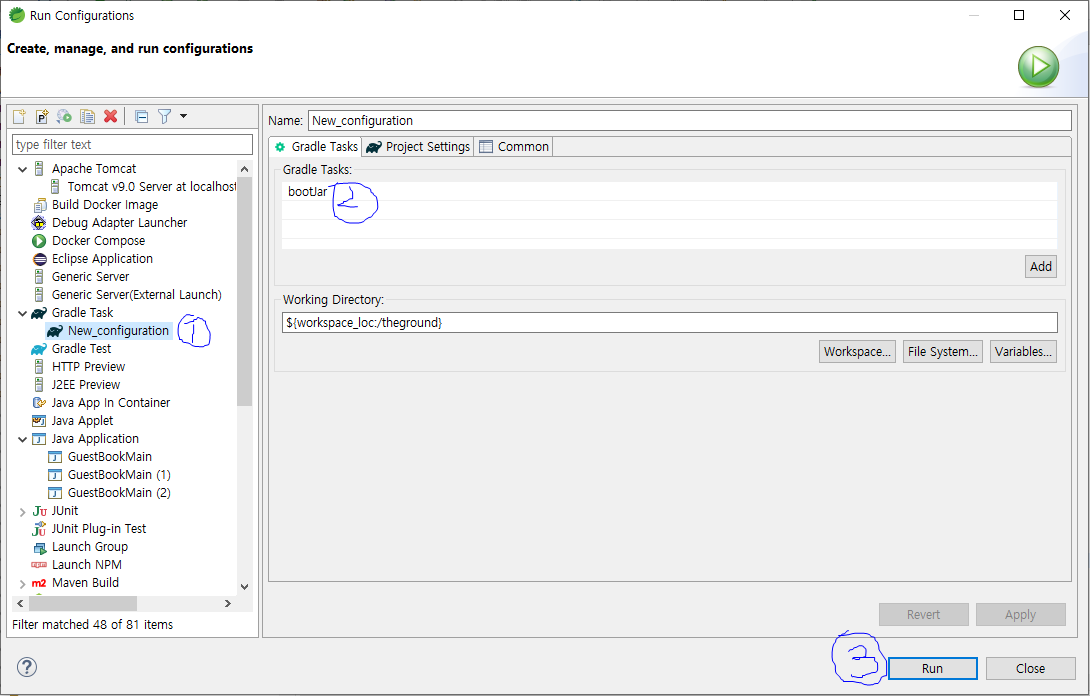
1. 버전 정리
   1. 유니티
      1. Development

* Unity Hub 3.3.0
* Unity Editor 2021.3.11f1
* Meta Oculus 1.76.0.0
  + 1. Build
* Meta Quest Developer Hub 3.0.1
  + 1. Packages
* Laboratory Interior 1.1
* Chemistry Laboratory Equipment 2.0
* Digestive System 1.0
* VR Interaction Framework 1.82
* Solar System 4.7
  1. 백엔드
     1. JDK
* Openjdk version 1.8.0 312
* OpenJDK Runtime Environment (build 1.8.0\_312-8u312-b07-0ubuntu1~20.04-b07)
* OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
  + 1. SpringBoot
* SpringBoot 2.7.3
  + 1. DB
* 10.3.34-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04
  1. CI/CD
     1. Aws ec2
* Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1019-aws x86\_64)
  + 1. Docker
* Docker version 20.10.18, build b40c2f6
  + 1. Jenkins
* Jenkins 2.367
  + 1. Nginx
* Nginx/1.18.0 (Ubuntu)

1. 빌드
   1. 백엔드
      1. Gui 빌드(STS 기준)
2. 프로젝트 우클릭->Run As->Run Configurations



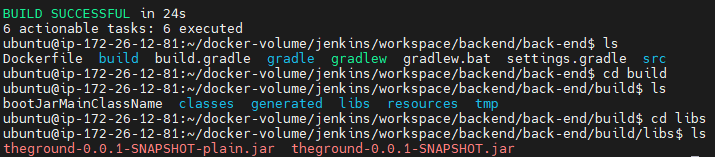
Gradle Task->New\_configuration 클릭한 뒤

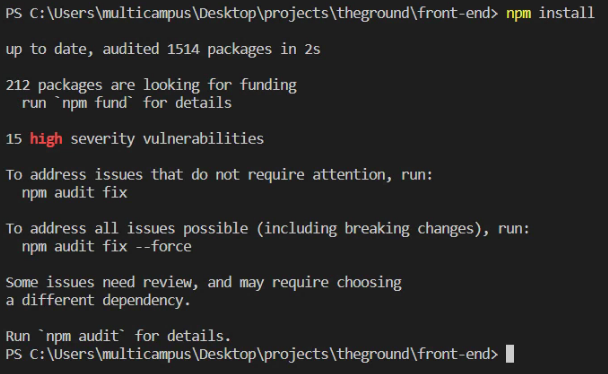
Gradle Tasks에 bootJar 추가해주고

Run 버튼을 클릭합니다. 빌드가 완료되면 해당 프로젝트의 build/libs 폴더에 jar파일이 생성됩니다.

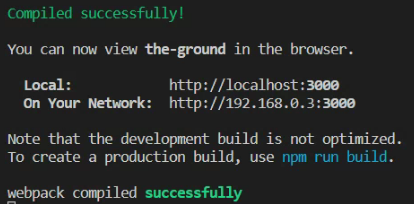
* + 1. 명령어 빌드

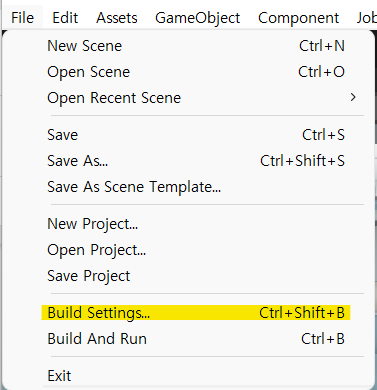
Gradlew파일이 있는 경로로 이동해서 sudo ./gradlew clean build -x test를 입력합니다. 자동적으로 빌드가 진행되고 마찬가지로 해당 프로젝트의 build/libs 폴더에 jar파일이 생성됩니다. 또한 위에서 입력한 것 처럼 clean이 붙게 되면 기존에 만들었던 jar을 지우고 jar을 생성해줍니다.

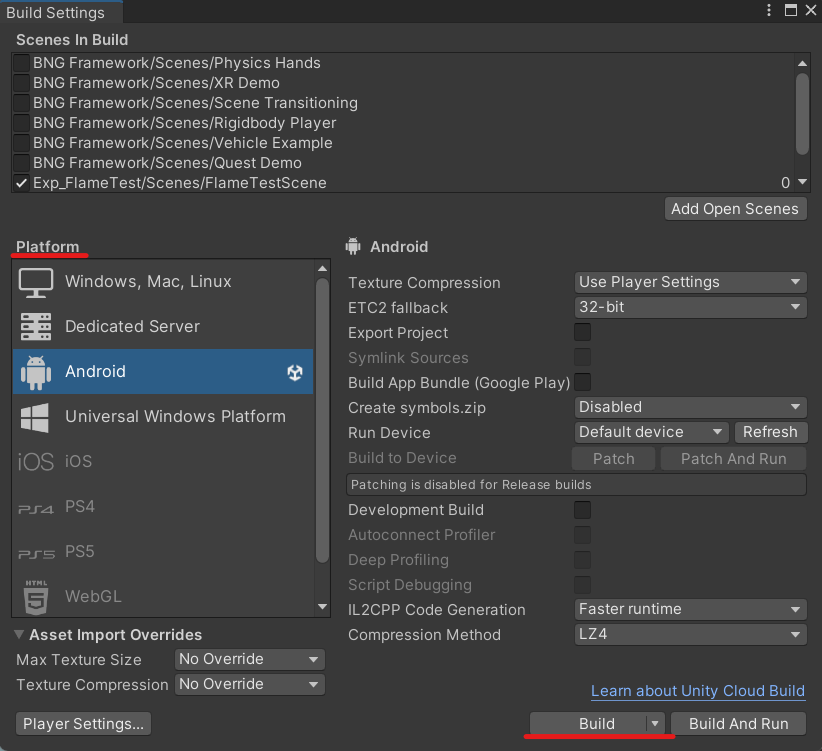


* 1. 프론트엔드
     1. 빌드

프론트엔드 프로젝트가 있는 경로에 가서, npm install을 해서 필요한 라이브러리를 다운받고, npm start를 통해 실행합니다.



* 1. 유니티
     1. 빌드

유니티 프로젝트 상단 메뉴에서 File - Build Setting 으로 들어갑니다.

Platform 에서 본인이 개발하려는 플랫폼을 선택하여 변경하고 Build 버튼을 눌러 apk 파일을 빌드합니다.

3. 배포 시 특이사항

A. CI/CD

aws ec2에 Mysql을 설치했고, Docker와 Jenkins를 통한 CI/CD 구축 및 ssl 을 통한 https설정이 되어있습니다. 아래에서 각각에 대한 설치/설정법을 설명하겠습니다.

우선 기본적으로, cmd에서 아래 커맨드를 통해 22번 포트에 접속합니다.

- ssh -i K7D101T.pem [ubuntu@k7d101.p.ssafy.io](mailto:ubuntu@k7d101.p.ssafy.io)

단, K7D101T.pem 파일이 있는 곳에서 실행해야 합니다.

i. Mysql

Mysql 설치

- sudo apt-get update

- sudo apt-get install mysql-server 명령어를 순서대로 입력해 mysql를 설치합니다.

- /etc/mysql/mysql.conf.d/mysqld.cnf를 수정해서 bind-address를 0.0.0.0으로 변경했습니다.

- Sudo service mysql restart를 통해 mysql을 재시작합니다.

계정 생성 및 권한 부여

sudo mysql -u root -p를 통해 mysql에 들어간 뒤,

- CREATE USER `ssafy`@'%' IDENTIFIED BY 'ssafy';

- GRANT ALL PRIVILEGES ON dream.\* TO `ssafy`@'%' IDENTIFIED BY 'ssafy';

- FLUSH PRIVILEGES;

- CREATE SCHEMA dream;

를 순서대로 입력해서 계정 생성 및 권한 설정, dream 스키마를 생성해줍니다.

ii. Openjdk 설치

아래 명령어를 차례대로 실행해서 openjdk를 설치합니다

- sudo apt-get update

- sudo apt-get install openjdk-11-jdk

iii. Docker 설치

아래 명령어를 순서대로 실행해서 Docker를 설치합니다.

- sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

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

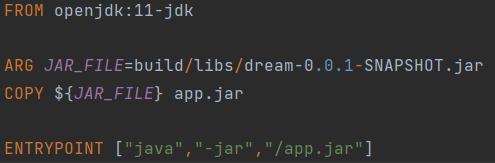
- sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

- sudo apt-get update

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

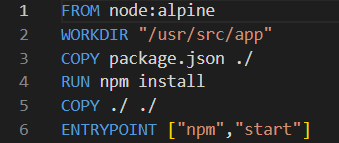
다음으로, gitlab의 springboot 프로젝트의 pom.xml이 있는 위치에 Dockerfile을 생성합니다.

Dockerfile은 다음과 같이 작성합니다.



마지막으로, gitlab의 react 프로젝트의 package.json이 있는 위치에 Dockerfile을 생성합니다.

Dockerfile은 다음과 같이 작성합니다.



iv. Docker에 Jenkins 설치 및 실행

아래의 명령어를 실행해서 Docker에 Jenkins를 설치하고 container를 생성합니다.

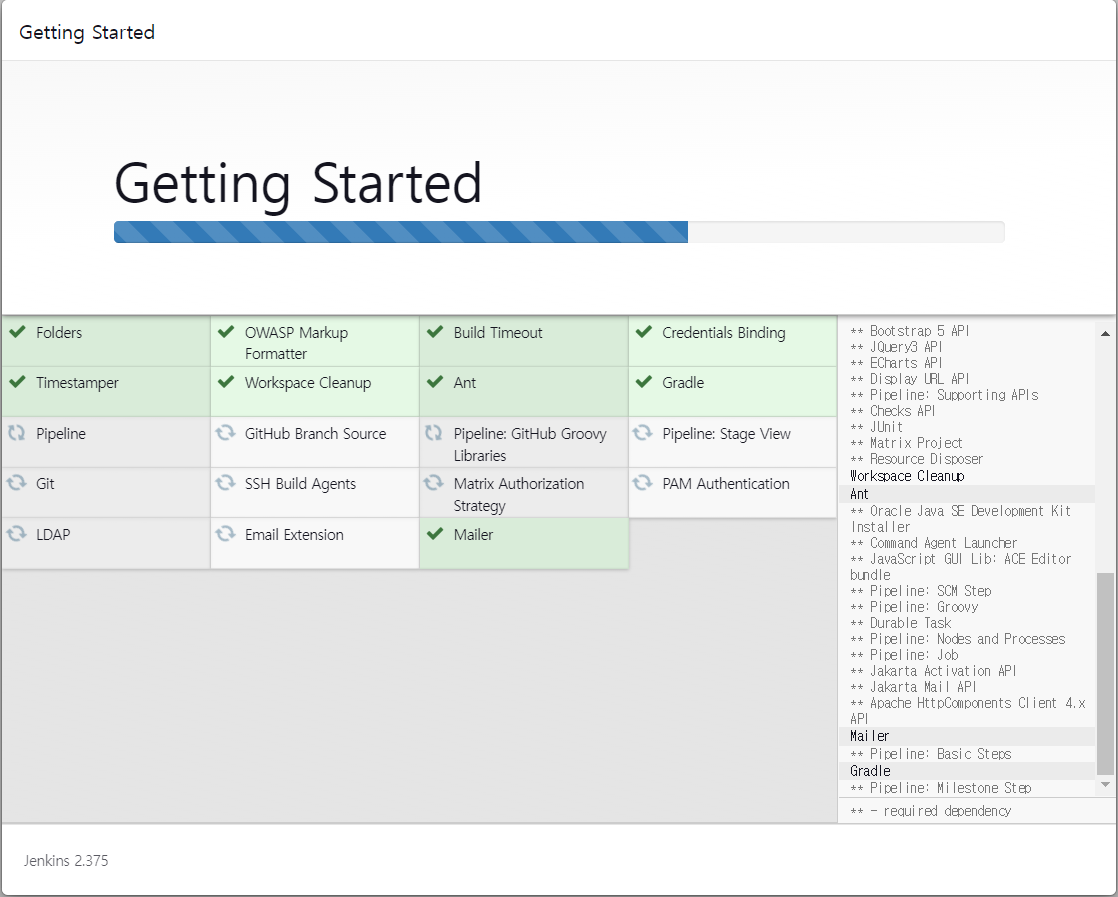
- sudo docker run -d --name jenkins -u root --privileged -p '9090:8080' -v '/home/ubuntu/docker-volume/jenkins:/var/jenkins\_home' -v '/var/run/docker.sock:/var/run/docker.sock' -v '/usr/bin/docker:/usr/bin/docker' jenkins/jenkins

아래의 명령어를 실행해서 Jenkins 설치에 필요한 패스워드를 가져옵니다

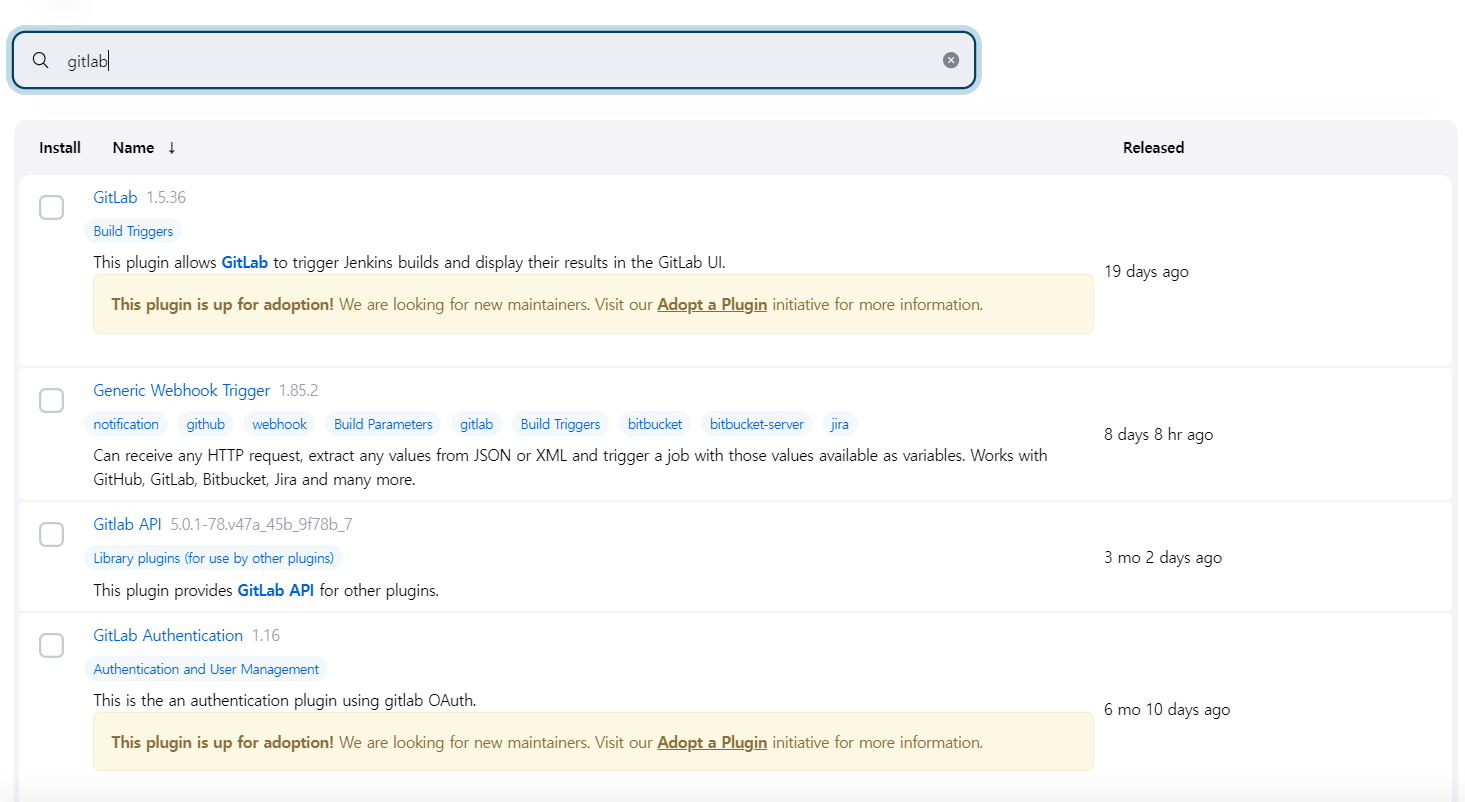
- sudo docker logs jenkins



다음으로, http://**원격서버주소**:9090을 입력해서 Jenkins 설치 페이지로 넘어갑니다.



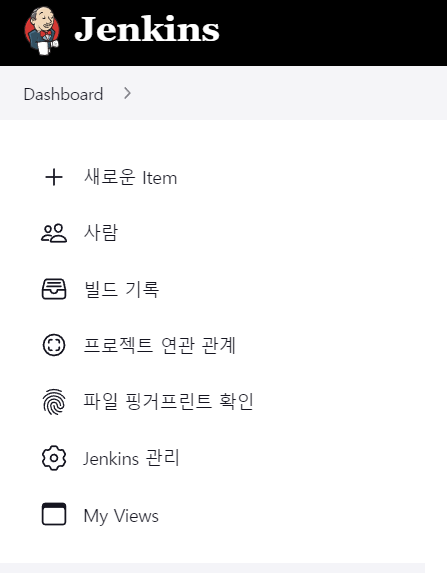
계정 설정을 한 뒤에 기본 패키지를 설치합니다



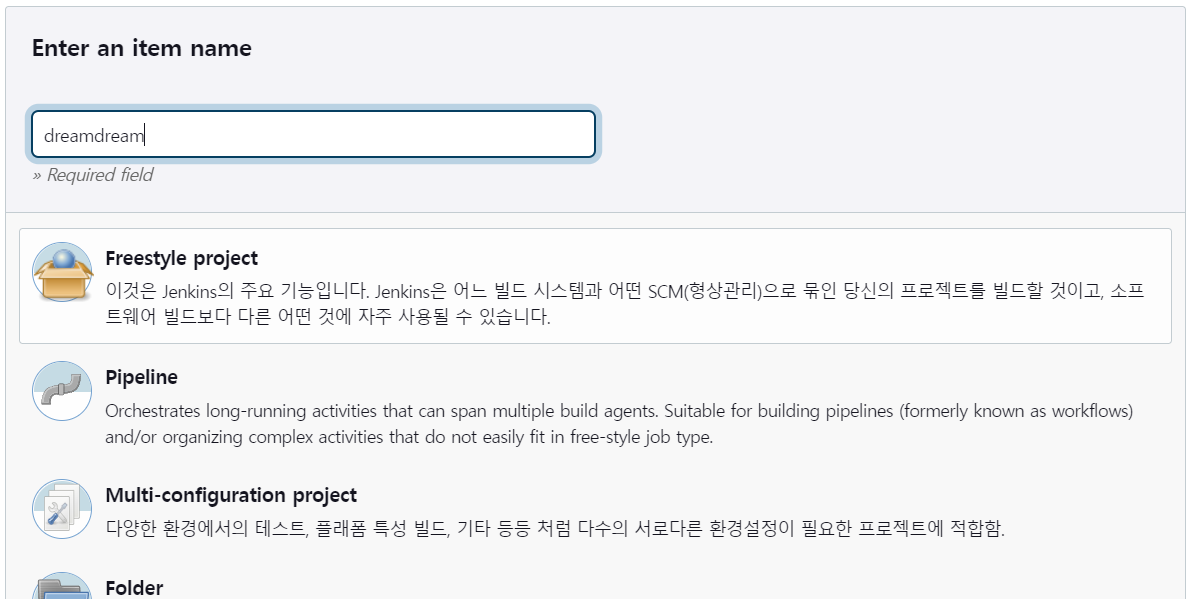
이후 Jenkins 관리 -> 플러그인 매니저에서 gitlab에 관련된 플러그인을 설치합니다.

v. Jenkins & Git lab으로 ci/cd 구축

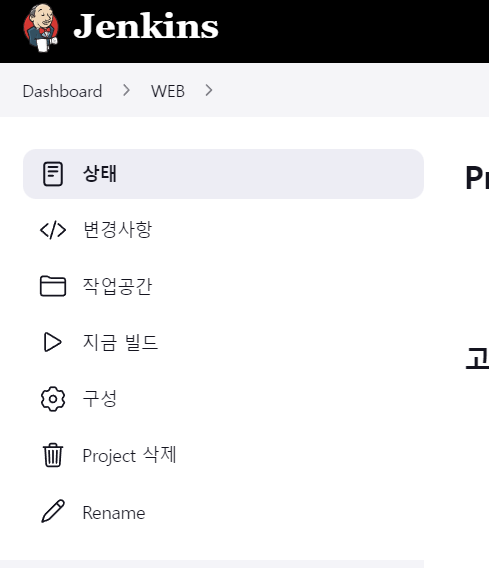
1) Jenkins 설정



젠킨스에 접속한 뒤 새로운 Item을 클릭합니다.

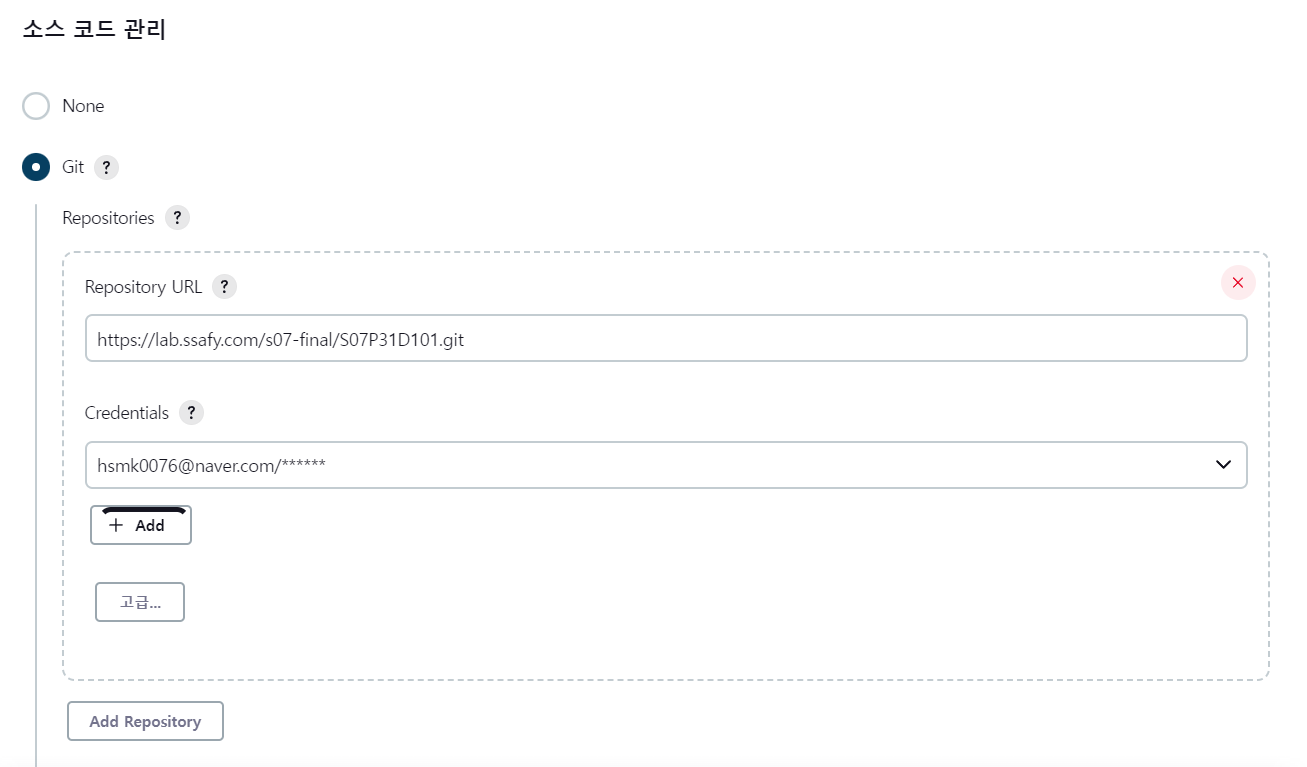


Freestyle project을 클릭하고 해당 아이템의 이름을 정해주고 OK를 누릅니다.



아이템이 만들어지면 해당 아이템을 클릭한 뒤에, 좌측에 있는 구성을 누릅니다.

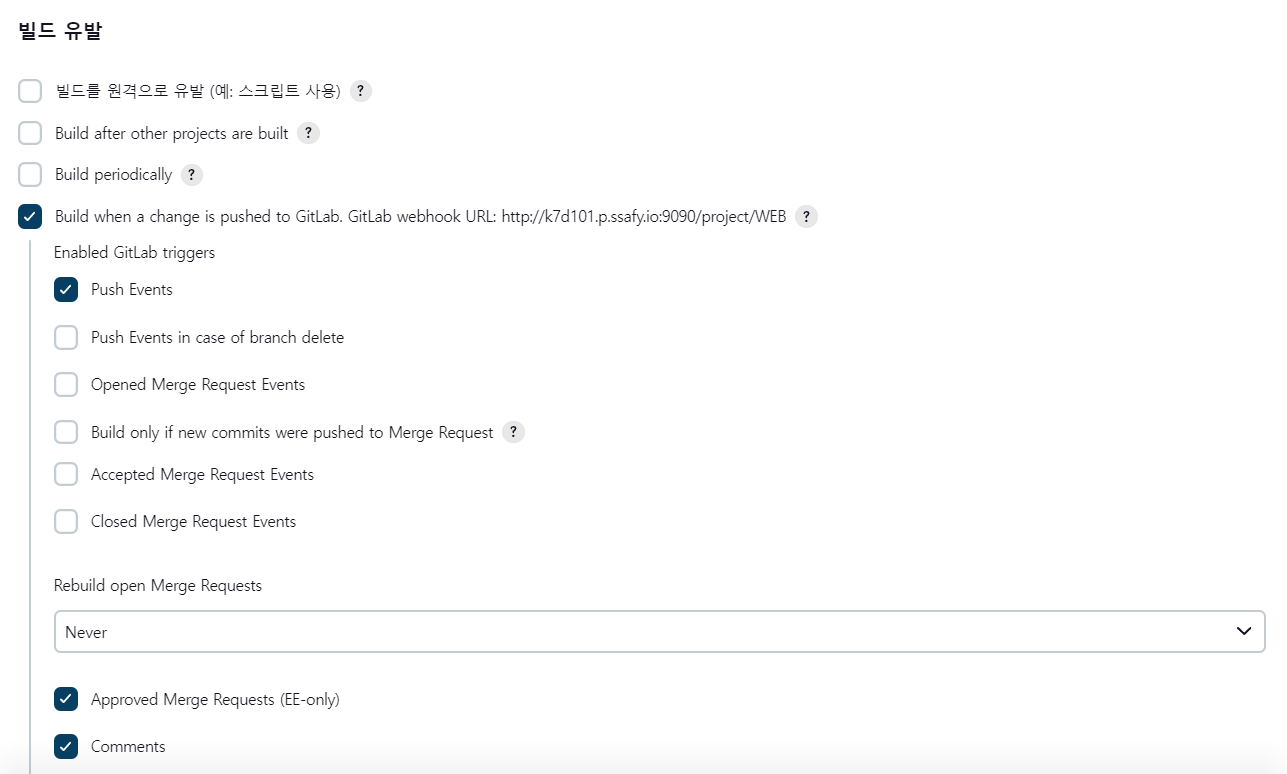
a. 프론트 서버 세팅



소스코드 관리의 git을 눌러서 repository url을 작성하고, Add를 통해 git lab 계정을 등록합니다.

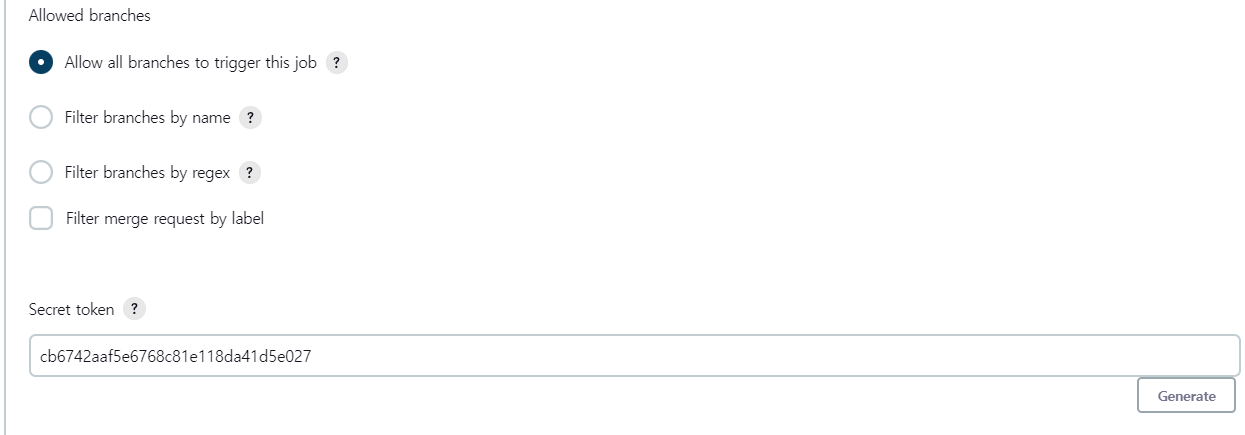


아래의 branches to build에 branch이름을 넣어서 어떤 브랜치를 빌드 할지 정해줍니다.



빌드 유발에서 각 브랜치에 변화가 생겼을 때만 build 하도록 설정합니다

또한 Git lab 연결을 위해 GitLab webhook URL을 저장해둡니다.



다음으로, Git lab 연결을 위해 빌드 유발의 고급 설정에서 Secret token을 Generate 하고 이를 저장해둡니다.

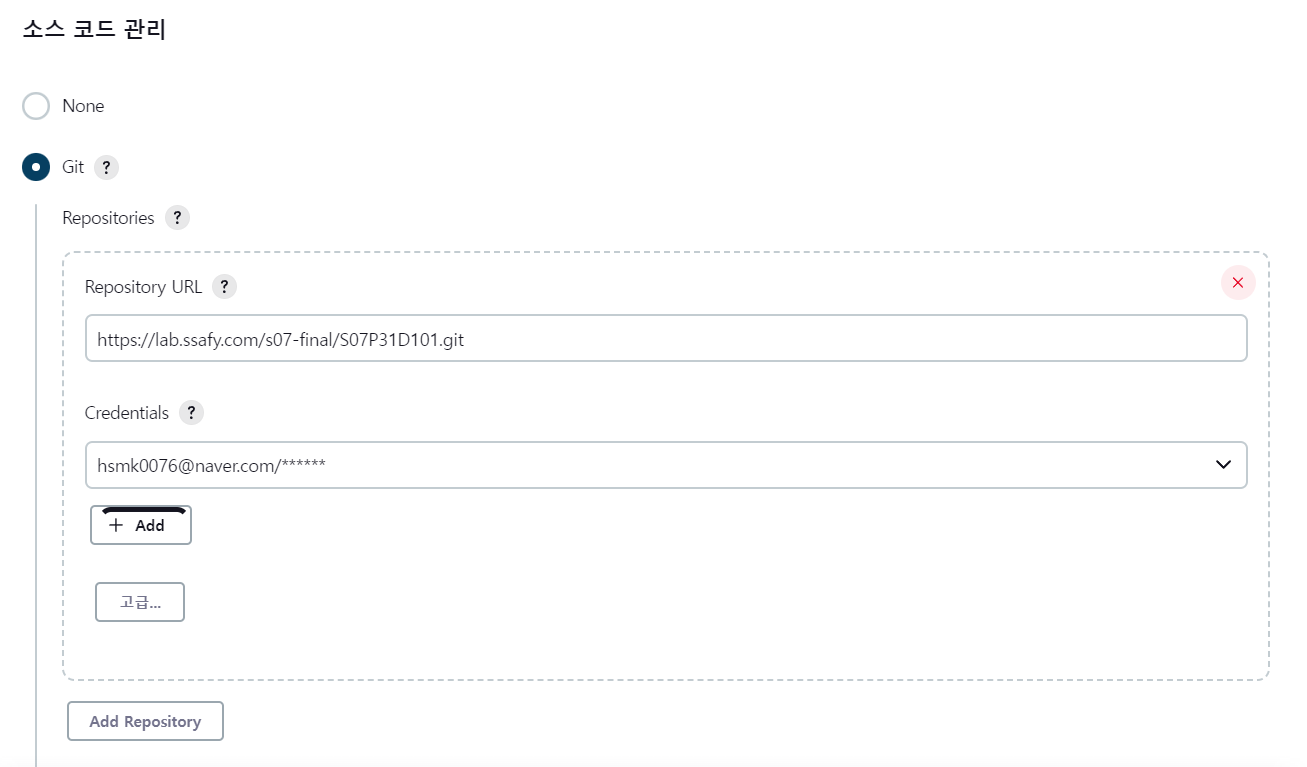


Build steps에는 docker file을 만들어서 해당 도커 이미지를 돌리는 코드를 작성해주면 됩니다.

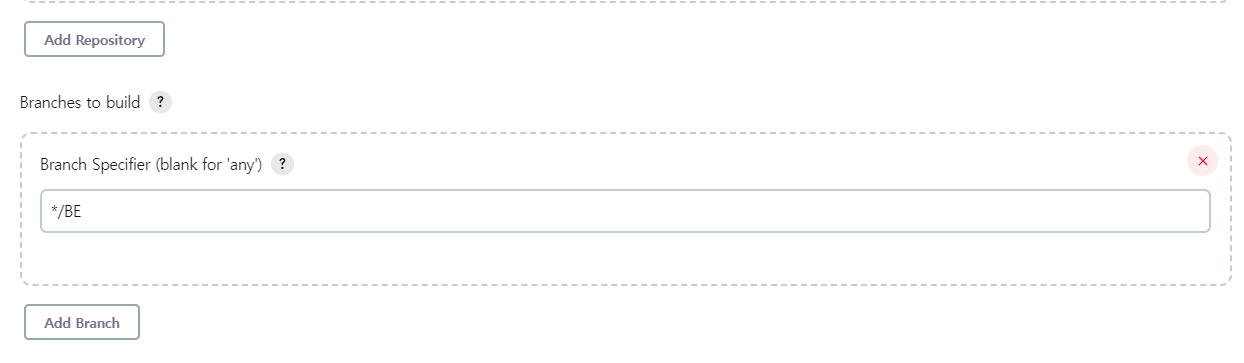
b. 백 서버 세팅

백엔드의 경우에는 Build Step에서 Invoke Gradle script을 통해서 dockerfile에 컴파일 관련 설정을 해주지 않아도 됩니다.

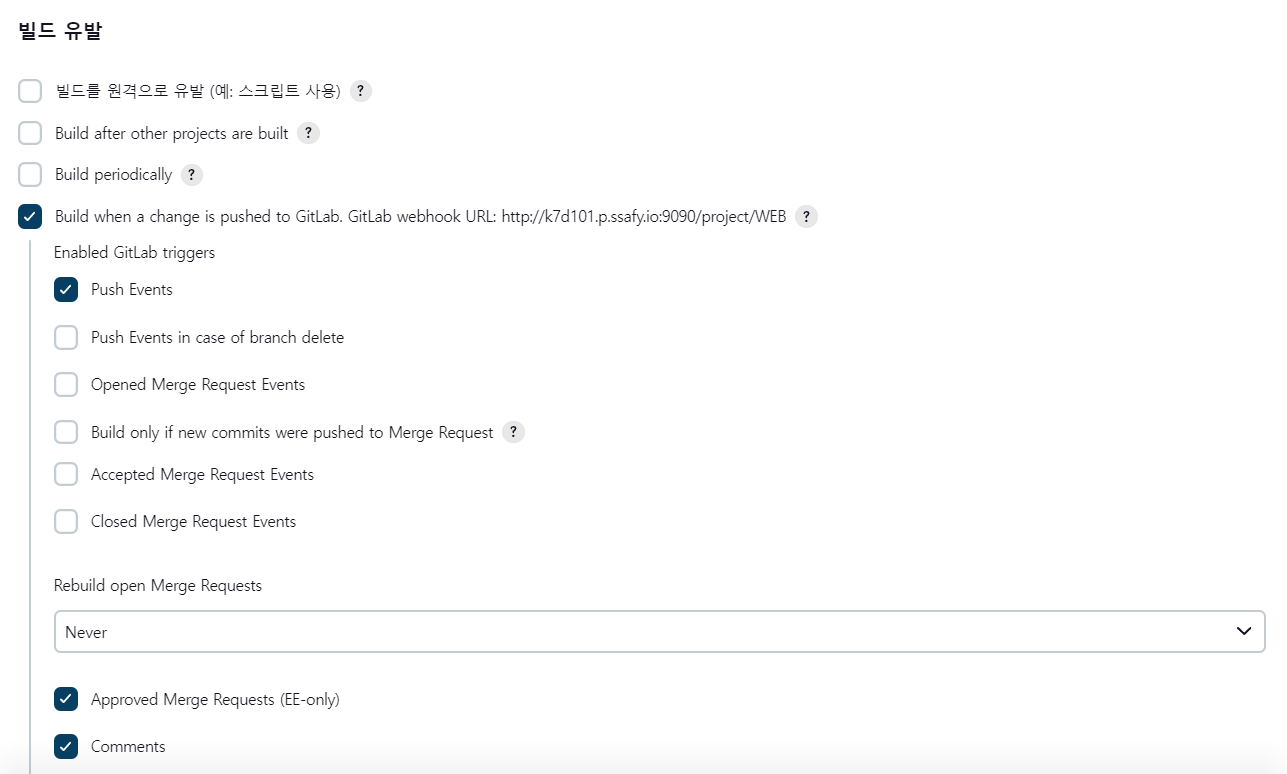
그래서 백엔드는 다음과 같이 코드를 작성하면 됩니다.



소스코드 관리의 git을 눌러서 repository url을 작성하고, Add를 통해 git lab 계정을 등록합니다.

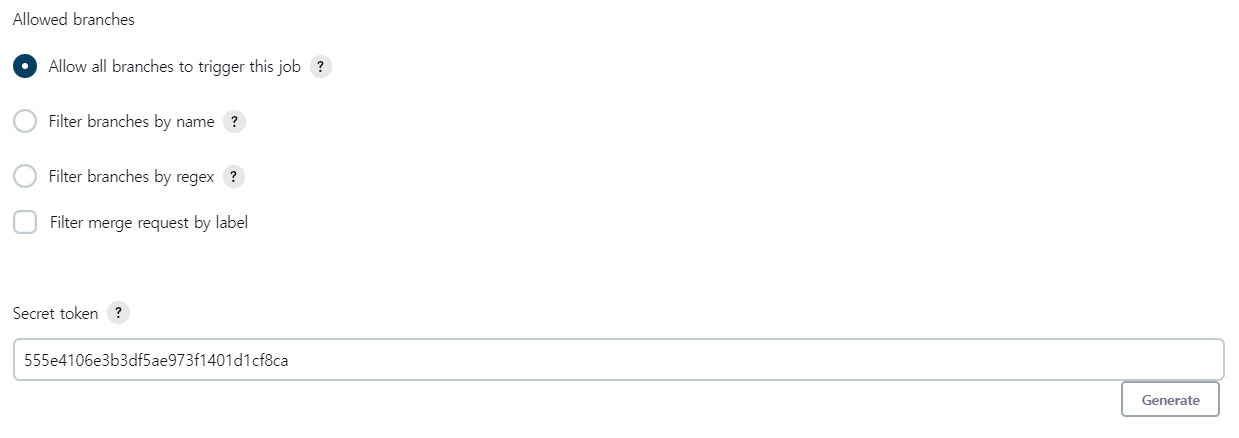


아래의 branches to build에 branch이름을 넣어서 어떤 브랜치를 빌드 할지 정해줍니다.



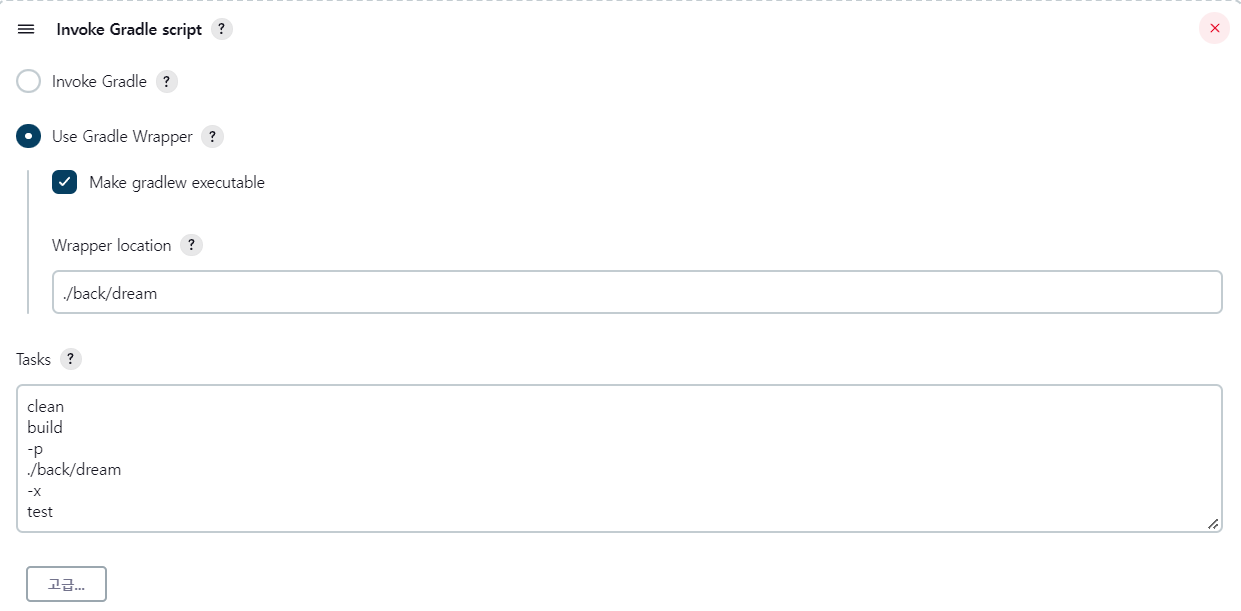
빌드 유발에서 각 브랜치에 변화가 생겼을 때만 build 하도록 설정합니다

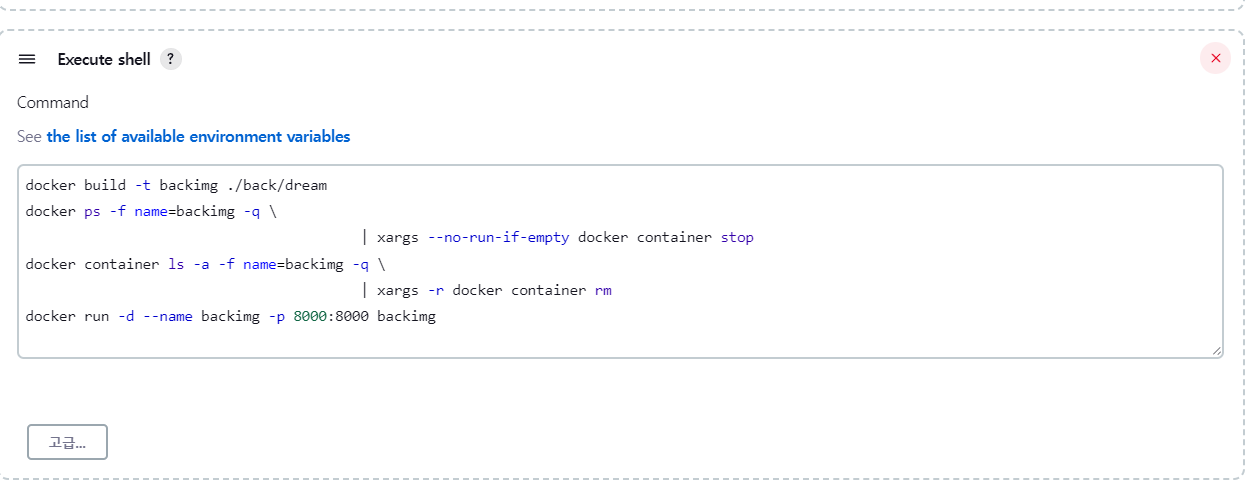
또한 Git lab 연결을 위해 GitLab webhook URL을 저장해둡니다.



다음으로, Git lab 연결을 위해 빌드 유발의 고급 설정에서 Secret token을 Generate 하고 이를 저장해둡니다.

Build steps에는 docker file을 만들어서 해당 도커 이미지를 돌리는 코드를 작성해주면 됩니다.





2) Git lab 설정

Key 파일 저장 위치를 묻는 내용으로 디폴트로 정해준 위치에 저장할거면 그냥 엔터⏎!  
        만약 위치를 변경하려면 원하는 위치 경로를 입력하면 된다.

우선 접속을 위해 SSH Public Key 라는 인증키를 아래 명령어로 생성합니다. key의 비밀번호는 생략하거나 입력하고, 입력하였다면 이를 기억해둡니다

ssh-keygen -t rsa -C "hsmk0076@naver.com"

다음으로, ~/.ssh 위치에 id\_rsa, id\_rsa.pub로 저장된 SSH Key를 사용자만이 읽기, 쓰기, 실행이 가능하도록 아래 명령어를 통해 권한을 부여합니다

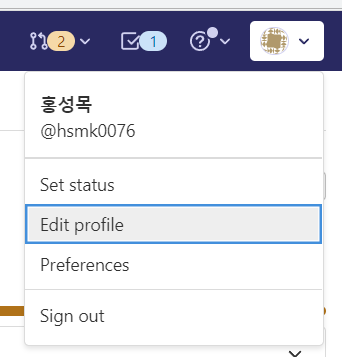
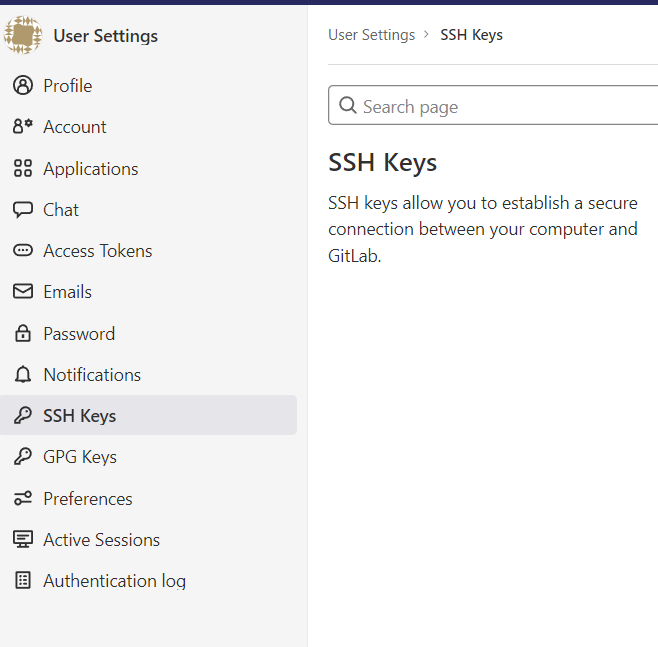
chmod 700 ~/.ssh/id\_rsa\*

다음으로, Key를 사용하기 위해 아래 명령어로 key값을 출력합니다.

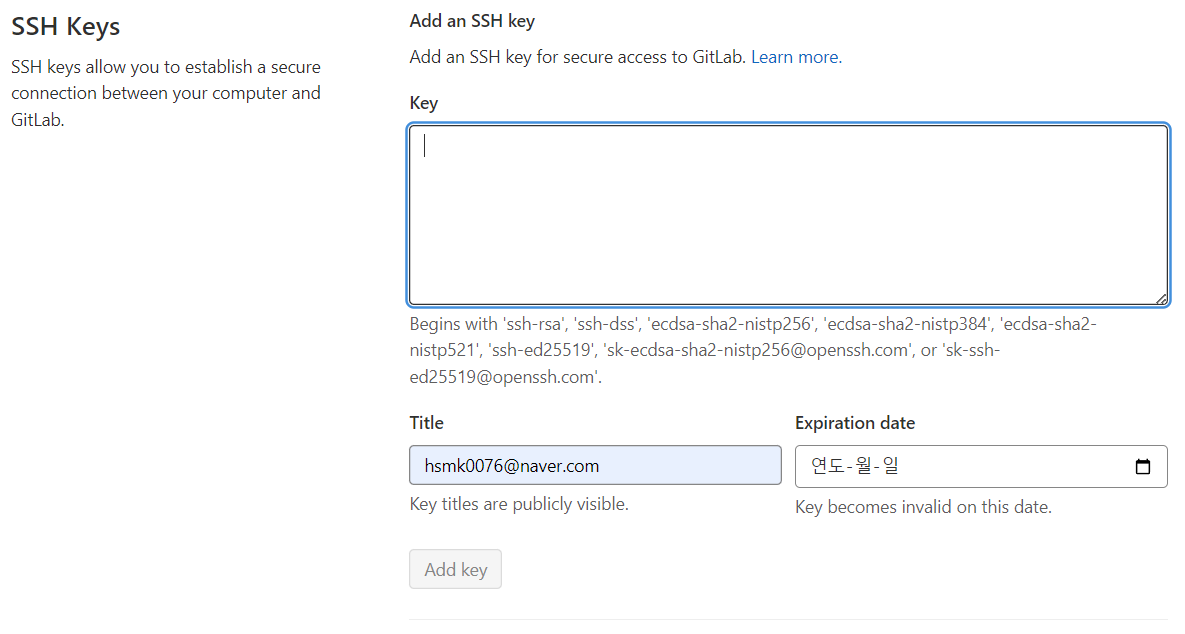
cat ~/.ssh/id\_rsa.pub

그 후 콘솔에 출력되는 모든 값이 Key 값이기 때문에, 이를 전부 저장해둡니다.

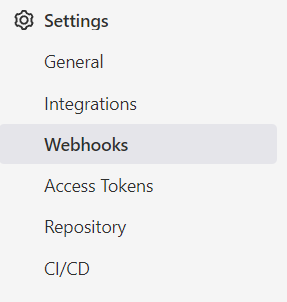
다음으로 Git lab에 접속합니다

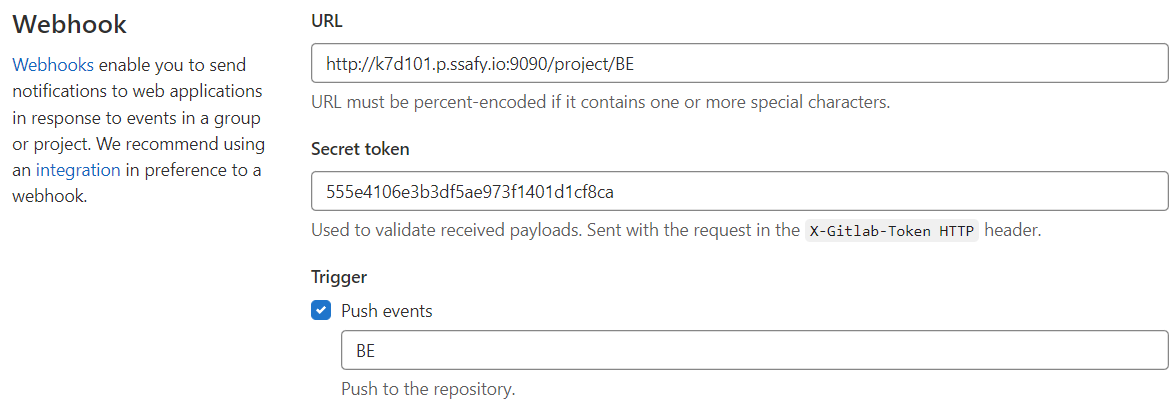
Edit profile – SSH keys에 접속합니다



미리 복사해둔 Key 값을 입력합니다. Title은 자동입력이지만 수정 가능하며, 만료일은 설정하지 않을 경우 무기한 사용 가능으로 설정됩니다.



다음으로 배포할 프로젝트의 Settings – webhooks로 접속합니다.





사전에 복사해둔 Url과 Secret token을 입력하고, 배포를 원하는 브랜치를 설정해둡니다.

해당 브랜치에 변화가 있을 경우에만 자동 빌드하기 위해서 Push events만 체크해둡니다.

또한 SSL을 사용할 것이기 때문에 SSL verification을 체크해둡니다.

프론트 웹훅과 백 웹훅을 각각 생성합니다.

vi. Nginx 설치 및 ssl 적용

Let's Encrypt의 certbot 서비스를 통해 무료로 인증서를 발급받아 적용합니다.

- sudo apt-get install nginx를 입력해 nginx를 설치합니다

- sudo service nginx stop을 입력해 nginx를 정지합니다.

- sudo apt install python3-certbot-nginx 을 입력해서 certbot을 설치합니다.

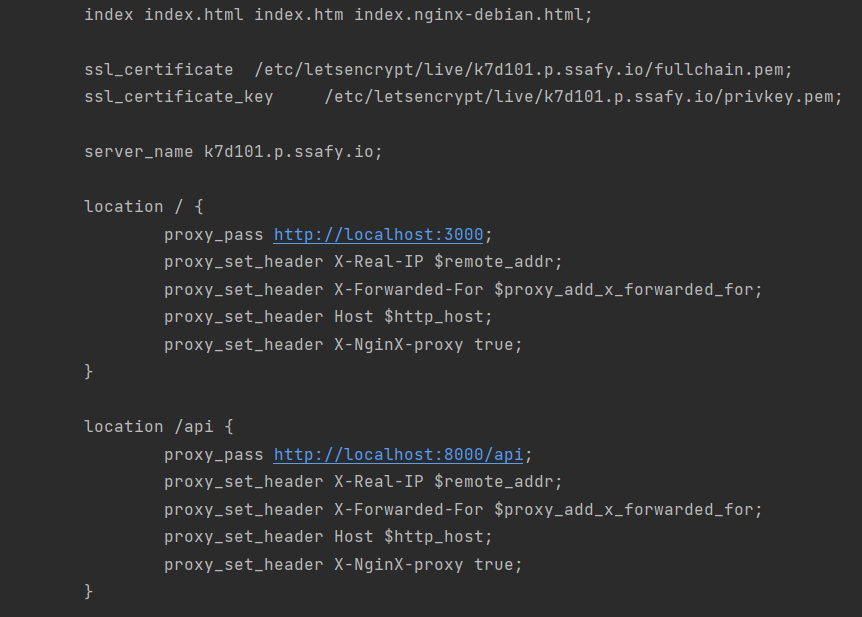
- sudo certbot certonly --nginx 를 입력해서 인증서를 발급받습니다. 인증받은 키는 nginx의 default파일에 설정을 해주어야 합니다. 질문에 대해 차례대로 A, N, 2를 입력하고 도메인은 사용중인 도메인을 입력합니다.

- sudo service nginx start을 입력해 nginx를 실행시킵니다.

- sudo vi /etc/nginx/sites-available/default 를 입력해서 default 설정을 수정합니다.



-> http 접속을 강제로 https로 리다이렉트



위와 같이 설정을 해줍니다. location /api에 있는 8000포트는 백엔드 서버가 돌아가는 포트번호이고, /은 프론트입니다.

설정을 끝내고 나면 sudo service nginx restart를 입력해 nginx를 재시작합니다.

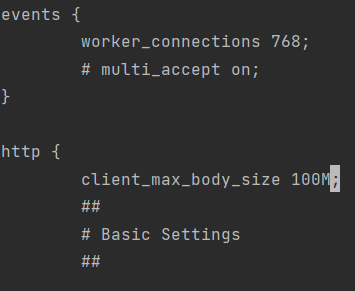
vii. Docker Volume 생성 및 연결

docker volume create dream 으로 볼륨을 생성합니다

docker volume inspect dream 으로 상세 정보를 확인합니다

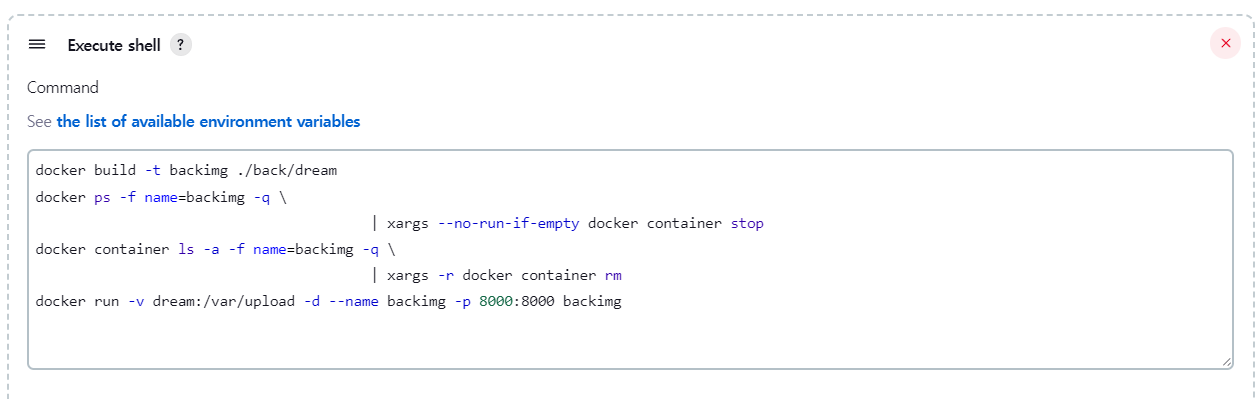
데이터 통신을 위해 sudo vim /etc/nginx/nginx.conf 를 수정합니다

client\_max\_body\_size 100M;



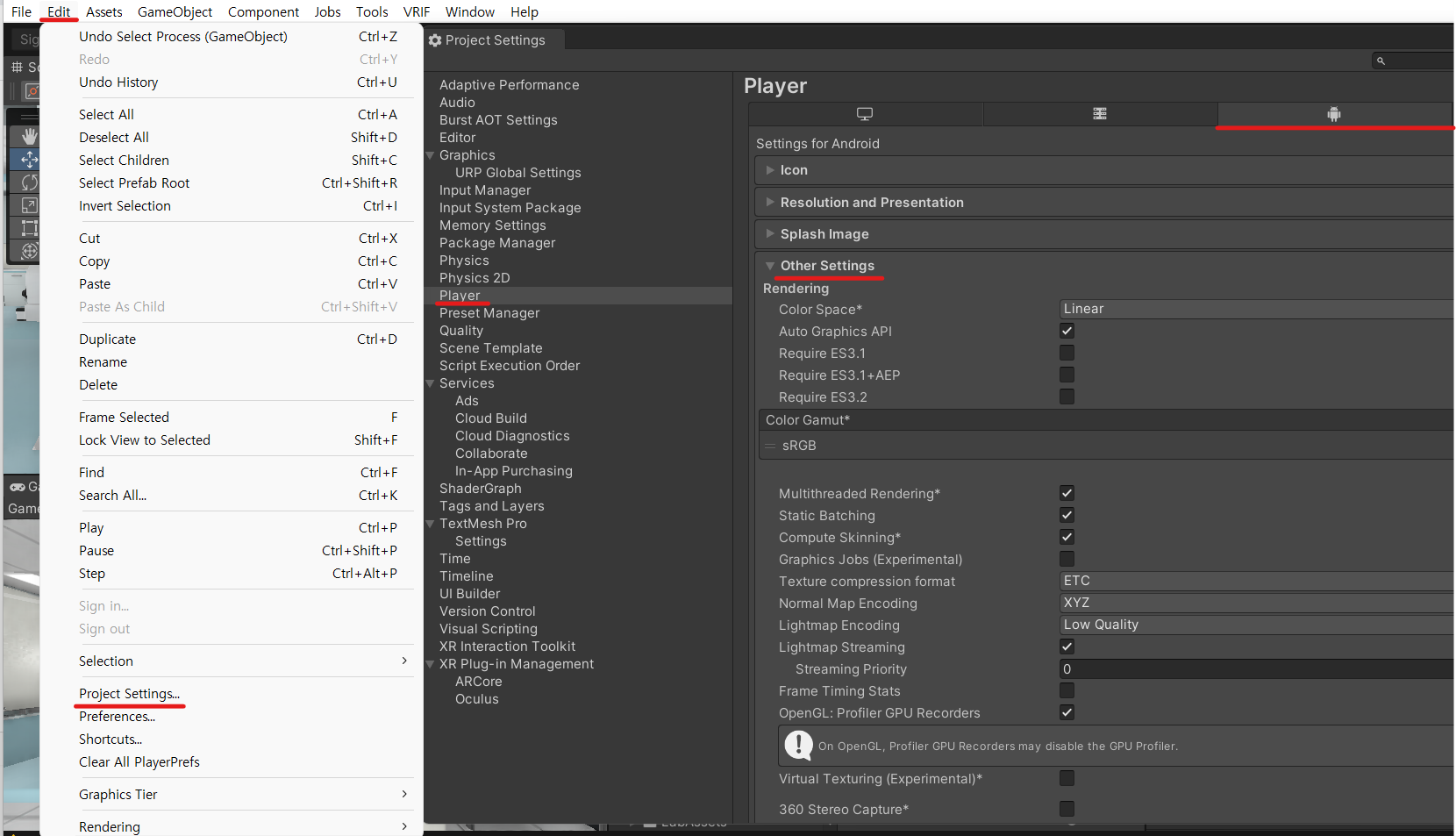
sudo service nginx restart

다음으로, BE 배포 시 볼륨이 연결되도록 jenkins를 수정합니다

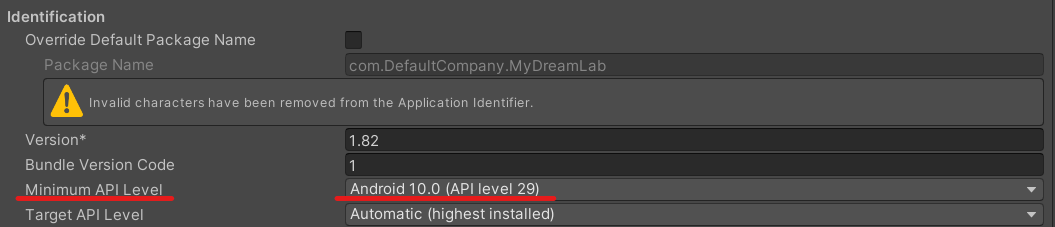


이 이후로 backend 서버의 var/upload와 외부의 dream 볼륨이 항상 연결됩니다.

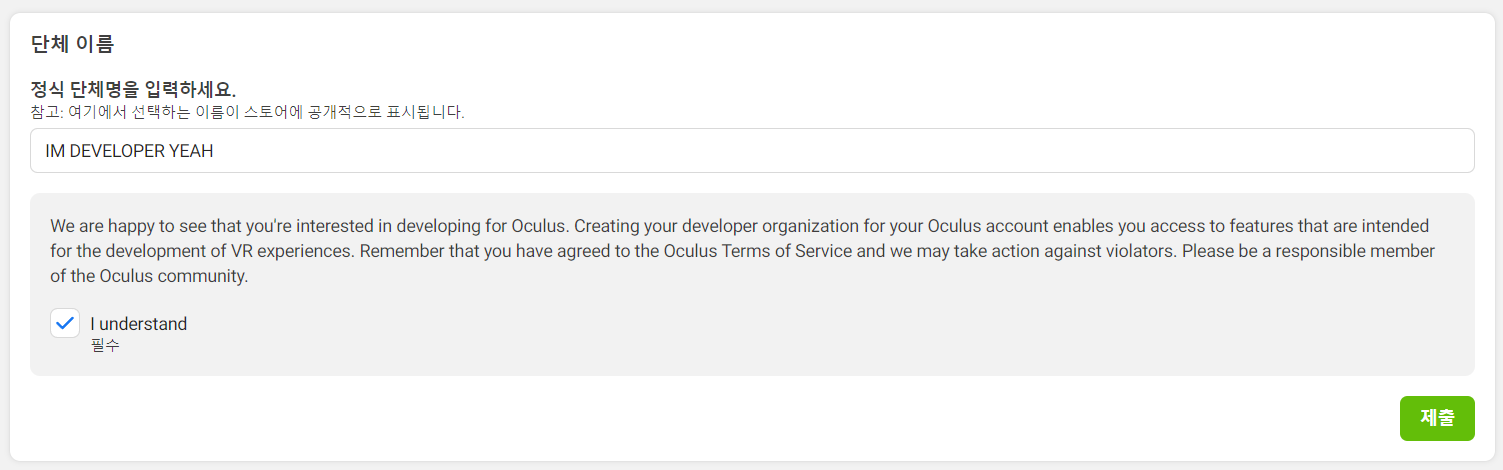
B. 빌드한 APK파일 오큘러스에 설치



유니티 프로젝트에서 상단 메뉴 File - Project Settings - Player - 안드로이드 아이콘 - Other Settings 에 들어간다.



Other Settings 탭에서 Minimum API Level 을 알맞게 설정한다. (오큘러스 퀘스트2의 안드로이드 버전은 10 이다.)

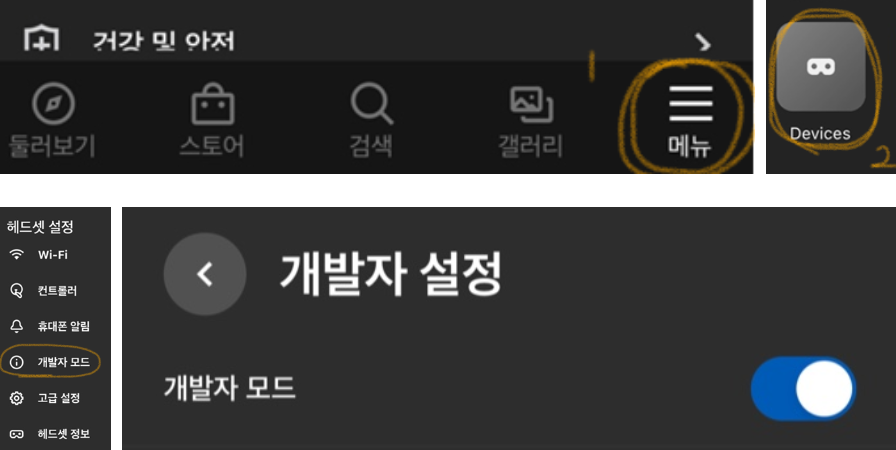


<https://developer.oculus.com/documentation/native/android/mobile-device-setup/#joining-or-creating-an-organization>

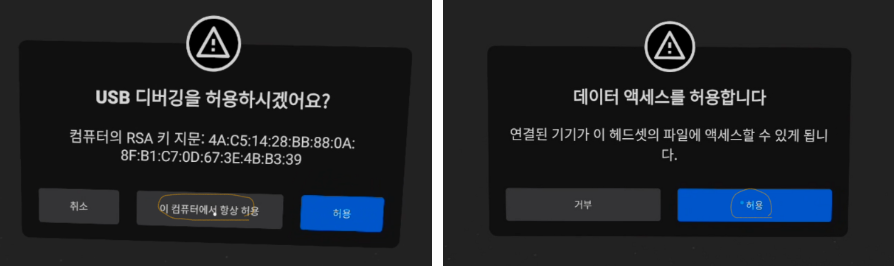
오큘러스로 플레이 하기 전 개발자 등록이 필요하다. 해당 링크로 들어가서 가이드에 따라 등록하면 된다.



스마트폰에 해당 어플을 다운 받는다.

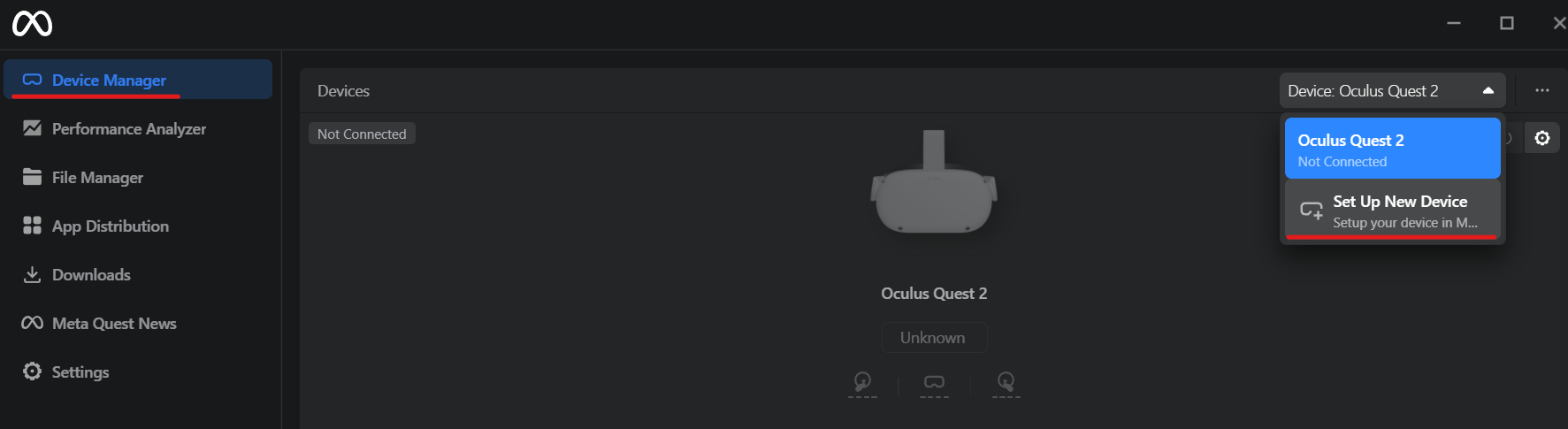


헤드셋 설정에서 개발자 모드를 활성화 하면



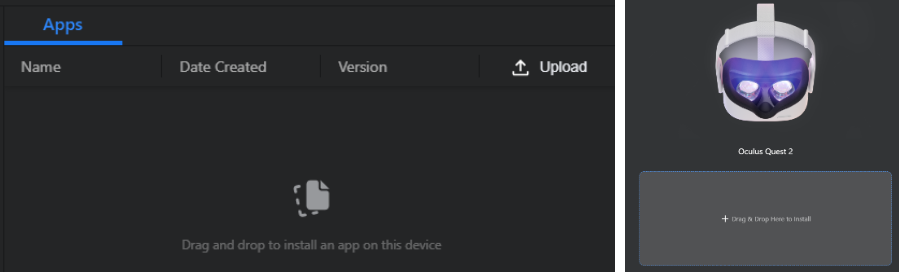
오큘러스에서는 다음과 같은 UI가 뜬다. 둘 다 허용을 눌러준다.

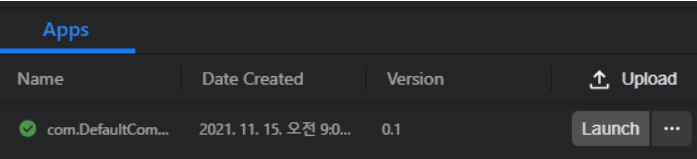




<https://developer.oculus.com/downloads/package/oculus-developer-hub-win/>

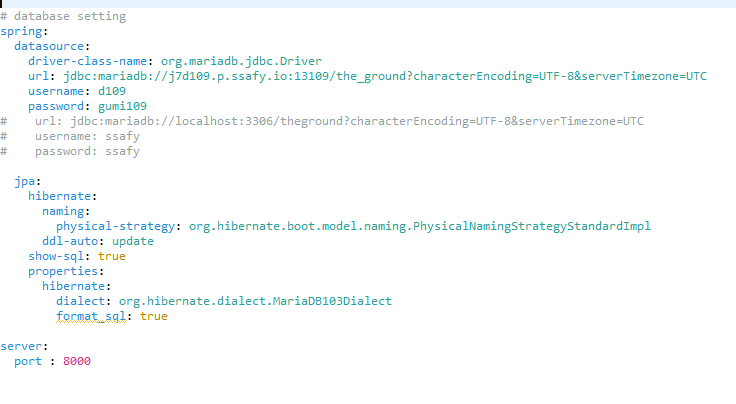
다운로드 후 로그인을 하여 Device Manager 탭에서 유선으로 연결된 기기를 등록한다.





오른쪽 위에 있는 Apps 구역에 사전에 빌드한 apk 파일을 드래그 하여 넣어 설치하고 Launch를 눌러 실행하거나 오큘러스 내에서 앱라이브러리에서 ‘알 수 없는 출처’ 를 클릭하면 설치한 앱을 확인할 수 있다.

1. 프로퍼티 정의 파일 목록
   1. 백엔드
      1. Application.yml



Application.yml에는 database에 접속하기 위한 주소,정보가 있습니다.