



# 포팅 매뉴얼

## 1. 버전 정보

- 1. Environments
- 2. Build & Distribute
- 3. Deployment Command
- 4. MariaDB Connection
- 5. Files ignored

## 2. 프로젝트에서 사용하는 외부 서비스 정보 정리

## 3. DB 덤프

## 4. 시연 시나리오

## 1. 버전 정보

### 1. Environments

- Backend
  - spring boot : 3.2.2
  - amazon corretto : 17.0.9
  - intellij : 2023.3.2
  - android studio : 2023.1.1.27
  - kotlin : 1.8.0
  - wearable : 18.1.0
  - mariadb : 10.3.23
- Mobile
  - flutter : 3.16.7
  - firebase : 9.0.3
  - firebase\_core : 2.24.2
  - firebase messaging : 14.7.10
  - provider : 6.0.1
  - kotlin : 1.8.0
  - android studio : 2023.1.1.27

- kakao\_flutter\_sdk : 1.8.0

## 2. Build & Distribute

- Docker
  - Portainer
  - Jenkins
  - nginx

## ▼ 3. Deployment Command

### ▼ docker-compose.yml

```
// docker-compose.yml

version: '3.0'

services:
  nginx:
    networks:
      - appnet
    ports:
      - '80:80'
      - '443:443'
    image: nginx

  springboot:
    networks:
      - appnet
    command: sh -c 'if [ -e /app.jar ]; then java -jar /app.j
    image: openjdk:17
    environment:
      - TZ=Asia/Seoul

  jenkins:
    ports:
      - '8080:8080'
    user: root
    networks:
      - jenkinsnet
    image: jenkins/jenkins:jdk17
```

```

sonarqube:
  ports:
    - '9000:9000'
  networks:
    - jenkinsnet
  image: sonarqube

portainer:
  ports:
    - '9443:9443'
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - portainer_data:/data
  image: portainer/portainer-ce:latest

volumes:
  portainer_data: {}

networks:
  jenkinsnet: {}
  appnet: {}

```

- EC2 서버에서 docker-compose.yaml에 쓰인 모든 포트 열어줘야한다.

```

// 포트 열려있나 확인하는 명령어
sudo ufw status

// 포트 여는 명령어
sudo ufw allow [포트번호]

```

- 편의를 위해 EC2 디렉터리 구조를 아래와 같이 만들었다. ( /home 에서 실행)

```

mkdir docker-compose
mkdir docker-compose/volumes
mkdir deploy
mkdir deploy/backend

```

- 이후 docker-compose 디렉토리로 이동하여 다음 명령어를 실행한다.

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

## Portainer에서 네트워크 묶기

Containers in network				
Container Name	IPv4 Address	IPv6 Address	MacAddress	Actions
docker-compose-nginx-1	172.21.0.3/16	-	02:42:ac:15:00:03	<a href="#">Leave Network</a>
docker-compose-springboot-1	172.21.0.2/16	-	02:42:ac:15:00:02	<a href="#">Leave Network</a>

- network 탭에서 다음과 같이 설정해준다.

## Jenkins

- 아래의 `deploy.sh` 파일을 만들어 `/deploy/backend` 디렉터리에 넣는다.

```
// docker-compose-jenkins-2 컨테이너의 jar 파일을 호스트(EC2서버)로 옮기  
docker cp docker-compose-jenkins-2:/var/jenkins_home/workspace/b  
  
// 호스트에 옮긴 jar파일을 docker-compose-springboot-1 컨테이너의 home  
docker cp /home/ubuntu/deploy/backend/app.jar docker-compose-spr  
  
// docker-compose-springboot-1 컨테이너를 재시작한다.  
docker restart docker-compose-springboot-1
```

### ▼ 이후 Jenkins 설정

웹 브라우저 URL 검색 창에 호스트 서버 IP:9090을 검색했을 때 아래의 화면이 나오면 정상 설치 된 것이다. 젠킨스는 편리하게 사용할 수 있도록 웹 인터페이스를 제공한다. 처음 접속 시 계정을 설정해야한다.

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

Continue

- portainer에서 jenkins 로그를 확인하면 첫 실행 시 비밀번호를 알 수 있다. 그 비밀번호를 입력하자.
  - 예시) 48acef15f078b1bdf20b29757bdaca9e

## 이후 과정

# 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.

Getting Started

## Create First Admin User

계정명

암호

암호 확인

이름

이메일 주소

Jenkins 2.441

[Skip and continue as admin](#) [Save and Continue](#)

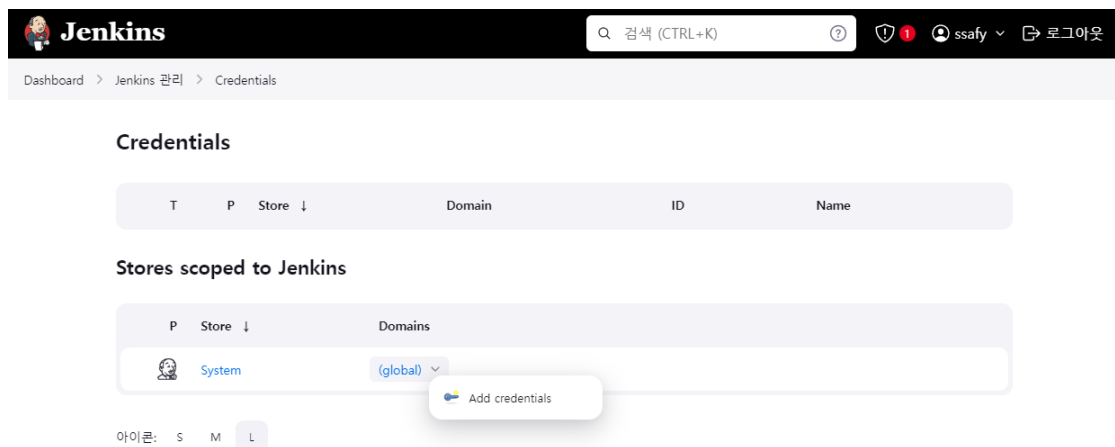
- 다음으로 필요한 플러그인을 설치한다.
  - Jenkins 관리 → Plugins → Available plugins 들어가서 아래의 플러그인을 설치한다.
    - gitlab : gitlab 사용을 위함
    - maven을 사용하면 maven Integration을 설치해야겠지만, 우리 프로젝트의 경우 Gradle을 사용하기 때문에 따로 설치하지 않았다. (Gradle 관련 플러그인 설정은 아예 안했음)

# Credentials 생성

Credentials는 외부 서버, 시스템에 대한 인증 정보(비밀키, 패스워드, 토큰값)을 말한다. **Credentials로 인증 정보를 저장해두고 젠킨스가 해당 Credentials을 통해 외부 서버의 접근 권한을 얻는다.**

젠킨스는 원격 저장소(GitHub, GitLab)로부터 git clone을 하여 원격 저장소의 파일을 가져와 빌드를 수행한다. 이를 위해서는 원격 저장소에 접근할 수 있는 권한이 필요하다. 때문에 gitlab의 access token을 발급받아 credentials을 만든다. 또한 필자의 젠킨스는 소나 큐브 서버에도 접근하므로 소나 큐브의 token을 발급받아 credentials을 만든다.

Jenkins 관리 → Credentials로 이동한다. 아래와 같이 Add credentials 버튼을 누른다.



kind는 Username and Password를 선택한다.

Username : 원격 저장소(GitLab, GitHub)의 닉네임

Password : gitlab의 토큰값 저장

ID : 임의로 지정

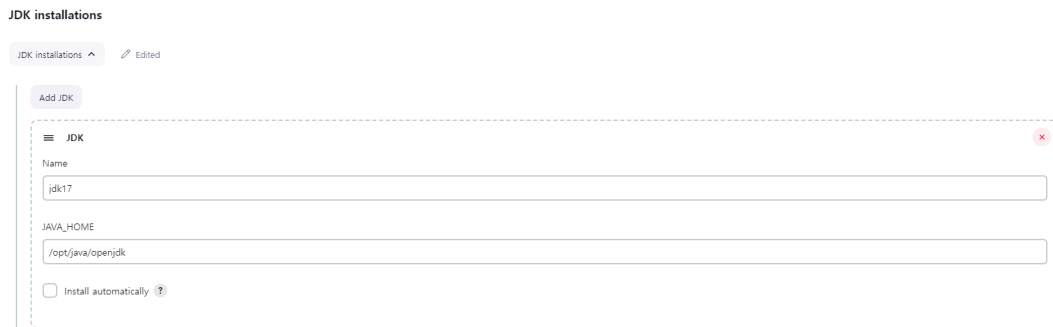
## 환경변수 설정

jenkins 관리 → Tool로 이동한다.

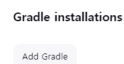
Spring Boot 앱 빌드에 사용되는 jdk가 jenkins 컨테이너 안에 있어야한다. jenkins 컨테이너를 받으면서 이미 jdk가 포함되어 있으므로 해당 jdk를 사용한다. 아래와 같이 jdk 경로



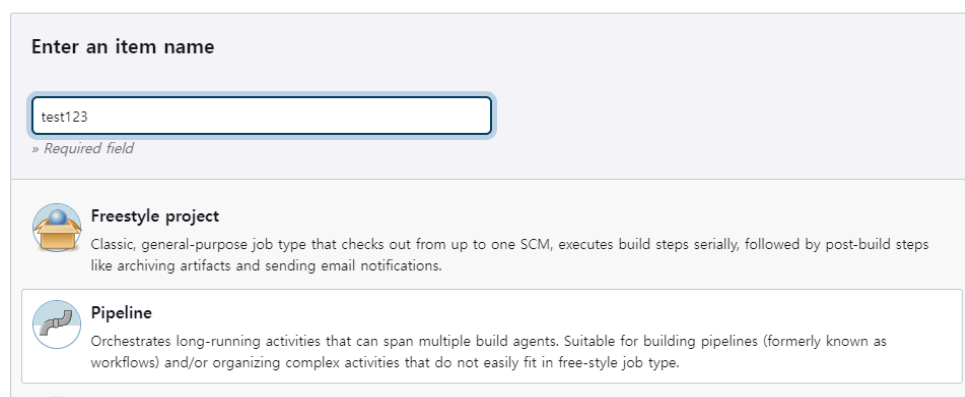
를 지정한다. 만약 설치되어 있지 않다면 Install automatically 체크박스를 체크한다.



- 여기에서도 Gradle은 따로 설정해주지 않았음



## 이제 파이프라인 테스트!



- 파이프라인부터 생성해주자

이후 구성으로 이동해서 Pipeline 스크립트에 다음과 같이 테스트 스크립트를 작성해주자.

```
pipeline {  
  agent any  
  
  stages {  
    stage('Clone Repository') {
```

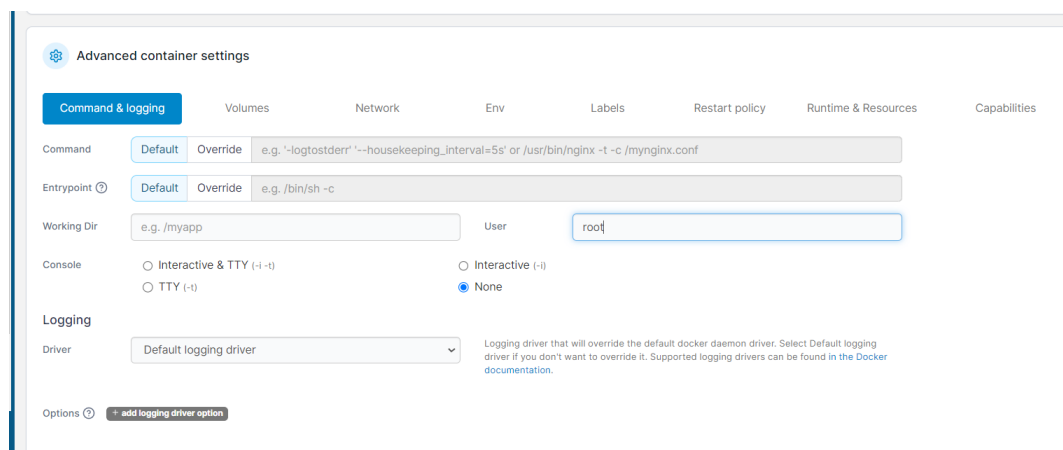
```

        steps {
            script {
                // 내가 배포할 브랜치 이름
                git branch: 'develop', credentialsId: 'git-credentials'
            }
        }
    }

    stage('Build') {
        steps {
            // dir 위치의 경우, 위에서 받아온
            // gradlew이 있는 경로를 지정해주기
            dir('back/yourpilling') {
                // sh 'whoami' => permission denied가 떴을
                sh 'chmod +x ./gradlew'
                // sh './gradlew build' => 테스트 코드 돌려보
                sh './gradlew build -x test' // 테스트 코드
            }
        }
    }
}

```

- 테스트 코드 에러가 나는 상황이었기 때문에 일단 테스트는 건너뛰고 테스트
  - success인 경우 이제 deploy까지 작성해주자.
- 만약 ./gradlew 하는 과정에서 Permission Denied 권한 문제 발생 시 젠킨스 컨테이너 생성하는 과정에서 root로 설정해줘야한다



```

pipeline {
    agent any

    stages {
        stage('Clone Repository') {
            steps {
                script {
                    git branch: 'develop', credentialsId: 'gitlab-credentials'
                }
            }
        }

        stage('Build') {
            steps {
                // dir('경로') => gitlab url에 있는 디렉토리
                // sh 명령어를 실행한다. (build.gradle 실행)
                // 그럼 jar파일이 빌드되는데, 이것을 EC2 서버에 업로드
                dir('back/yourpilling') {
                    // sh 'whoami'
                    sh 'chmod +x ./gradlew'
                    // sh './gradlew build'
                    sh './gradlew build -x test'
                }
            }
        }

        // build.gradle에 소나큐브 정보 추가하고 다시 해보기
        // stage('SonarQube analysis') {
        //     steps {
        //         // withSonarQubeEnv('sonarqube') {
        //         //             dir('back/yourpilling') {
        //         //                 sh './gradlew sonarqube'
        //         //             }
        //         //         }
        //     }
        // }

        stage('Deploy') {
            steps {
                // EC2 서버에 ssh로 접속해서 deploy.sh 실행
                // deploy.sh 내용
            }
        }
    }
}

```

```
// - EC2서버로 jar파일 던져줌
// - EC2서버에서 springboot 컨테이너
// - springboot 컨테이너 재시작하
sh 'ssh ubuntu@i10b101.p.ssafy.io "sudo sh ~/
```

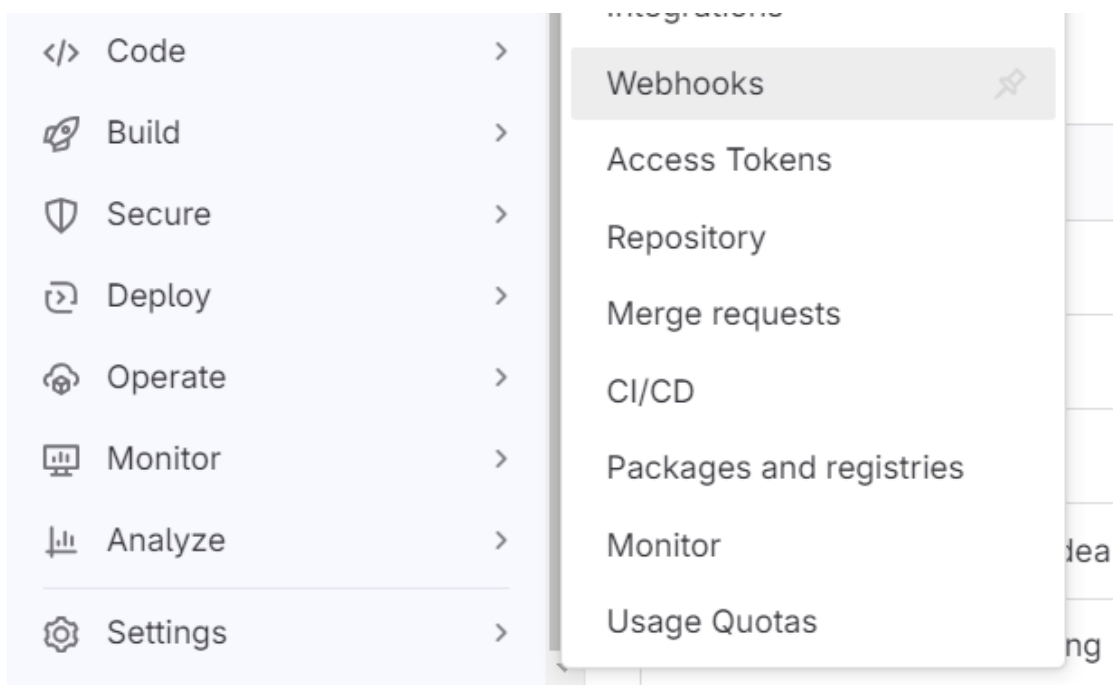
## ※ 스프링부트 컨테이너에서는 어떻게 app.jar를 실행하는가?

```
CMD sh -c if [ -e /app.jar ]; then java -jar /app.jar --spring.profiles.active=production; else echo "app.jar not found, skipping execution"; fi
```

```
sh -c if [ -e /app.jar ]; then java -jar /app.jar --spring.pr
```

- 스프링부트 컨테이너 생성 시 위의 cmd를 입력해주기 때문
  - 내용 : app.jar를 실행하고, 만약 app.jar가 존재하지 않는다면 not found 예외처리를 해준다

## 마지막으로 gitlab에서 웹훅 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

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

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

### Trigger

☒ Push events

☐ All branches

☐ Wildcard pattern

☒ Regular expression

Regular expressions such as `^(feature|hotfix)/` are supported.

## 깃랩에서 웹훅 설정 - 정규식

The screenshot shows the Jenkins configuration interface. On the left, the 'General' tab is active. In the 'Build Triggers' section, the checkbox 'Build when a change is pushed to GitLab' is checked. Below it, under 'Enabled GitLab triggers', 'Push Events' and 'Opened Merge Request Events' are also checked. The 'Rebuild open Merge Requests' dropdown menu is set to 'Never'. At the bottom, there are '저장' (Save) and 'Apply' buttons.

## 젠킨스에서 GitLab 업데이트 시 build하기 체크

참고: <https://rexiann.github.io/2020/11/25/connect-jenkins-and-gitlab-using-docker.html>

## nginx

▼ 먼저 nginx 설정 파일이다.

```

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

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {

        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }
}

server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

```

```

root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;
    server_name i10b101.p.ssafy.io; # managed by Certbot


location / {
    # as directory, then fall back to displaying a 404.
    #

    proxy_pass http://127.0.0.1:8080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;

    #try_files $uri $uri/ =404;

}


listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/i10b101.p.ssafy.io/
ssl_certificate_key /etc/letsencrypt/live/i10b101.p.ssafy
include /etc/letsencrypt/options-ssl-nginx.conf; # manage
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed

}
server {
    if ($host = i10b101.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80 ;
    listen [::]:80 ;
    server_name i10b101.p.ssafy.io;

```

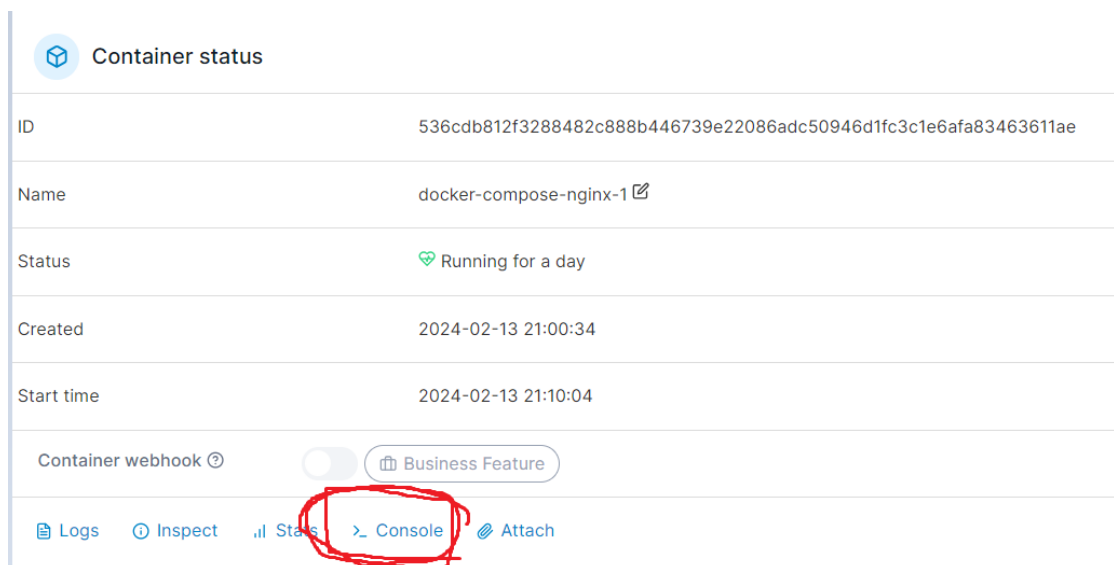
```

location / {
    proxy_pass http://127.0.0.1:8080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

return 404; # managed by Certbot
}

```

- 이후 portainer에서 nginx 콘솔에 접속하여 다음과 같이 명령어를 입력한다.



```

# portainer에서 nginx 콘솔 접속
cd /etc/nginx/conf.d
vi default.conf # 이후 상술한 nginx 정보 복사붙여넣기
# 그리고 portainer에서 같은 ip로 묶어놨으므로 127.0.0.1 부분을
# 172.어쩌구 스프링부트 내부 ip로 바꿔줘야함

# 그리고 https 사용하는 경우 EC2 서버의 letsencrypt 파일 옮겨줘야한다
sudo docker cp letsencrypt/ docker-compose-nginx1:/etc

```

- 이후 portainer를 이용하여 nginx restart



## 4. MariaDB Connection

- Spring Boot에서 연결
- Standard TCP/IP 연결

## 5. Files ignored

- Flutter 프로젝트

```
gradle-wrapper.jar
/.gradle
/captures/
/gradlew
/gradlew.bat
/local.properties
GeneratedPluginRegistrant.java

# Remember to never publicly share your keystore.
# See https://flutter.dev/docs/deployment/android#reference-the-key.properties
**/*.keystore
**/*.jks
```

- SpringBoot 프로젝트

```
HELP.md
.gradle
build/
!gradle/wrapper/gradle-wrapper.jar
!**/src/main/**/build/
!**/src/test/**/build/

### STS ###
.appt_generated
.classpath
.factorypath
.project
.settings
.springBeans
```

```

.sts4-cache
bin/
!*/src/main/**/bin/
!*/src/test/**/bin/

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr
out/
!*/src/main/**/out/
!*/src/test/**/out/

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/

### VS Code ###
.vscode/
/src/main/resources/application-dev.yml
/logs/

```

## 2. 프로젝트에서 사용하는 외부 서비스 정보 정리

- 카카오 로그인 API
  - <https://developers.kakao.com/docs/latest/ko/rest-api/getting-started>
- 네이버 쇼핑 API
  - <https://developers.naver.com/docs/serviceapi/search/shopping/shopping.md#쇼핑>
- ▼ FCM
  - Springboot 프로젝트에서 Firebase에 접근하기 위해 사용하는 json

```
{
  "type": "service_account",
  "project_id": {project_id},
  "private_key_id": {private_key_id},
  "private_key": {private_key},
  "client_email": "firebase-adminsdk-4vyxk@pillin-d420f.iam.gse
  "client_id": "109372200846868691933",
  "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/oa
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/r
  "universe_domain": "googleapis.com"
}
```

- Andriod, Kotlin에서 Firebase와 FCM 토큰을 주고 받기 위한 json 파일

```
{
  "project_info": {
    "project_number": "1055153492019",
    "project_id": {project_id},
    "storage_bucket": {storage_bucket}
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:1055153492019:android:28b763f93b8
        "android_client_info": {
          "package_name": "com.example.push"
        }
      },
      "oauth_client": [],
      "api_key": [
        {
          "current_key": {current_key}
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    }
  ]
}
```

```

    },
    {
      "client_info": {
        "mobilesdk_app_id": "1:1055153492019:android:4c982d7949",
        "android_client_info": {
          "package_name": "com.example.yourpilling"
        }
      },
      "oauth_client": [],
      "api_key": [
        {
          "current_key": {current_key}
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    }
  ],
  "configuration_version": "1"
}

```

### 3. DB 덤프

pillin\_dump.sql

### 4. 시연 시나리오

- 눈에 좋은 루테인 영양제를 등록하는 상황
- 등록 후 메인 확인
- 알람설정 1분
- 영양제 복용 하나씩 먹다가 전체복용!!
- 정리하다가 영양제를 떨어뜨림
- 재고 수정을 하고나서, 재고가 다 떨어지고 push알림이 옴

- 재구매 알림 push가 온다
- 부족한 영양제를 구매하러 간다.
- 지금 구매해야 루틴이 안 끊기겠다!! 구매까지 간다..
- 워치에서 복용 확인가능, 분석리포트, 복용 기록 확인