**Piece Of Cake 포팅 매뉴얼**

* **Docker 설치**

$ sudo apt-get update // 의존성 패키지 설치

$ sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common // Docker 패키지 인증 키 추가

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

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

$ sudo apt-get update

$ sudo service docker start

$ sudo apt-cache search docker-ce // 도커 패키지 확인

$ sudo apt-get install docker-ce docker-ce-cli containerd.io // 도커 설치

$ sudo usermod -aG docker $USER // 사용자를 도커 그룹에 포함

* **Jenkins Container 설치**

$ sudo docker pull jenkins/jenkins:lts // jenkins 이미지 준비

$ sudo docker run --name jenkins-docker -d -p 8080:8080 -p 50000:50000 -v /home/jenkins:/var/jenkins\_home -u root jenkins/jenkins:lts 젠킨슨 빌드 구성시 command

cd BackEnd/common\_pjt

ls

chmod +x gradlew

./gradlew bootRun

* **MySQL Container 설치**

$ sudo docker pull mysql/mysql:lts // docker 이미지 준비

$ docker run --name mysql-commonpjt -e MYSQL\_ROOT\_PASSWORD=zeitfox -d -p 13306:3306 mysql:latest

$ docker exec -it mysql-commonpjt // 도커 컨테이너 접속

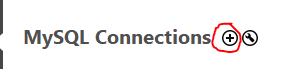
$ create user 'ssafy'@'%' identified by 'ssafy'; // 사용자 추가

GRANT ALL PRIVILEGES ON \*.\* TO ‘root’@’%’;

GRANT ALL PRIVILEGES ON \*.\* TO ‘ssafy’@’%’;

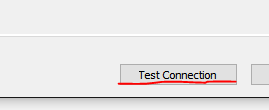
Flush privileges; // 권한 부여

MYSQL workbench 설정



텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

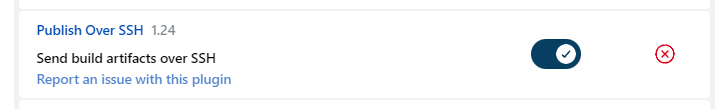
자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

이후 OK를 누르면 접속설정완료

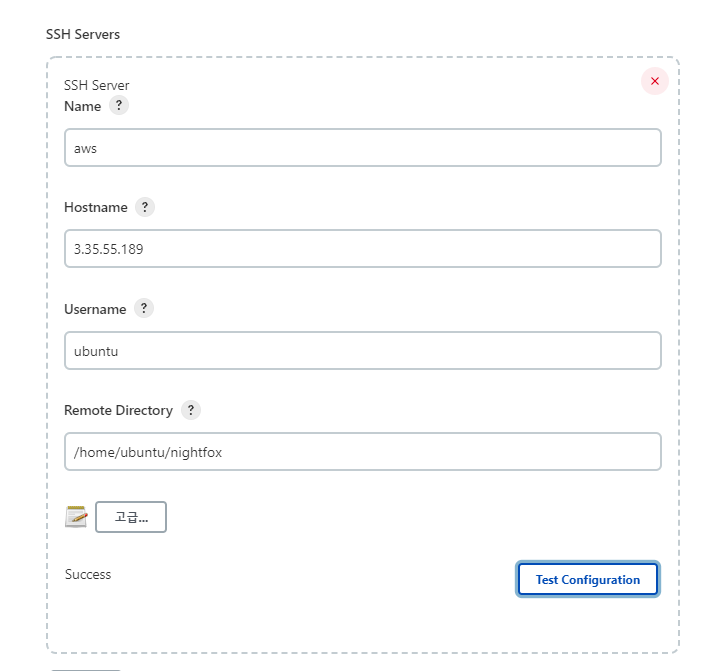
* **배포 자동화**

1. Jenkins > Dashboard > Plugin Manager에서 해당 플러그인 설치

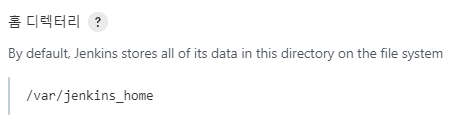


1. Jenkins > DashBoard > Configure System에서 최하단에 위치한 Publish OverSSH 관련 셋팅

원격 배포 서버에서 어떤 디렉터리를 root로 접근할지 명시해주고 Test Configuration을 통해 통신이 잘 되는지 확인한다.



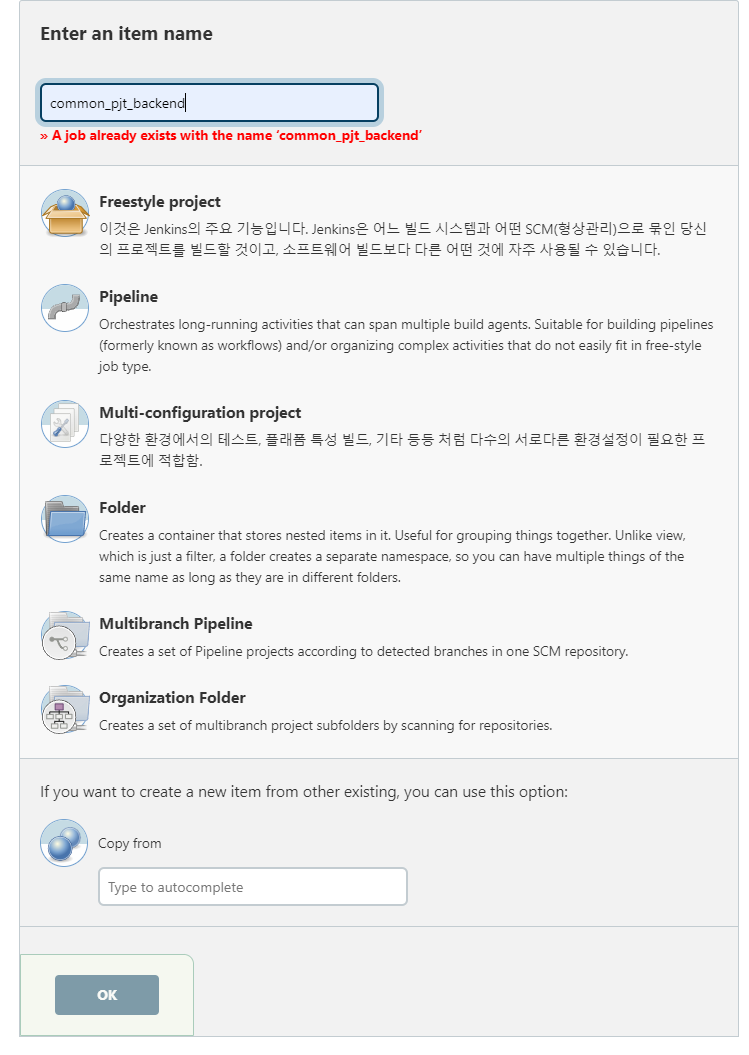
홈 디렉터리 경로를 확인한다.



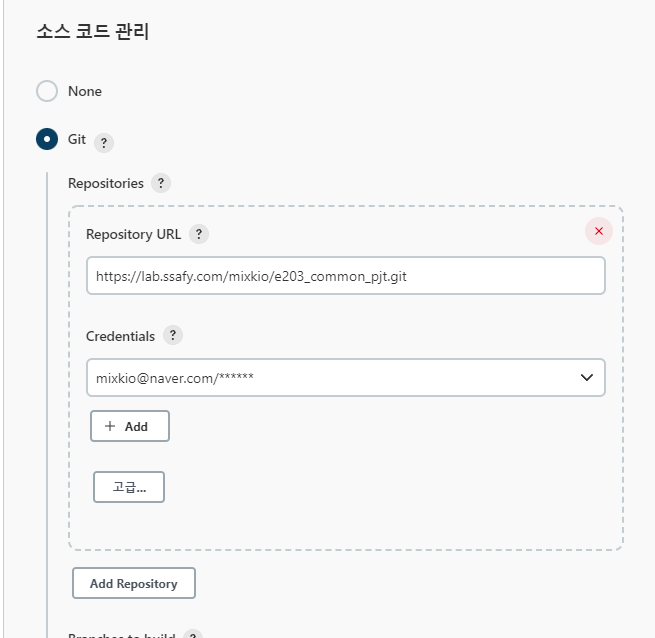
새로운 ITEM 클릭



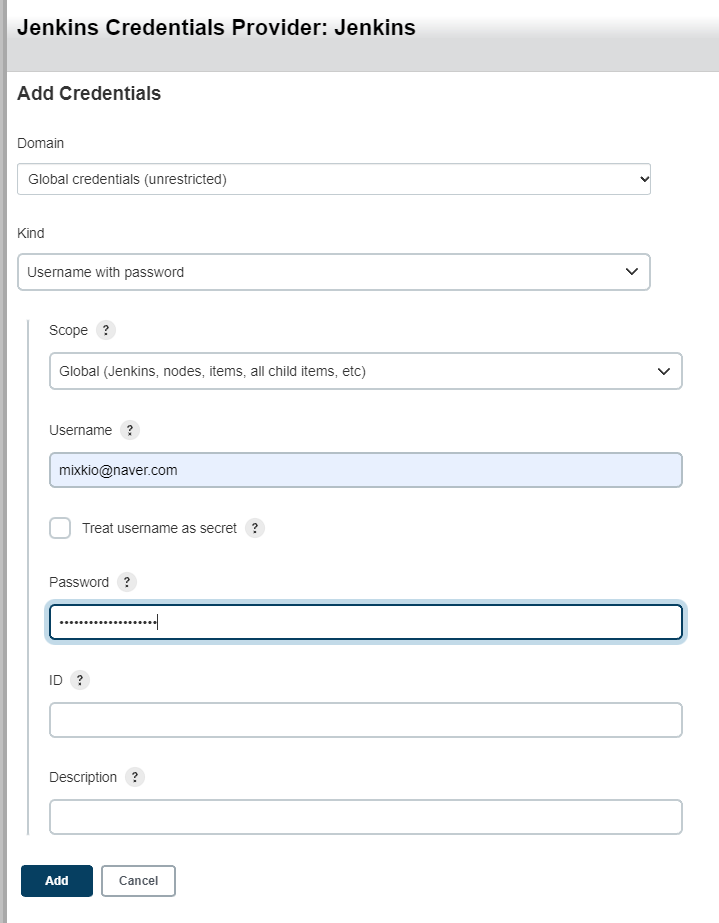
ITEM NAME 입력 후 FRRESTYLE PROJECT 클릭 후 생성



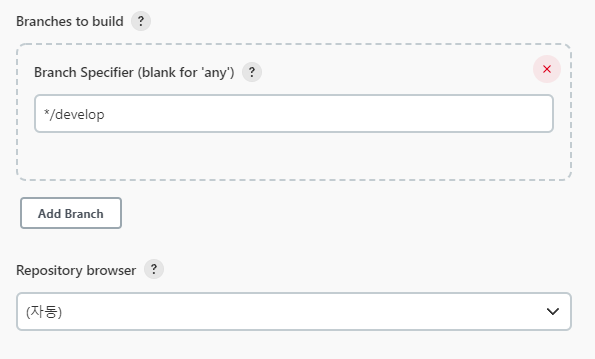
소스 코드 관리에서 Git 선택 후 배포할 Repository URL을 입력



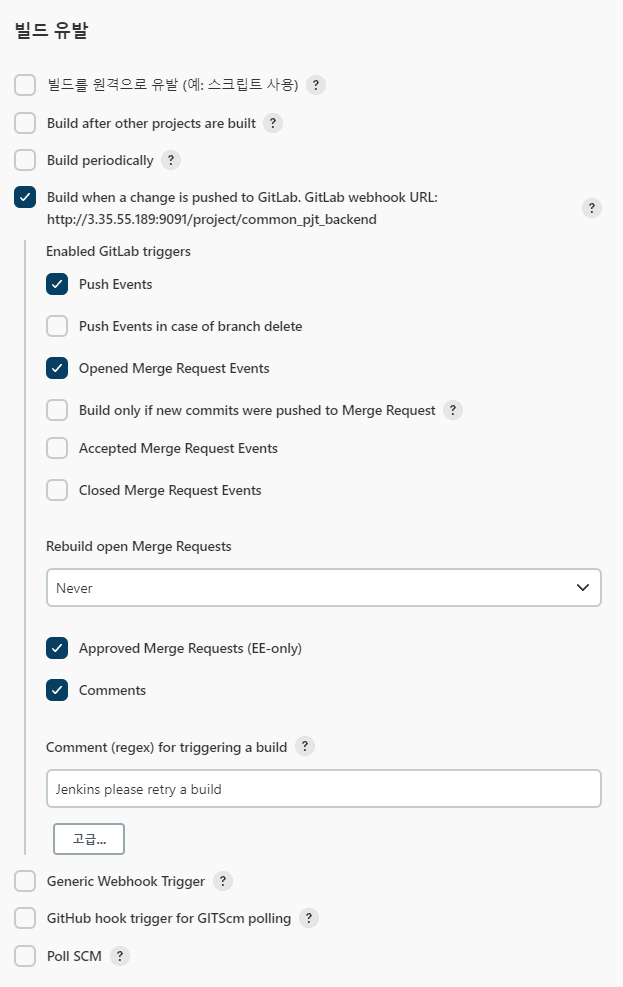
Credentials에 Add를 클릭 후 GitLab에 접속 가능한 사용자 계정명 및 토큰이나 비밀번호를 입력



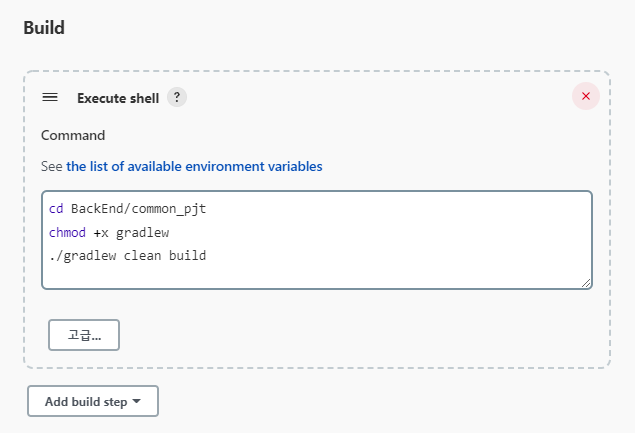
어떤 브랜치를 기준으로 빌드를 실행할지 결정



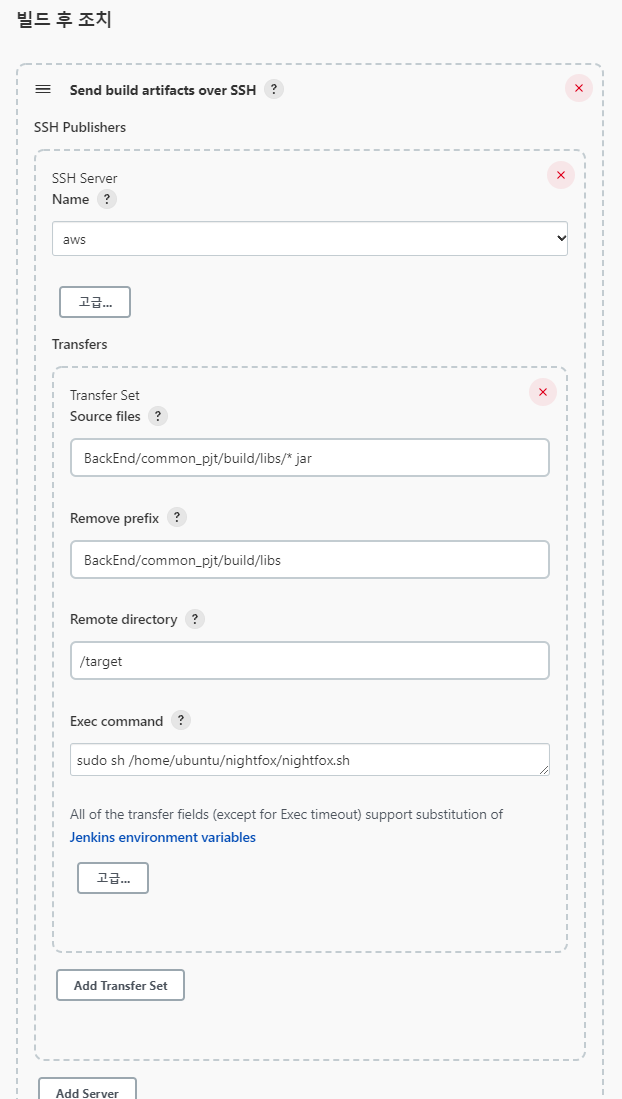
빌드에 관한 추가 사항 체크



Add build step을 클릭 후 Execute shell 선택



빌드가 완료된 파일을 배포 서버에 전송해주고 해당 명령어 파일을 실행하여 배포 서버에서 서버를 실행하여 준다.



Jenkins에서 빌드 된 파일 / 배포 서버로 전송된 파일





기존에 실행되고 있는 서버의 PID를 찾아 종료한 뒤 백그라운드 .jar 파일로 서버를 실행시키며 /logs 경로에 log 파일을 남긴다

/home/ubuntu/nightfox/nightfox.sh

echo "PID Check..."

CURRENT\_PID=$(ps -ef | grep java | grep common\_pjt\* | awk '{print $2}')

echo "Running PID:{$CURRENT\_PID}"

if "$CURRENT\_PID" [ -z CURRENT\_PID ];then

echo "Project is not running" <200b>

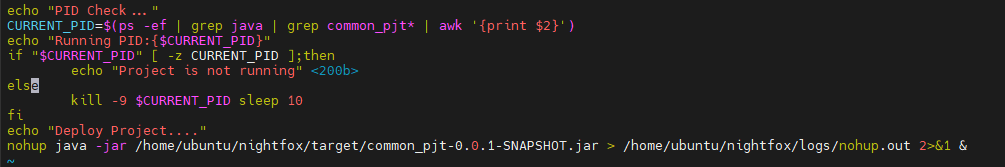
else

kill -9 $CURRENT\_PID sleep 10

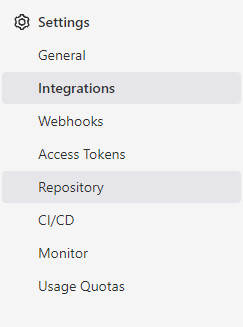
fi

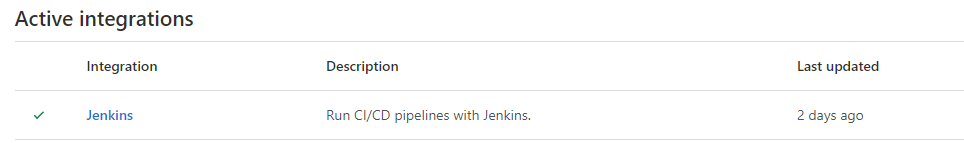
echo "Deploy Project...."

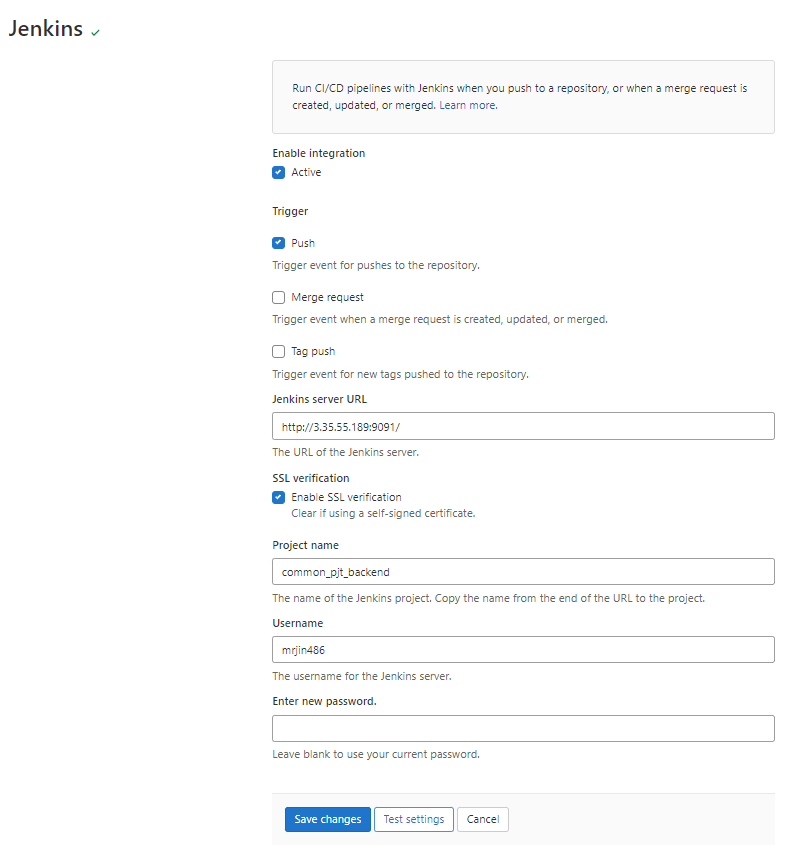
nohup java -jar /home/ubuntu/nightfox/target/common\_pjt-0.0.1-SNAPSHOT.jar > /home/ubuntu/nightfox/logs/nohup.out 2>&1 &



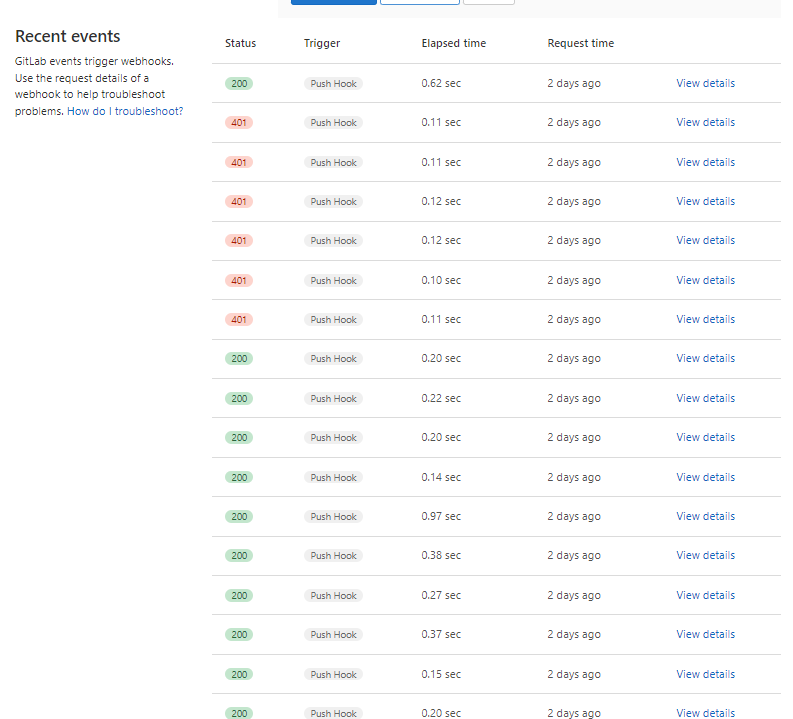
GitLab Jenkins 활성화

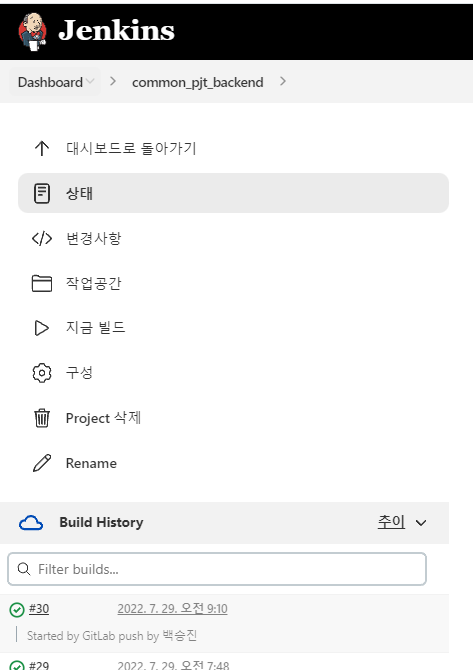






다음과 같이 push 이벤트가 발생되면 Jenkins으로 Trigger를 전송하고 자동 빌드 및 배포가 된다





**AWS http → https Redirect**

**Apt 최신화**

sudo apt-get update

sudo apt-get install software-properties-common

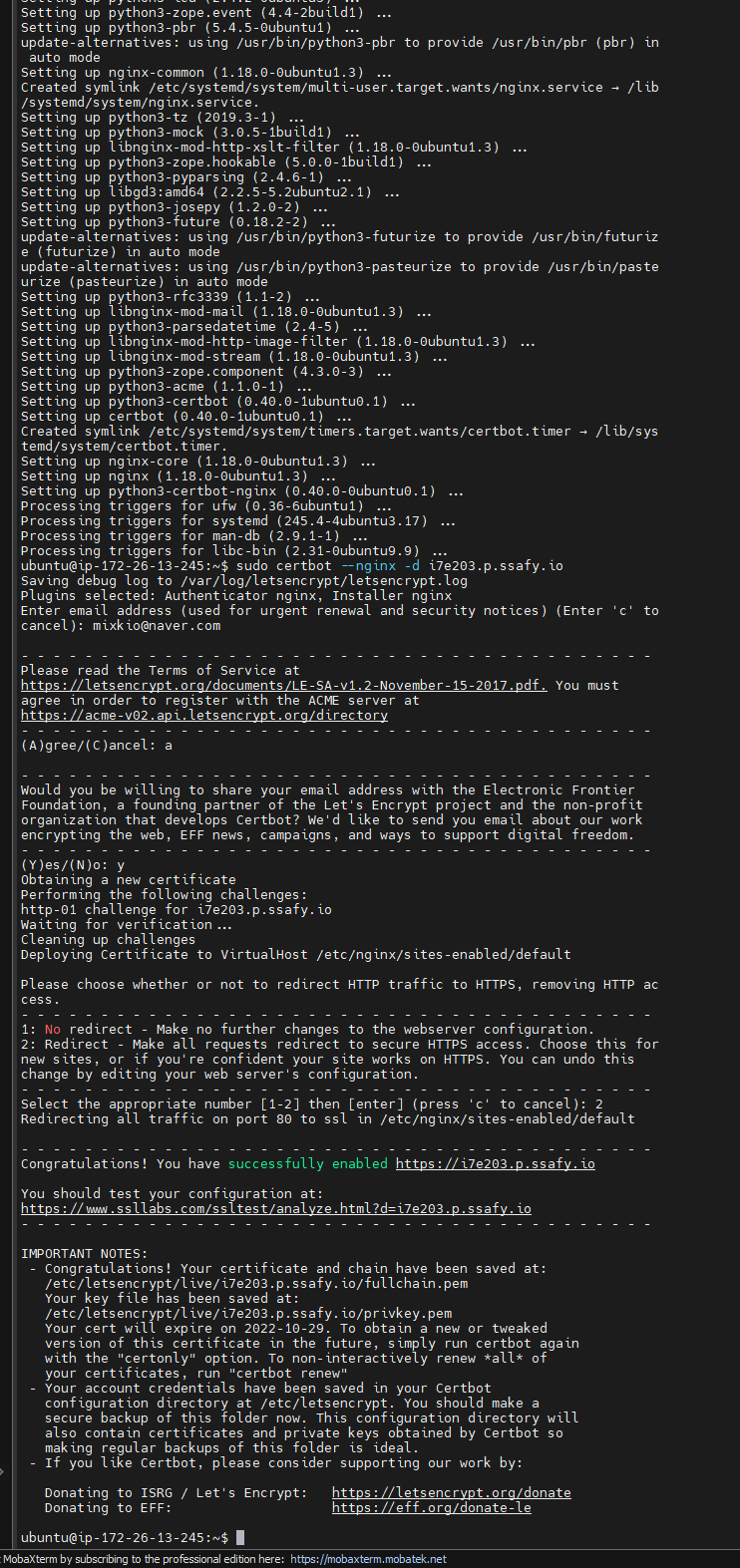
**certbot 설치 (무료 SSH 발급)**

sudo apt-get install certbot python3-certbot-nginx

**certbot을 이용하여 https 적용**

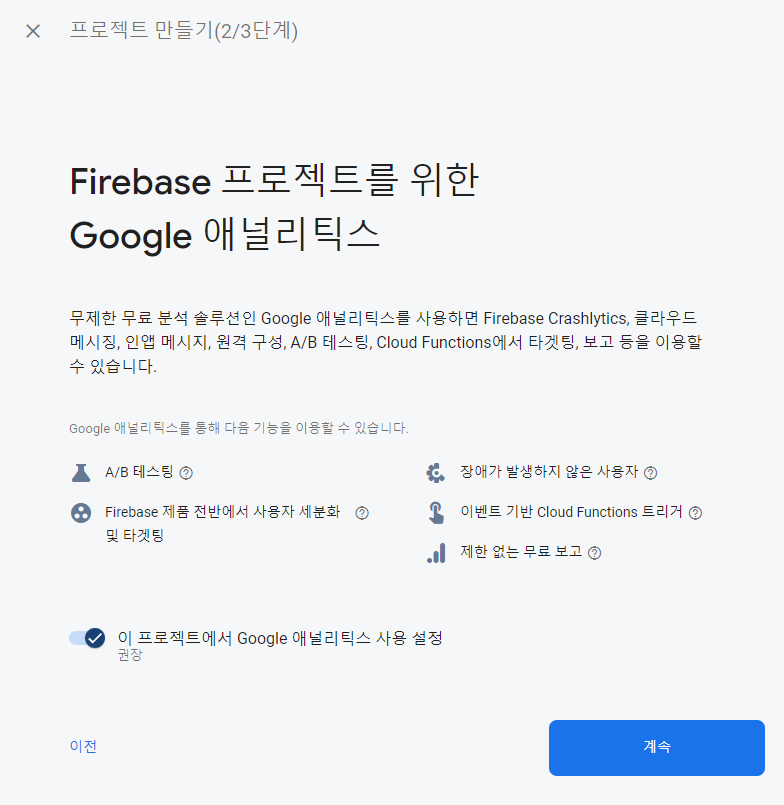
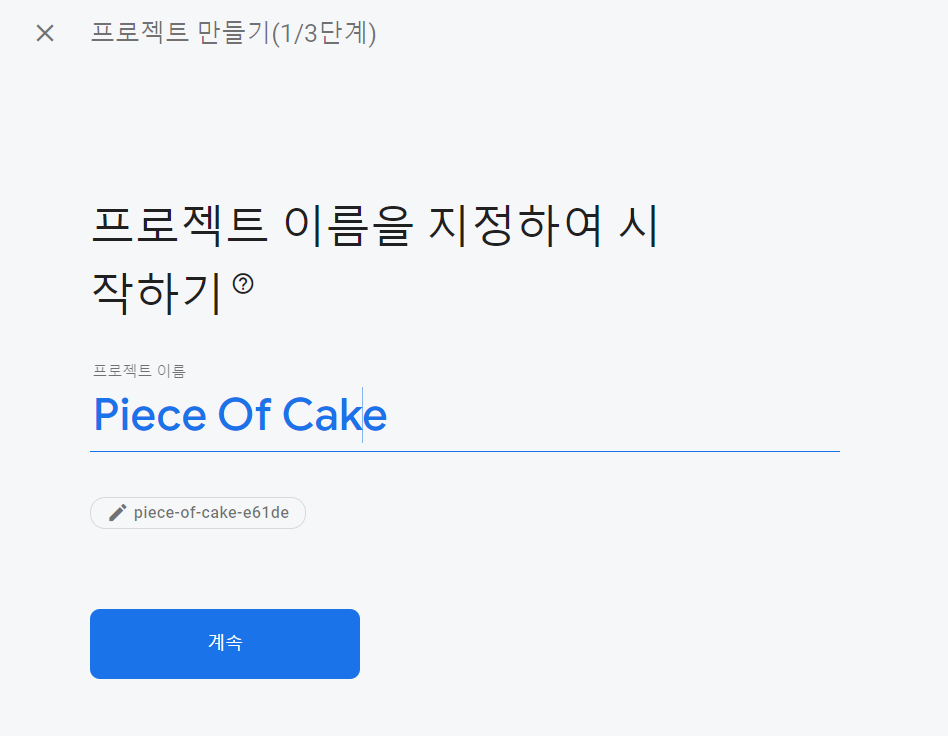
sudo certbot --nginx -d exampledomain

nginx 설치 후 2번 옵션을 적용하여 모든 인풋을 https로 redirect되게 지정



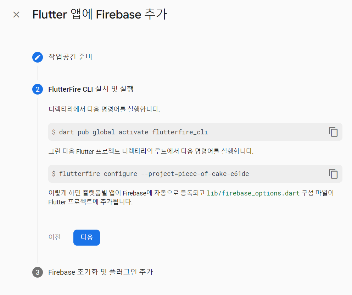
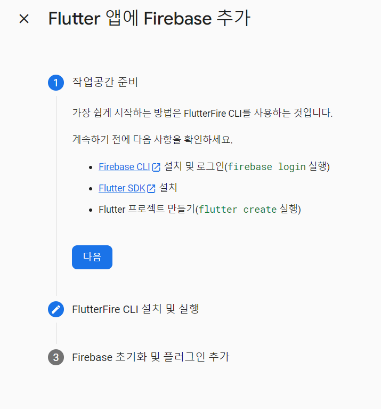
**Firebase**

1. Firebase에 접속하여 프로젝트 만들기

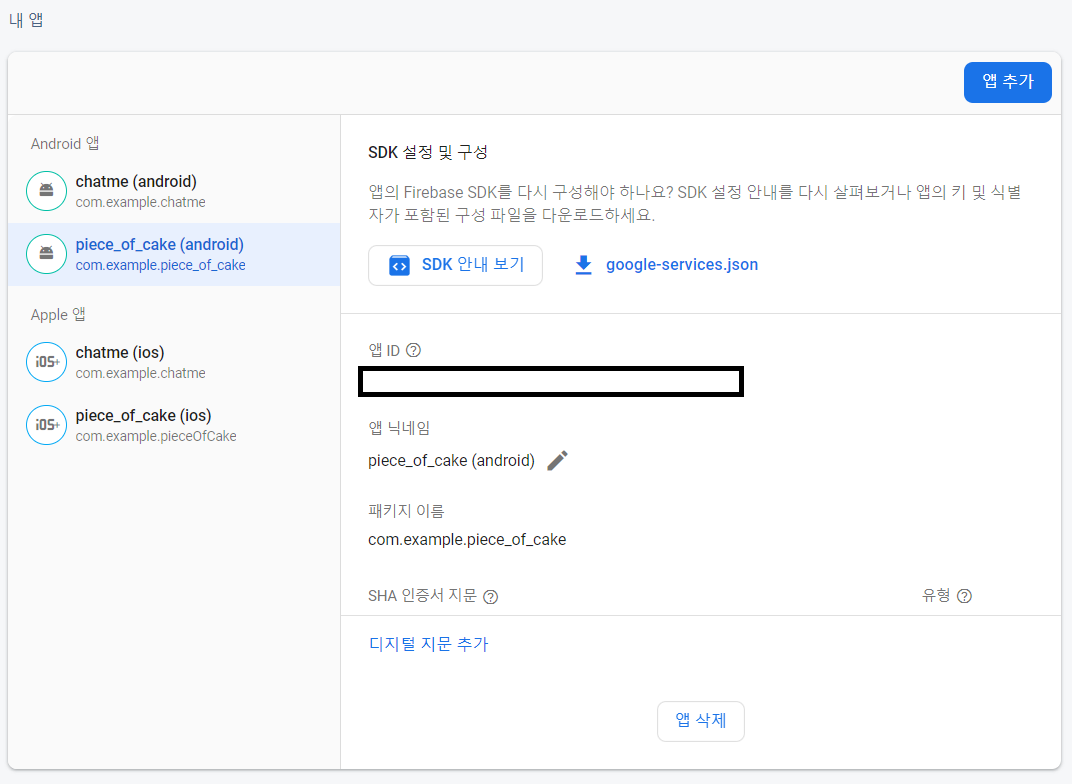


1. 프로젝트 홈에서 가이드를 따라 Flutter 앱을 추가

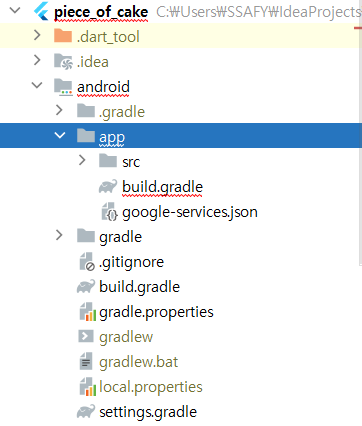




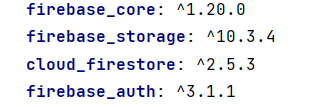
1. 프로젝트 설정에서 추가된 앱을 확인하고 google-services.json을 다운받는다.



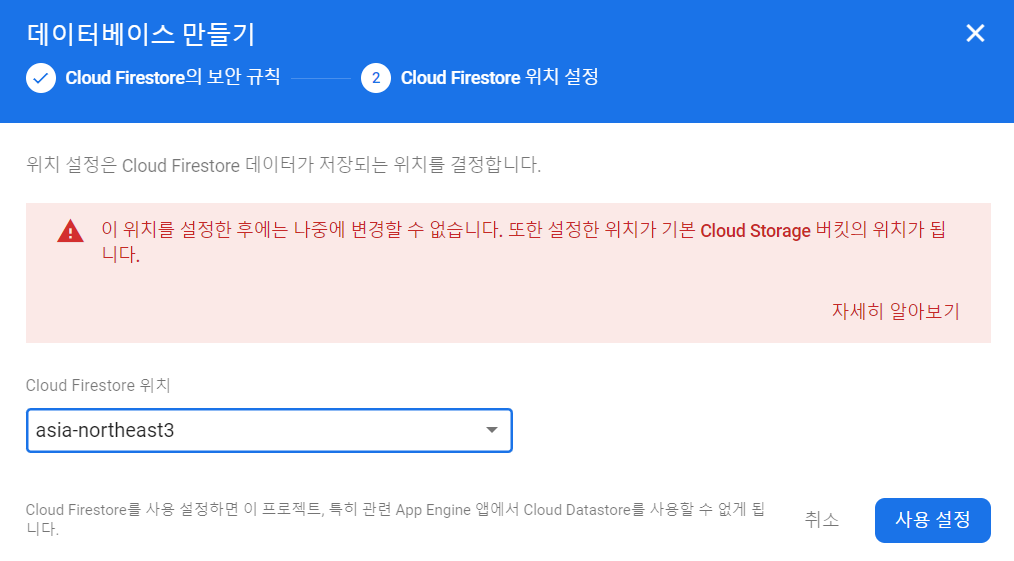
1. Flutter 프로젝트의 android/app 폴더에 google-services.json을 넣는다.



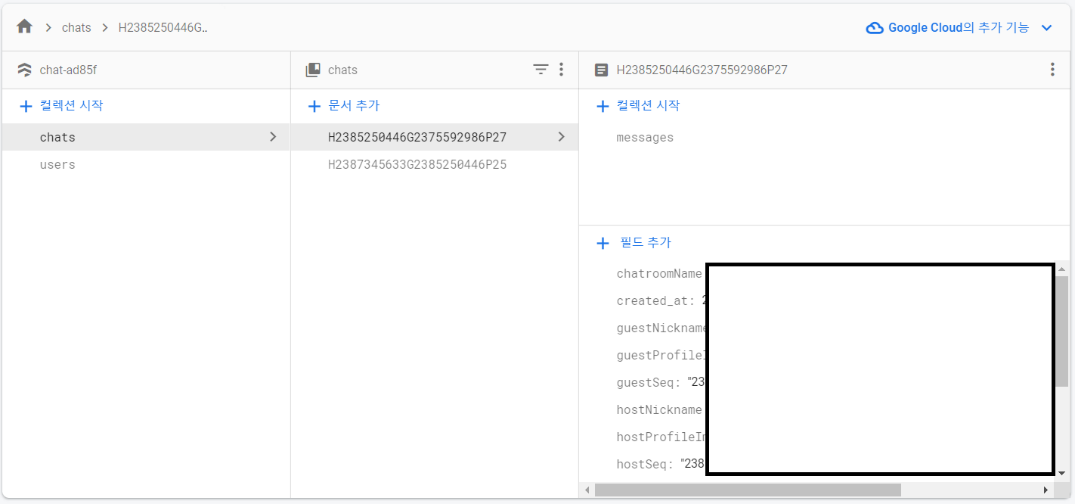
1. Flutter 프로젝트의 pubspec.yaml 파일에 사용할 dependencies를 추가한다.

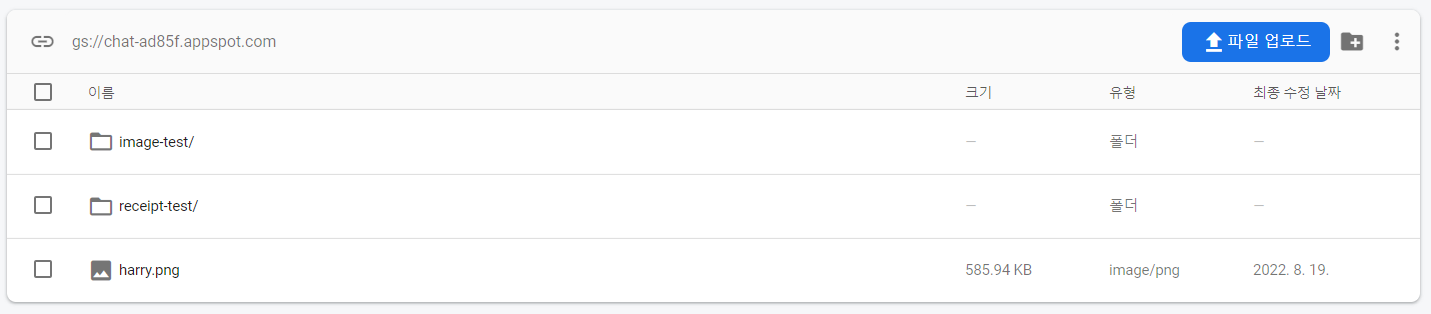


1. 사용할 서비스를 생성한다.



1. Cloud Firebase와 Storage를 사용한다.





**GoogleMap**

텍스트이(가) 표시된 사진

자동 생성된 설명

해당 부분에 API KEY 추가

**Kakao Login**

1. developers.kakao.com에서 애플리케이션 생성



1. 비즈 앱 등록
2. 닉네임, 프로필 사진, 카카오계정(이메일) 필수 동의 설정

테이블이(가) 표시된 사진

자동 생성된 설명

1. 애플리케이션 실행시 출력되는 keyHas를 developers.kakao.com 플랫폼의 키 해시에 추가

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1. AndroidManifest 설정

텍스트이(가) 표시된 사진

자동 생성된 설명

**애플리케이션 아이콘 설정**

1. flutter\_launcher\_icons 설치(터미널)

flutter pub add flutter\_launcher\_icons

1. 앱 아이콘 설정(pubspec.yaml 제일 하단)

Flutter\_icons:

Android: “launcher\_icon”

Ios: true

Image\_path: “assets/app-icon.png”

1. 앱 아이콘 생성(터미널)

flutter pub run flutter\_launcher\_icons:main

**버전 정리**

**개발툴**

Vscode 1.70.0

IntelliJ IDEA 2022.1.3 (Ultimate Edition)

Android Studio Chipmunk 11.0.12+7

MYSQL workbench

MobaXterm\_Personal\_22.1

**Back End**

springBoot 2.7.1

jdk zulu-8

Node.js 16.13.2

springBoot dependencies

spring-boot-starter-web

junit 4.13.1

lombok

jpa

querydsl

commons-io 2.11.0

mysql-connector-java 8.0.28

springfox-swagger-ui 3.0.0

jenkins : latest

mysql : latest

**Front End**

android api 30

flutter sdk 3.0.5

dart sdk ">=2.17.6 <3.0.0"

flutter dependencies

cupertino\_icons: ^1.0.2

http: ^0.13.4

curved\_navigation\_bar: ^1.0.3

kakao\_flutter\_sdk: ^1.2.1

carousel\_slider: ^4.0.0

smooth\_page\_indicator: ^0.3.0-nullsafety.0

like\_button: ^2.0.4

flutter\_speed\_dial: ^6.0.0

provider: ^6.0.3

flutter\_plugin\_android\_lifecycle: ^2.0.7

cached\_network\_image: ^3.2.1

flutter\_native\_splash: ^2.2.7

dropdown\_button2: ^1.7.1

syncfusion\_flutter\_gauges: ^20.2.43

firebase\_core: ^1.20.0

firebase\_storage: ^10.3.4

image\_picker: ^0.8.5+3

dotted\_border: ^2.0.0+2

cloud\_firestore: ^2.5.3

firebase\_auth: ^3.1.1

google\_maps\_flutter: ^2.1.10

geolocator: ^8.0.1

permission\_handler: ^8.3.0

bootpay: ^4.2.5

bootpay\_bio: ^4.2.2

get: ^4.6.5

환경 변수

Flutter 3.0.5 sdk 경로 설정 \flutter\bin(path)에 추가

JDK zulu-8 경로 설정 \Zulu\zulu-8\bin(path)에 추가

JAVA\_HOME 변수 추가 \Zulu\zulu-8