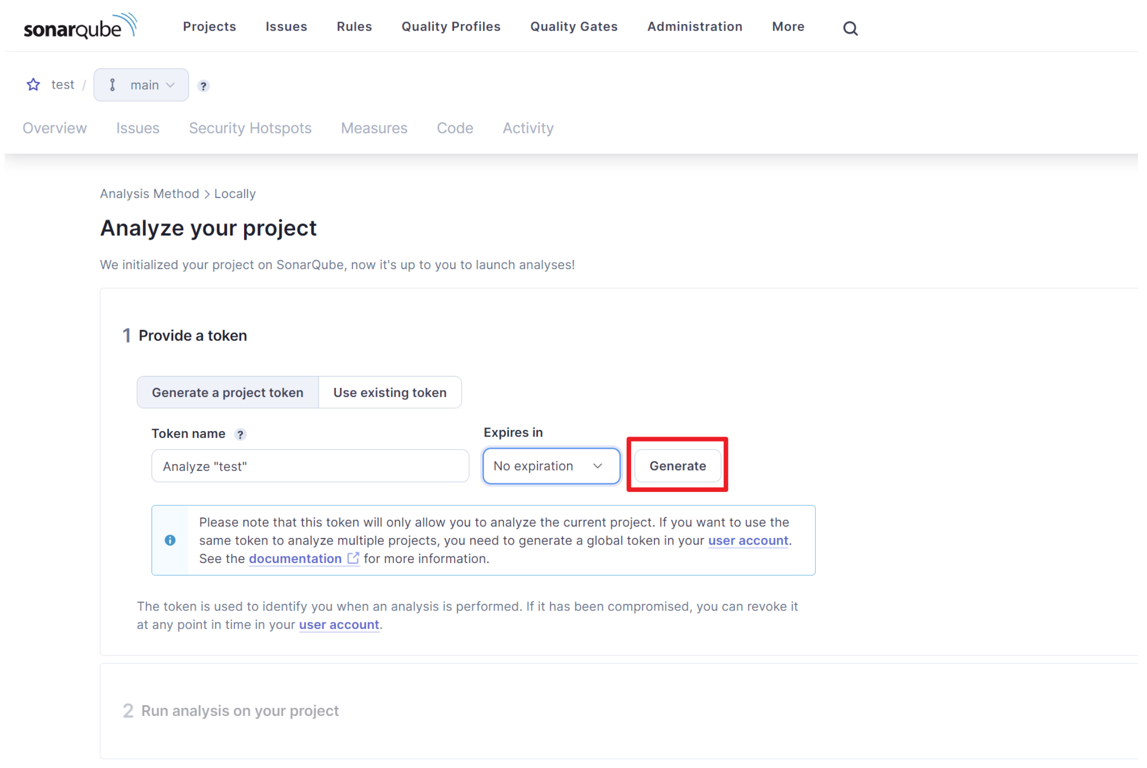
Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.
Recommended for projects following continuous delivery.

Reference branch

Choose a branch as the baseline for the new code.
Recommended for projects using feature branches.

프로젝트 토큰 발급



Analysis Method > Locally

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

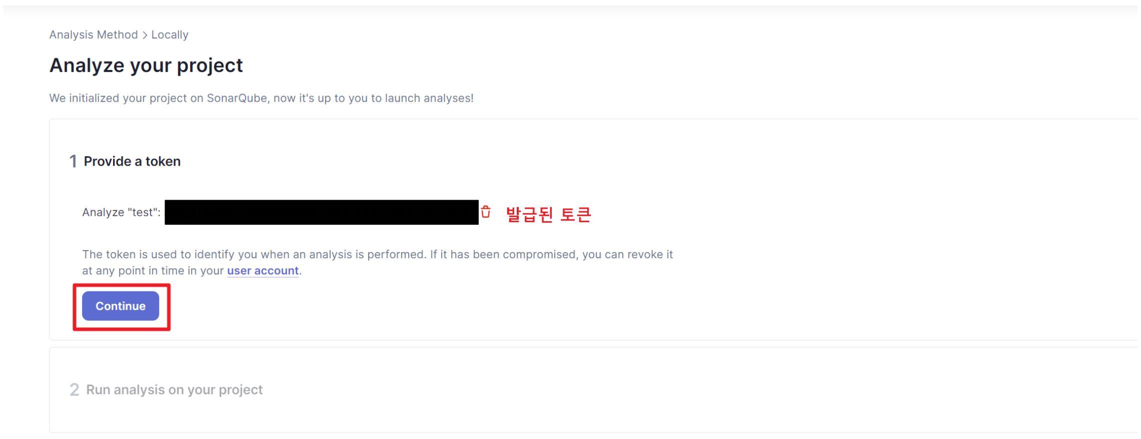
Token name: Analyze "test"

Expires in: No expiration

Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your user account. See the documentation for more information.

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your user account.

2 Run analysis on your project


Analysis Method > Locally

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

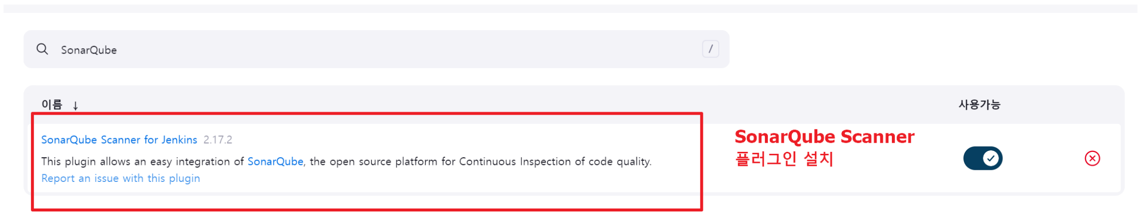
Analyze "test": [redacted] 발급된 토큰

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your user account.

2 Run analysis on your project

2. jenkins 연동

Jenkins의 SonarQube Scanner 플러그인 설치



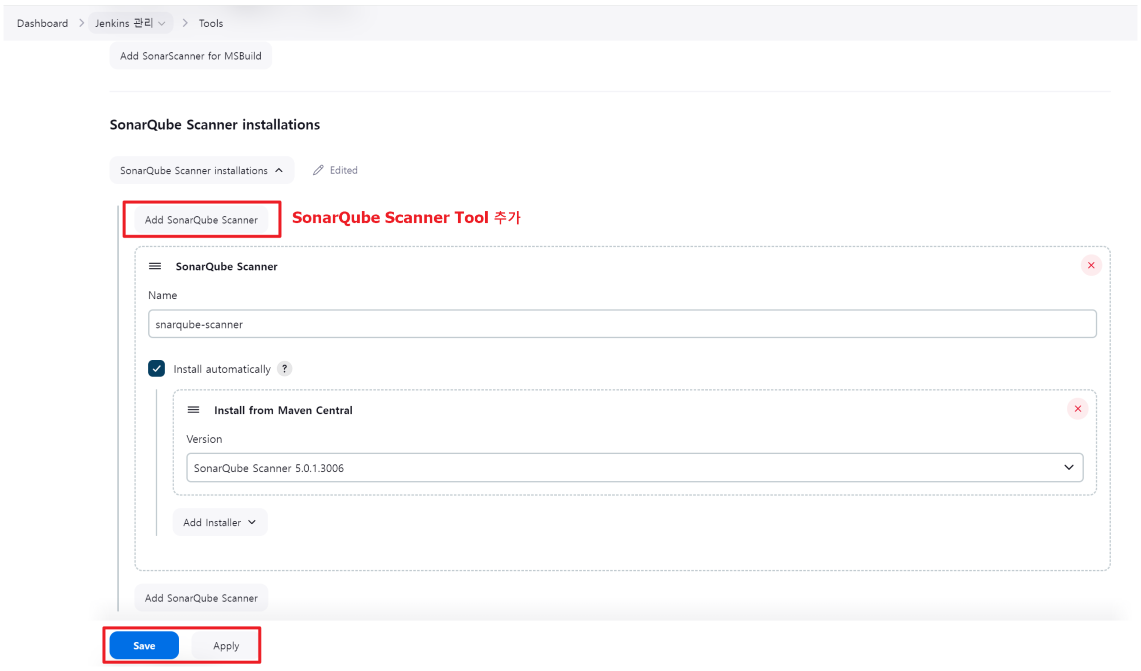
SonarQube Scanner for Jenkins 2.17.2

This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.

Report an issue with this plugin

SonarQube Scanner 플러그인 설치

Dashboar > Jenkins관리 > Tools 에 SonarQube Scanner 추가


Dashboard > Jenkins 관리 > Tools
Add SonarScanner for MSBuild

SonarQube Scanner installations

SonarQube Scanner installations
Edited
Add SonarQube Scanner

SonarQube Scanner Tool 추가

SonarQube Scanner

Name: snarqube-scanner

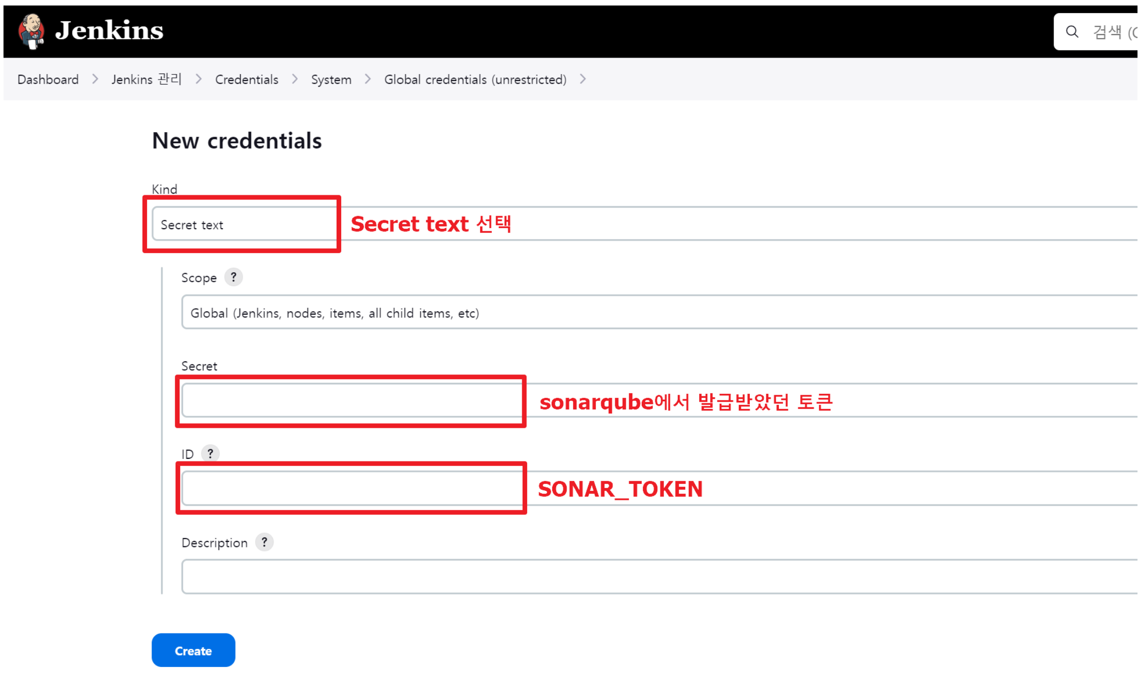
✓ Install automatically

Install from Maven Central

Version: SonarQube Scanner 5.0.1.3006

Add Installer
Add SonarQube Scanner
Save
Apply

SonarQube 접속 Credential 추가


Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Secret text — **Secret text 선택**

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

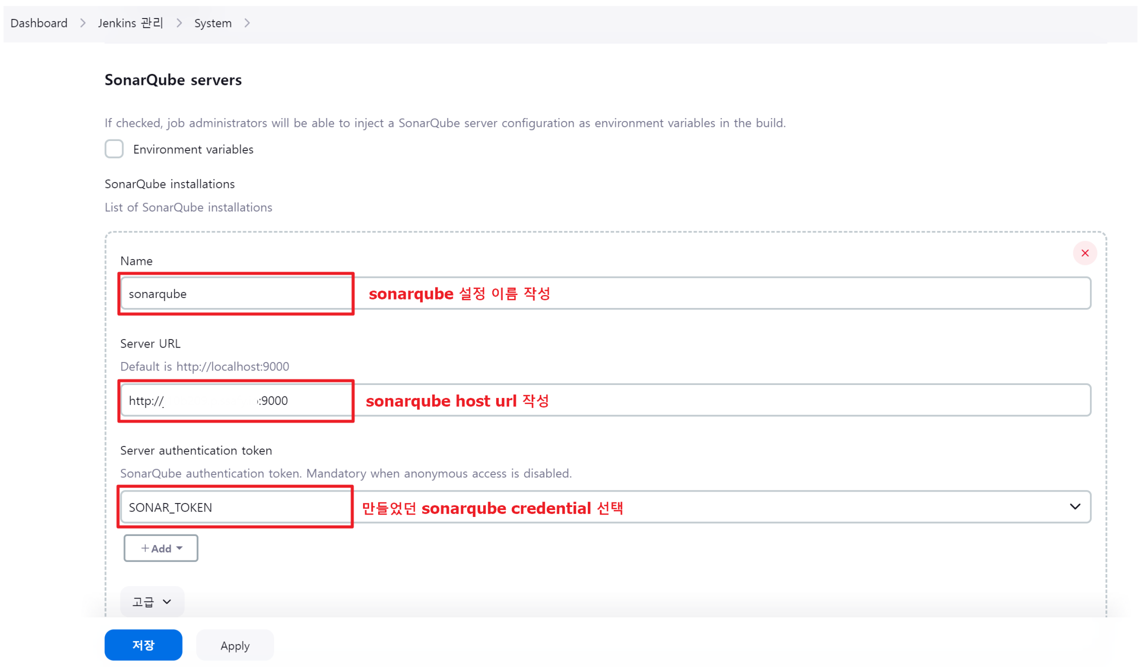
Secret: — **sonarqube에서 발급받았던 토큰**

ID: — **SONAR_TOKEN**

Description:

Jenkins 시스템 설정에 SonarQube 관련 설정 등록

Dashboard > Jenkins 관리 > System


Dashboard > Jenkins 관리 > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables

SonarQube installations

List of SonarQube installations

Name: sonarqube — **sonarqube 설정 이름 작성**

Server URL

Default is http://localhost:9000

http:// :9000 — **sonarqube host url 작성**

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

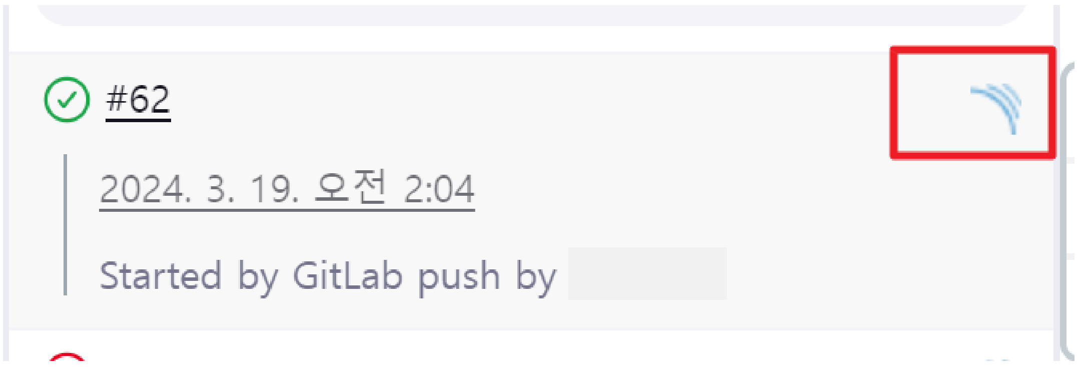
SONAR_TOKEN — **만들었던 sonarqube credential 선택**

Jenkins 파이프라인 SonarQube 관련 코드 추가

```
...

        stage('SonarQube') {
            steps {
                    # credentialId : 등록한 credential Id 값
                    # installationName : 등록한 SonarQube Server system Nam
                withSonarQubeEnv(credentialsId:"SONAR_TOKEN",installationN
                    dir('server'){
                        dir('hansotbab'){
                            sh '''
                                ls -al
                                chmod +x ./gradlew
                                ./gradlew sonar
                            '''

                        }
                    }
                }
            }
        }

...
```

3. sonarqube 코드 분석

   Jenkins & SonarQube 연동 실행



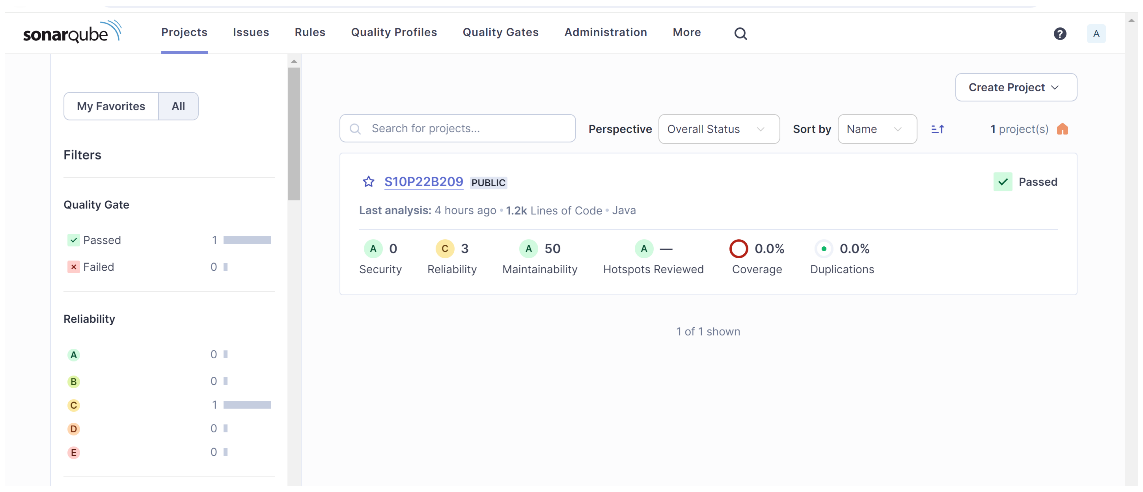
#62

2024. 3. 19. 오전 2:04

Started by GitLab push by [redacted]

- SonarQube 연동 완료

Sonarqube 코드 분석



Filters

Quality Gate

✓ Passed	1
✗ Failed	0

Reliability

A	0
B	0
C	1
D	0
E	0

1 project(s)

S10P22B209 PUBLIC — ✓ Passed

Last analysis: 4 hours ago • 1.2k Lines of Code • Java

Security	Reliability	Maintainability	Hotspots Reviewed	Coverage	Duplications
A 0	C 3	A 50	A —	0.0%	0.0%

1 of 1 shown

Nginx

```
## apt 업데이트
sudo apt-get update
sudo apt-get upgrade

## nginx 설치
sudo apt-get install nginx

## nginx 시작 및 상태 확인
sudo systemctl start nginx
sudo systemctl status nginx


## 방화벽 설정 (443, 8080, 80)
sudo ufw allow 443
sudo ufw allow 8080
sudo ufw allow 80


## 방화벽 설정 확인
sudo ufw status


## ssl 설정
sudo vim /etc/nginx/nginx.conf

server {
    listen 80 default_server;
    listen [::]:80 default_server;
    root /var/www/html;
    server_name j10b209.p.ssafy.io
}
server{
        server_name dev.example.com

        # 443 포트로 접근시 ssl을 적용한 뒤 8000포트로 요청을 전달하는 설정
        location / {
            proxy_pass (주소);
            proxy_set_header Host $host;
        }


}


## nginx 재시작
sudo systemctl restart nginx

## certbot 설치
sudo snap install --classic certbot
sudo certbot --nginx

sudo ln -s /snap/bin/certbot /usr/bin/certbot

sudo certbot --nginx -d j10b209.p.ssafy.io
```

Hadoop 설치 & HDFS 구축 with Docker

분산 시스템을 위한 EC2 서버 1대를 별도로 운영했습니다.
Hadoop 분산 시스템은 해당 서버 내에서 운영되는 3개의 Docker Container에 구축했습니다.

1. **하둡을 올릴 ubuntu를 Docker Image로 다운로드 & Docker Container 시작**

```
$ docker pull ubuntu:18.04
$ docker run -it --name hadoop-base ubuntu:18.04
```

2. **ubuntu 컨테이너 내부 접속**

```
$ docker exec -it hadoop-base /bin/bash
```

3. **라이브러리 설치 & SSH 설정 & Hadoop 설정**

3-1. 라이브러리 및 jdk 설치

```
$ apt-get update
$ apt-get install -y net-tools vim iputils-ping wget
$ apt-get install -y openssh-server openssh-client
$ apt-get install -y openjdk-8-jdk
$ java -version
$ javac -version
```

3-2. SSH 설정

```
$ service ssh restart
$ netstat -plant | grep 22
```

3-3. SSH 접속할 수 있도록 키 파일 생성 / 권한 설정

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

3-4. ssh localhost 접속 확인

```
$ ssh localhost
```

3-5. 오류가 발생하지 않도록 관련 디렉토리 생성

```
$ mkdir /var/run/sshd
```

3-6. 하둡 다운로드 & 설치

```
$ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.4/hadoop-3.3.4.tar.g
$ sudo tar -zxvf hadoop-3.3.4.tar.gz
```

3-7. bashrc 파일에 환경 변수 추가

```
$ vi ~/.bashrc
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/root/hadoop-3.3.4
export HADOOP_CONFIG_HOME=$HADOOP_HOME/etc/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
$ source ~/.bashrc
```

3-8. Hadoop 설정파일 세팅

```
$ cd $HADOOP_CONFIG_HOME

$ vi core-site.xml
<configuration>
        <property>
        <name>hadoop.tmp.dir</name>
                <value>/root/temp</value>
        </property>
        <property>
        <name>fs.default.name</name>
        <value>hdfs://master:9000</value>
        <final>true</final>
        </property>
</configuration>

$ vi hdfs-site.xml
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
                <final>true</final>
        </property>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>/root/namenode_home</value>
                <final>true</final>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>/root/datanode_home</value>
                <final>true</final>
        </property>
</configuration>

$ vi mapred-site.xml
<configuration>
        <property>
                <name>mapred.job.tracker</name>
                <value>master:9001</value>
        </property>
</configuration>

$ vi hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

$ vi yarn-site.xml
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
</configuration>

$ vi yarn-env.sh
JAVA=$JAVA_HOME/bin/java
JAVA_HEAP_MAX=-Xmx1000m

$ vi workers
master
slave1
slave2

$ vi hadoop-env.sh
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
export YARN_NODEMANAGER_USER="root"
```

3-9. root 디렉토리에서 namenode 포맷

```
$ cd
$ hadoop namenode -format
```

4. **Hadoop이 세팅된 ubuntu를 Docker image로 빌드**

```
$ docker commit 이미지이름 계정or팀이름/repository이름
ex. docker commit hadoop-base b209/hadoopbase
```

5. **namenode 1개, datanode 2개 Docker Container로 띄우기**

```
$ docker run -it -h master --name master -p 9870:9870  b209/hadoopbase
$ docker run -it -h slave1 --name worker1 --link master:master b209/hadoopbas
$ docker run -it -h slave2 --name worker2 --link master:master b209/hadoopbas
```

6. **각 node를 연결하기 위한 Docker network 설정**

6-1. Docker Network 생성 (bridge 타입)

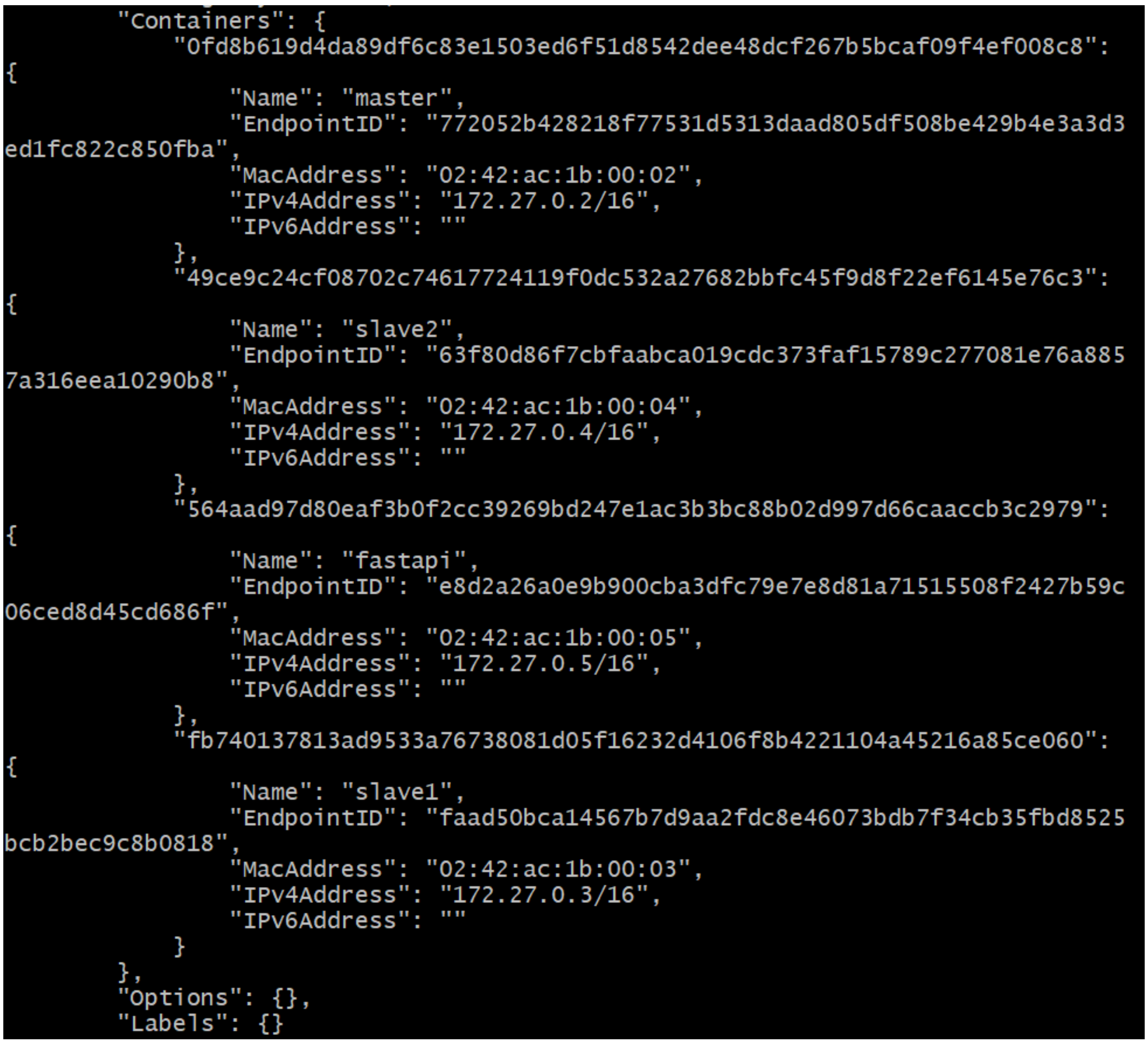
```
$ docker network create --driver bridge 네트워크이름
ex. docker network create --driver bridge hadoop-network
```

6-2. Hadoop node 컨테이너와 docker network 연결

```
$ docker network connect hadoop-network master
$ docker attach master
$ docker network connect hadoop-network worker1
$ docker attach worker1
$ docker network connect hadoop-network worker2
$ docker attach worker2
```

6-3. network 설정 확인

```
$ docker network inspect hadoop-network
```



```
        "Containers": {
            "0fd8b619d4da89df6c83e1503ed6f51d8542dee48dcf267b5bcaf09f4ef008c8":
{
                "Name": "master",
                "EndpointID": "772052b428218f77531d5313daad805df508be429b4e3a3d3
ed1fc822c850fba",
                "MacAddress": "02:42:ac:1b:00:02",
                "IPv4Address": "172.27.0.2/16",
                "IPv6Address": ""
            },
            "49ce9c24cf08702c74617724119f0dc532a27682bbfc45f9d8f22ef6145e76c3":
{
                "Name": "slave2",
                "EndpointID": "63f80d86f7cbfaabca019cdc373faf15789c277081e76a885
7a316eea10290b8",
                "MacAddress": "02:42:ac:1b:00:04",
                "IPv4Address": "172.27.0.4/16",
                "IPv6Address": ""
            },
            "564aad97d80eaf3b0f2cc39269bd247e1ac3b3bc88b02d997d66caaccb3c2979":
{
                "Name": "fastapi",
                "EndpointID": "e8d2a26a0e9b900cba3dfc79e7e8d81a71515508f2427b59c
06ced8d45cd686f",
                "MacAddress": "02:42:ac:1b:00:05",
                "IPv4Address": "172.27.0.5/16",
                "IPv6Address": ""
            },
            "fb740137813ad9533a76738081d05f16232d4106f8b4221104a45216a85ce060":
{
                "Name": "slave1",
                "EndpointID": "faad50bca14567b7d9aa2fdc8e46073bdb7f34cb35fbd8525
bcb2bec9c8b0818",
                "MacAddress": "02:42:ac:1b:00:03",
                "IPv4Address": "172.27.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
```

위와 같이 연결한 컨테이너들이 모두 출력되어야 합니다.

7. **각 node에 대해 ssh 연결 설정**

7-1. master, worker1, worker2 노드의 ip 주소 확인

```
$ docker inspect master | grep IPAddress
$ docker inspect worker1 | grep IPAddress
$ docker inspect worker2 | grep IPAddress
```

7-2. master node 컨테이너 진입

```
$ docker exec -it master /bin/bash
```

7-3. 설정 파일 수정

```
$ vi /etc/hosts
aaa.xxx.yyy.zzz master
aaa.xxx.yyy.ppp worker1
aaa.xxx.yyy.qqq worker2
(inpsect로 확인한 ip 주소 대입)
```

7-4. master, worker1, worker2에서 ssh 재시작

```
$ service ssh restart
$ exit
$ docker exec -it worker1 /bin/bash
$ service ssh restart
$ exit
$ docker exec -it worker2 /bin/bash
$ service ssh restart
$ exit
```

7-5. ssh 접속 확인

```
$ docker exec -it master /bin/bash
$ ssh worker1
$ ssh worker2
$ ssh worker1
$ ssh master
$ ssh worker2
$ ssh master
```

8. **Data Node 설정 (master에서 수행)**

```
$ cd $HADOOP_CONFIG_HOME
$ vi workers
worker1
worker2
master
```

9. **Hadoop namenode format (master에서 수행)**

```
$ hadoop namenode -format
```

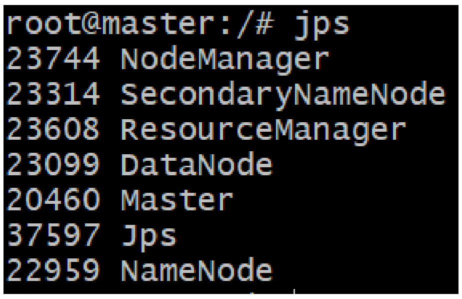
10. **Hadoop 실행 & 작동 확인 (master에서 수행)**

10-1. root로 이동 후 hadoop 실행

```
$ cd
$ start-all.sh
```

10-2. 작동 확인

```
$ jps

ResourceManager, DataNode, SecondaryNameNode, NodeManager, Jps, NameNode가
모두 출력되어야 합니다.
```



```
root@master:/# jps
23744 NodeManager
23314 SecondaryNameNode
23608 ResourceManager
23099 DataNode
20460 Master
37597 Jps
22959 NameNode
```

FastAPI (with Docker) & HDFS 연결

Hadoop이 설치된 EC2 서버와 Spring Boot, DB 등을 구동하는 EC2 서버가 서로 분리되어 있으므로, 사용자가 입력한 데이터를 HDFS에 저장할 때 Spring Boot와 HDFS를 연결해 줄 FastAPI 애플리케이션을 별도로 두었습니다.

해당 Fast API 애플리케이션은 Hadoop이 설치된 EC2 서버에서 도커 컨테이너로 구동되기 때문에, FastAPI 컨테이너와 Hadoop의 master node의 컨테이너 역시 같은 네트워크로 연결하여 FastAPI 컨테이너에서 HDFS에 접근할 수 있도록 설정했습니다.

1. **Docker Hub에서 FastAPI 이미지 pull & Run Container**

```
$ docker pull repository이름/image이름
ex. docker pull st3llartois17/hsb-fastapi

$ docker run --name container이름 -d -p 8000:8000 repository이름/image이름
ex. docker run --name hsb-fastapi -d -p 8000:8000 st3llartois17/hsb-fastapi

$ docker network connect hadoop-network hsb-fastapi
$ docker restart hsb-fastapi
```

2. **iptables의 FORWARD chain 중 DOCKER-USER에 inbound rule 추가**

```
$ iptables -I DOCKER-USER -p tcp --dport 8000 -j DROP
$ iptables -I DOCKER-USER -s <EC2 IP> -p tcp --dport 8000 -j ACCEPT
```

3. **FastAPI 컨테이너와 Hadoop Namenode 컨테이너 ssh로 연결**

3-1. FastAPI 컨테이너 내부에 진입 -> SSH 키 생성

```
$ docker exec -it hsb-fastapi /bin/bash

$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys

hsb-fastapi의 id_rsa.pub 복사 -> master의 .ssh/authorized_keys에 저장
master의 id_rsa.pub 복사 -> hsb-fastapi의 .ssh/authorized_keys에 저장
```

3-2. 설정파일 변경

```
$ docker exec -it master /bin/bash

$ vi /etc/hosts
hadoop-network의 hsb-fastapi ip 주소 추가 (ex. 123.456.789.000 hsb-fastapi)
```

3-3. ssh 설정 반영 및 접속 테스트

```
$ service ssh restart
$ exit

$ docker exec -it worker1 /bin/bash
$ service ssh restart
$ exit

$ docker exec -it worker2 /bin/bash
$ service ssh restart
$ exit

$ docker exec -it hsb-fastapi /bin/bash
$ service ssh restart
$ exit

$ docker exec -it master /bin/bash
$ ssh hsb-fastapi
$ ssh master
$ exit
```