Live 포팅매뉴얼

LIVE

당신이 원하는 집을 앉아서
쉽고 간편하게

🖥️개발 환경

형상관리

- Gitlab

이슈관리

- Jira

테스트 툴

- Postman

UI/UX

- FIgma

SSL

- Certbot

배포

- AWS EC2
  - Docker : 20.10.12
  - Jenkins : 2.375.2

의사소통

- Matermost
- Notion
- Google Meet

DB

- MySQL : 8.0.32
- Redis : 7.0.8
- S3 : 1.12.387

IDE

- Intellij
- Vscode

빌드환경 & OS

- Ubuntu : 20.04.5 LTS
- Window 10

Server

WAS

- Tomcat

WS

- Nginx : 1.18.0

미디어서버

- Kurento : 6.18.0

백엔드

- Java, : Open JDK8 - zulu 8.68.0.19
- SpringBoot : 2.7.7
- Spring Gradle 7.6
  - Spring Data JPA
  - lombok
  - Thymeleaf : 3.1.1.RELEASE

프론트엔드

- React : 18.2.0
- React-Redux : 8.0.5
- Node : 16.16.0
- React-Router-Dom : 6.7.0
- Javascript

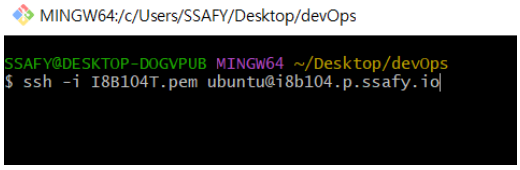
- Jwt : 0.11.2

💾배포

🌈Ubuntu 접속

I8B104T.pem

- **.pem파일**이 있는 곳에서 터미널로 Ubuntu 접속
- 명령어 : `ssh -i I8B104T.pem ubuntu@i8b104.p.ssafy.io`



🌈Docker 설치

- **Repository 설정하기**
  - repository를 이용하기 위해 pakcage 들을 설치

```
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

- **Docker의 Official GPG Key 를 등록 및 오래된 버전 삭제**

```
sudo apt-get remove docker docker-engine docker.io containerd runc
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

- **Docker Engine 설치 및 버전확인 - 최신버전**

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker --version
```

🌈Jenkins 설치 - Use Docker

- **JDK 설치 - verion 8**

```
sudo apt-get install openjdk-8-jdk
```

- **Jenkins 이미지 다운로드**

```
docker pull jenkins/jenkins:lts
```

- **젠킨스 컨테이너 띄우기**

```
sudo docker run -d -p 8080:11111 -v /jenkins:/var/jenkins_home --name jenkins -u root jenkins/jenkins:lts
```

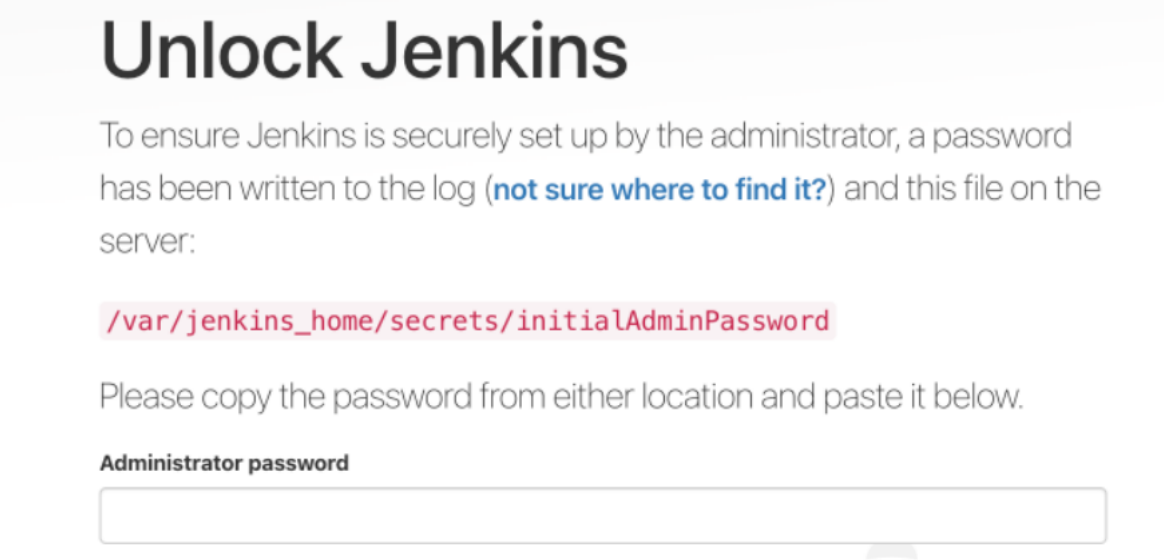
- **-p 8080:11111 :** 컨테이너 외부와 내부 포트를 **포워딩**
- **-v /jenkins:/var/jenkins_home :** 볼륨 마운트, 데이터 접근
- **-u root :** 사용자 계정을 root 로 명시

- 젠킨스 초기 비밀번호 찾기

```
docker logs jenkins
cat ~/jenkins/secrets/initialAdminPassword -> 로그안에 초기 관리자 비밀번호 복사
```

- **젠킨스 실행**

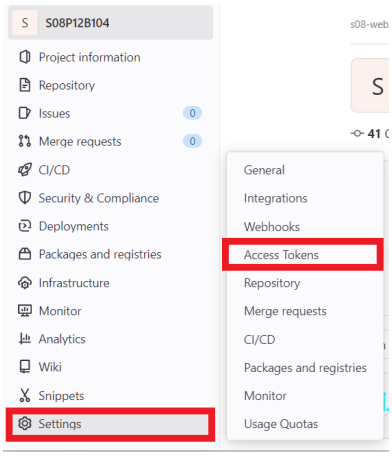
http://i8b104.p.ssafy.io:11111/



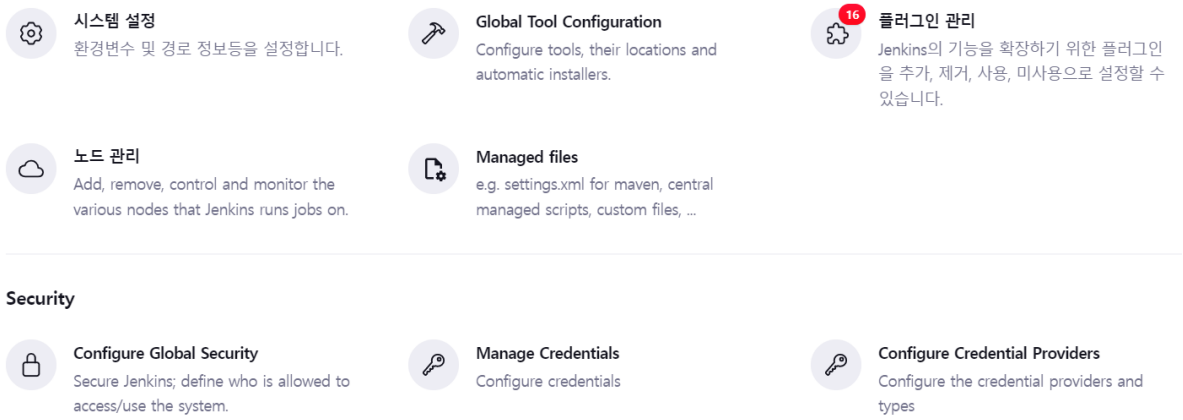
  - 사전의 초기 비밀번호 입력

🌈Jenkins 설정

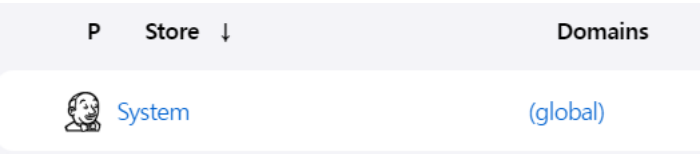
- **GitLab Access Token 등록**
  - Gitlab Repository의 Settings → Access Tokens
  - 양식 작성후 Key 발급 → 저장하기 → Jenkins Credential 설정



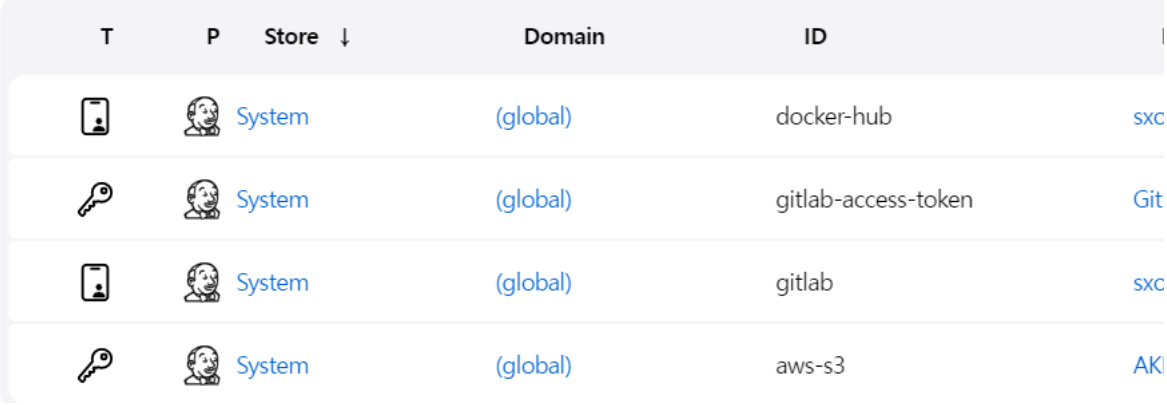
- **Jenkins 관리 → Manage Credentials 이동**



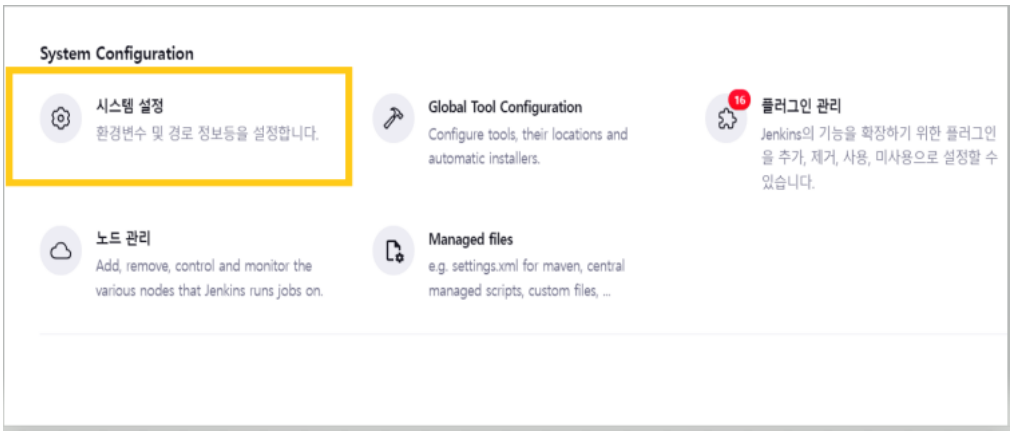
- **하단의 global 클릭후 Credential 생성**



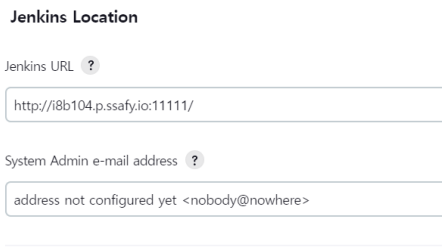
- **작성한 Credential 확인**



🌈Jenkins 시스템 설정

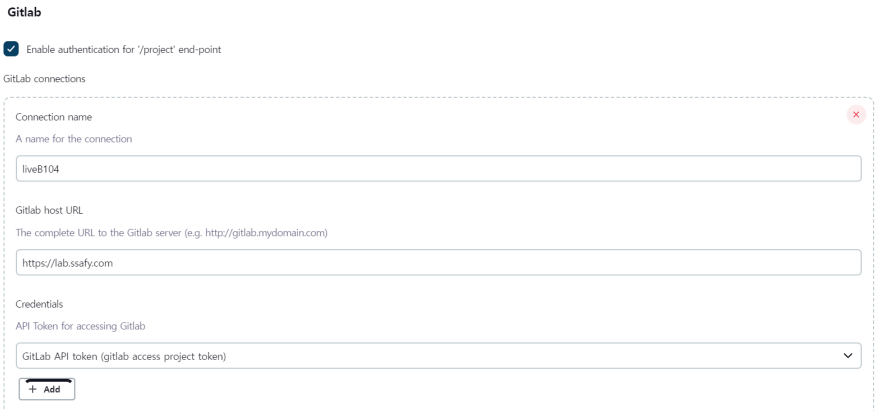


1. **Jenkins Location**



2. **Jenkins 전역 변수 설정**

3. **Gitlab 설정 - Credential 이용**



🌈Front & Back Docker File 설정

> 💡 경로 설정을 위해 DockerFile은 **Root경로**에 만들어줍니다.

Front(React) - DockerFile

```
## Dockerfile(client)

# nginx 이미지를 사용 - 포트를 여는 용도로만 사용
FROM nginx

# work dir
WORKDIR /app

# work dir 에 build 폴더 생성 : /home/test/client/build
RUN mkdir ./build

# host pc의 현재경로의 build 폴더를 workdir 의 build 폴더로 복사
```

```
ADD ./build ./build

# nginx 의 default.conf 를 삭제
RUN rm /etc/nginx/conf.d/default.conf

# host pc 의 nginx.conf 를 복사
COPY ./default.conf /etc/nginx/conf.d

# 80 포트 오픈
EXPOSE 3000

# container 실행 시 자동으로 실행할 command. nginx 시작
CMD ["nginx", "-g", "daemon off;"]
```

Back(Spring) - DockerFile

```
# base-imagee
FROM openjdk:8-jdk-alpine
# 변수 설정 (빌드 파일의 경로)
ARG JAR_FILE=server/build/libs/live-0.0.1-SNAPSHOT.jar
# 빌드파일을 컨테이너로 복사
COPY ${JAR_FILE} app.jar
# jar 파일 실행
ENTRYPOINT ["java", "-Dkms.url=ws://live-live.store:8888/kurento", "-jar", "app.jar"]
```

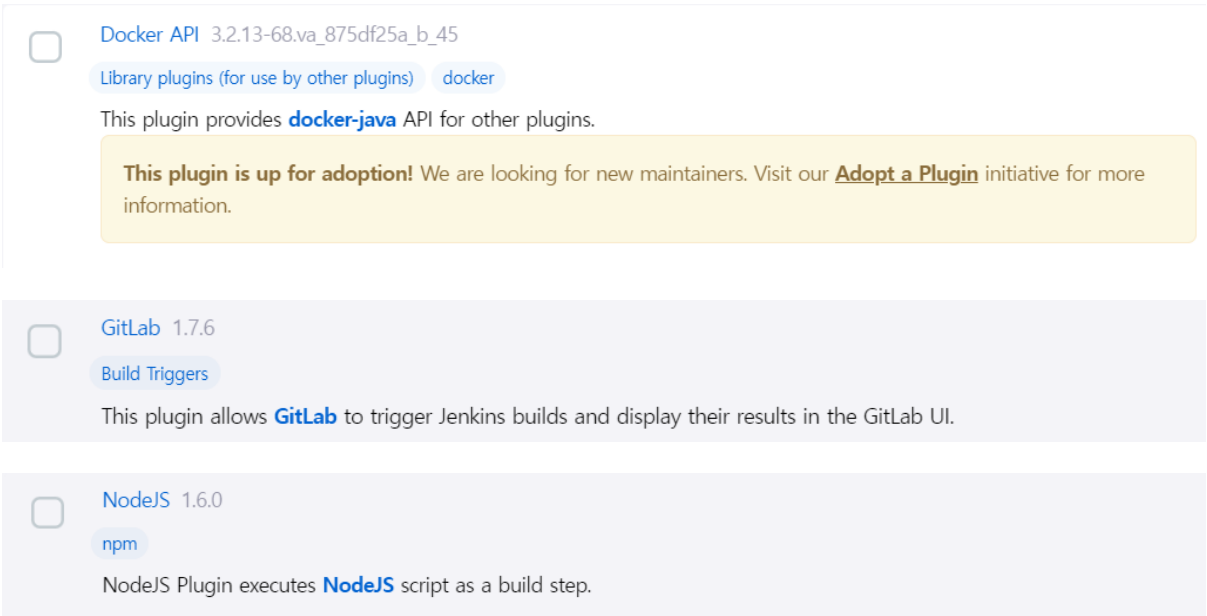
- Kurento 서버의 JVM실행 경로를 위해 `"-Dkms.url=ws://live-live.store:8888/kurento"` 설정

Back(Spring) - Nginx default.conf

```
server {
  listen 3000;
  location / {
    root   /app/build;
    index  index.html index.htm index.php;
    try_files $uri $uri/ /index.html;
  }

  error_page   500 502 503 504  /50x.html;
  location = /50x.html {
    root   /app/build;
  }
}
```

- 3000포트 지정

🌈Jenkins Plug-in 설치

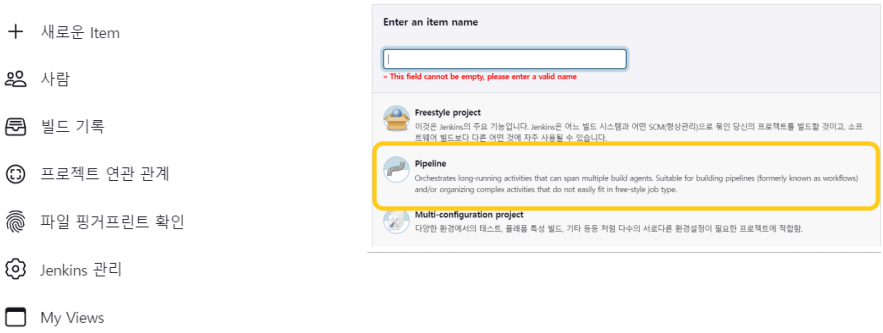


- React 빌드를 위해 NodeJS 설치

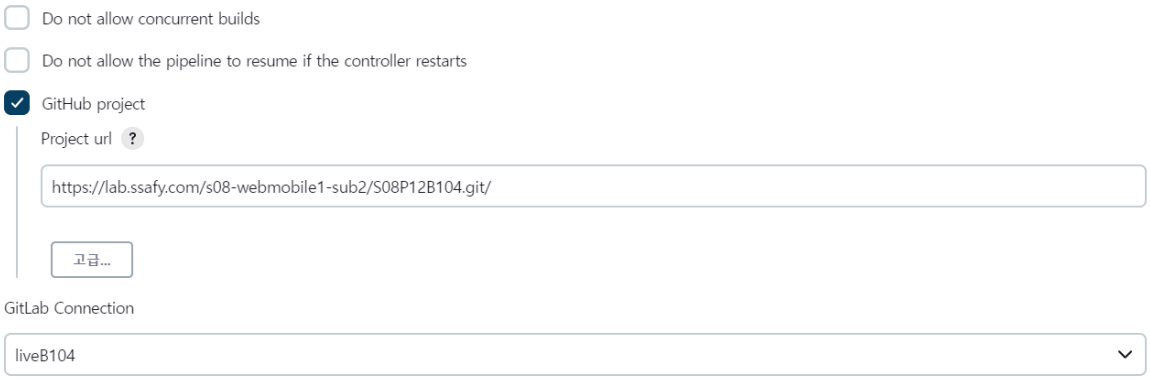
🌈Front & Back Pipeline 만들기

Front

- Jenkins 새로운 Item → 파이프라인 만들기

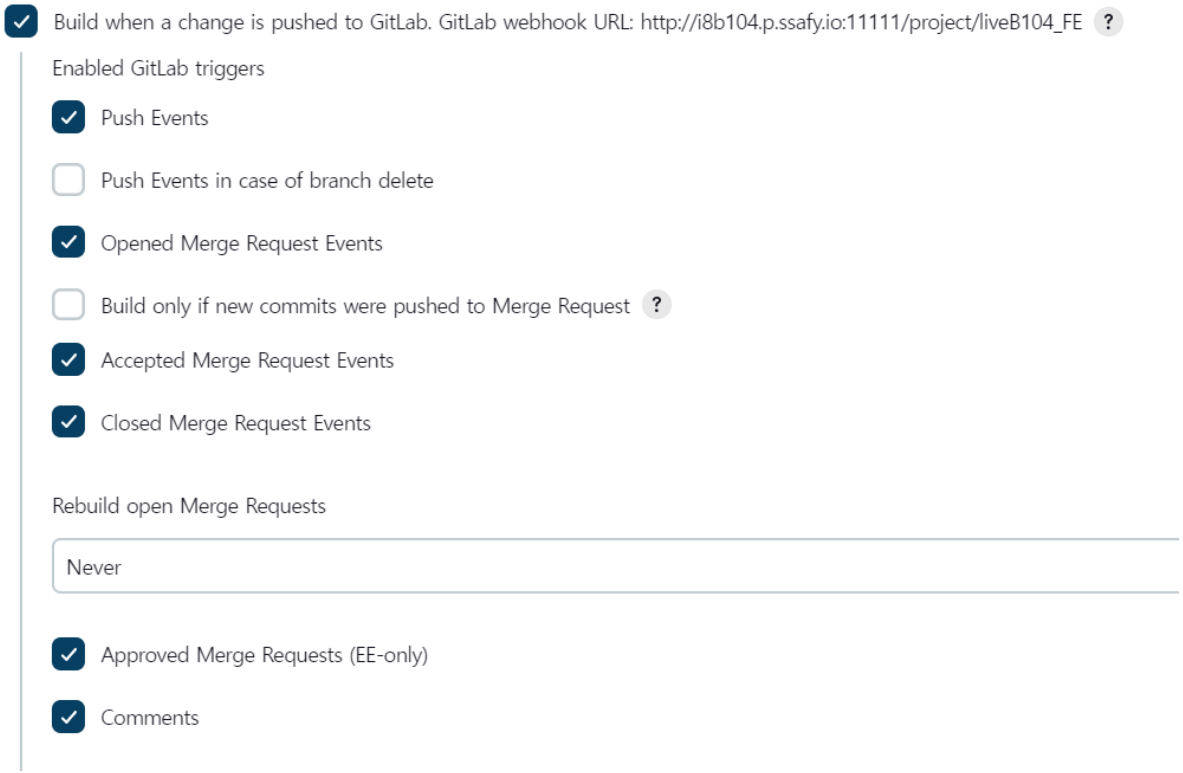


1. **Gitlab url 등록 및 Credential 사용**



- Gitlab Connection에 이전에 만들어놓은 Credential 클릭 → Credential ID

2. **Check Box - Webhook Trigger**



3. **Pipeline 작성 -Script**

```
pipeline {
    agent any
    tools {
        nodejs "live_nodejs"
    }

    stages {
        stage('Prepare') {
            steps {
                git branch: 'FE/develop',
                credentialsId: 'gitlab',
                url: 'https://lab.ssafy.com/s08-webmobile1-sub2/S08P12B104'
            }
        }

        stage('React Build') {
            steps {
                dir("server") {
                    sh "npm install"
                    sh "CI=false npm run build"
                }
            }
        }

        stage('Docker Build') {
            steps {
                sh 'docker build -t live/front ./server'
            }
        }

        stage('Deploy') {
            steps{
                sh 'docker ps -f name=front -q | xargs --no-run-if-empty docker container stop'
                sh 'docker container ls -a -fname=front -q | xargs -r docker container rm'
                sh 'docker images --no-trunc --all --quiet --filter="dangling=true" | xargs --no-run-if-empty docker rmi'
                sh 'docker run -d --name front -p 3000:3000 live/front'
            }

        }

        stage('Finish') {
            steps{
                sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
            }
        }
```

```
    }
}
```

- Pipeline 단계 세분화
  - Prepare - 준비 단계
  - React Build - Build에 필요한 npm 설치
  - Docker Build - docker file Build (작성한 docker file 설정 순서로)
  - Depoly - 빌드된 Docker 이미지 실행 → 이미 만들어진 같은 이미지 파일이 있다면 삭제

Back

- 기존 설정은 위와 동일

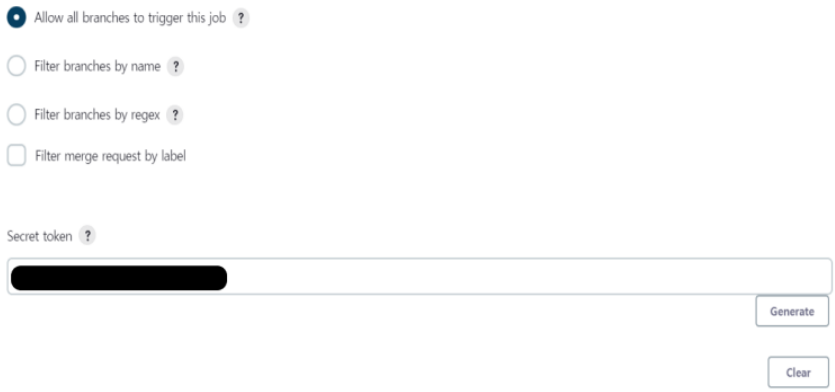
```
pipeline {
    agent any

    stages {
        stage('Prepare') {
            steps {
                git branch: 'BE/develop',
                credentialsId: 'gitlab',
                url: 'https://lab.ssafy.com/s08-webmobile1-sub2/S08P12B104'
            }
        }

        stage('Spring Build') {
            steps {
                dir("server") {
                    sh 'chmod +x gradlew'
                    sh  './gradlew clean build'
                    sh 'ls -al ./build'
                }
            }
        }

        stage('Docker Build') {
            steps {
                sh 'docker build --build-arg JAR_FILE=build/libs/live-0.0.1-SNAPSHOT.jar -t live/back ./server'
            }
        }

        stage('Deploy') {
            steps{
                sh 'docker ps -f name=back -q | xargs --no-run-if-empty docker container stop'
                sh 'docker container ls -a -fname=back -q | xargs -r docker container rm'
                sh 'docker images --no-trunc --all --quiet --filter="dangling=true" | xargs --no-run-if-empty docker rmi'
                sh 'docker run --name back -v /live/records:/live/records -p 8080:8080 \
                    -e "ACCESS_KEY_AWS_S3=${ACCESS_KEY_AWS_S3}" \
                    -e "SECRET_KEY_AWS_S3=${SECRET_KEY_AWS_S3}" \
                    -e "EMAIL_ADDRESS=${EMAIL_ADDRESS}" \
                    -e "BUCKET_ADDRESS=${BUCKET_ADDRESS}" \
                    -e "COMMON_PASSWORD=${COMMON_PASSWORD}" \
                    -e "KEY_STORE_PASSWORD=${KEY_STORE_PASSWORD}" \
                    -e "NAVER_ACCESSKEY=${NAVER_ACCESSKEY}" \
                    -e "NAVER_SECRETKEY=${NAVER_SECRETKEY}" \
                    -e "NAVER_SERVICEID=${NAVER_SERVICEID}" \
                    -e "NAVER_CALLING_NUMBER=${NAVER_CALLING_NUMBER}" \
                    -e "JWT_SECRETKEY=${JWT_SECRETKEY}" \
                    live/back'
            }
        }

        stage('Finish') {
            steps{
                sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
            }
        }
    }
}
```

- Pipeline 단계 세분화
  - Prepare - 준비 단계
  - Spring Build - 빌드를 위한 gradlew 권한 변경
  - Docker Build - docker file Build (작성한 docker file 설정 순서로)

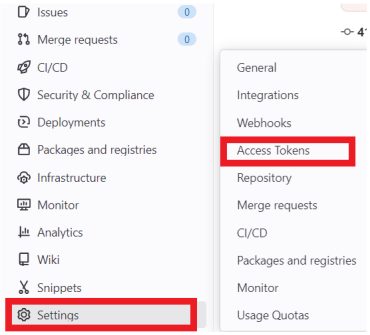
        - yml파일의 암호, Key 중요 정보를 Jenkins 환경변수에 설정된 변수로 넘겨주기
    - Depoly - 빌드된 Docker 이미지 실행 → 이미 만들어진 같은 이미지 파일이 있다면 삭제

🌈Gitlab Branch Webhook 설정

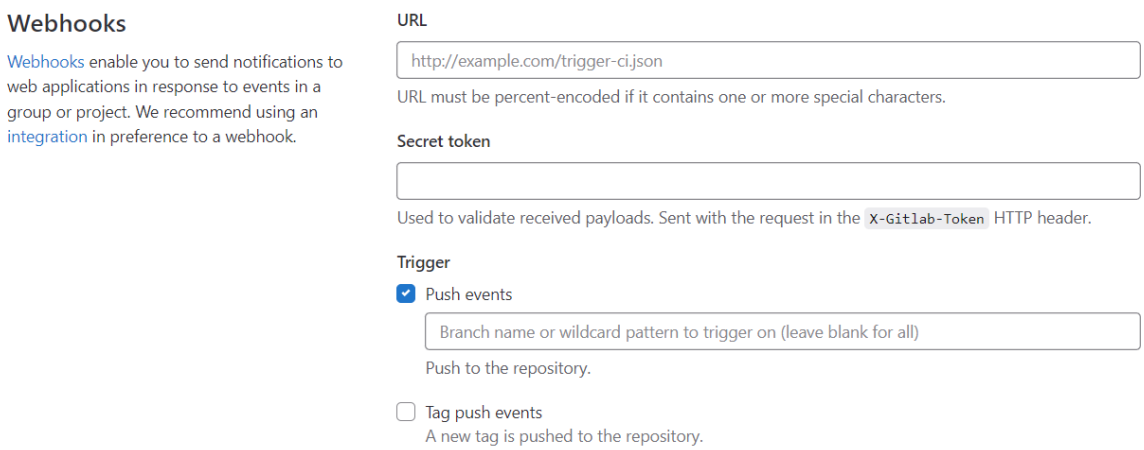
- **Pipeline 설정 Build Triggers → Secret Token 발급**
    - Generate로 **생성 후 저장**

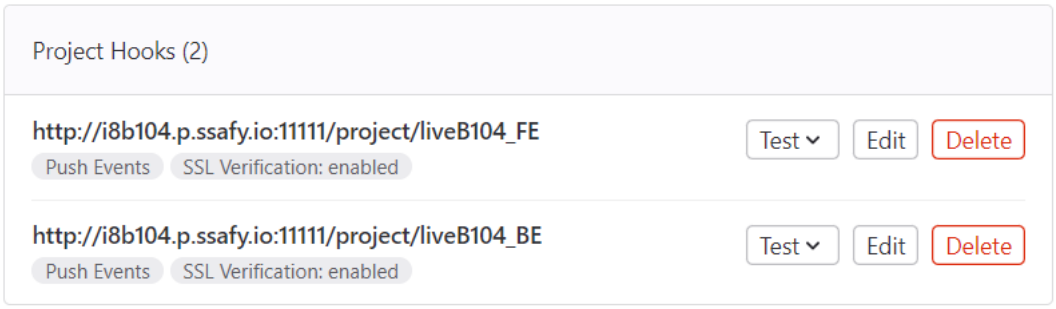


- **GitLab 이동 → Settings → Webhook**



- Branch Trigger 설정 및 확인
    - Jenkins에서 발급받은 Secret 키 입력 → Secret Token
    - CI/CD 배포 Branch 설정 후 확인





🌈Jenkins 실행 및 확인

- **Jenkins Pipeline진행 확인 - 서버상태 체크**

- **Docker 컨테이너 확인**

```
docker ps -a
```

```
ubuntu@ip-172-26-15-64:/redis$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS          PORTS
AMES
21dd4e66c3cd   live/front           "/docker-entrypoint.…"   7 minutes ago   Up 7 minutes    80/tcp, 0.0.0.0:3000->3000/tcp, :::3000->3000/t
ront
2781ca91bbd1   live/back            "java -Dkms.url=ws:/…"   2 hours ago     Up 2 hours      0.0.0.0:8080->8080/tcp, :::8080->8080/tcp
ack
```

🌈Kurento 설치 - Docker File

- **Kurento 이미지 파일 내려받기**

```
docker pull kurento/kurento-media-server:latest
```

- Kurento Media Server 실행

```
docker run -d --name kms --network host \kurento/kurento-media-server:latest
```

- –network: 컨테이너의 네트워크 설정
  - host : 컨테이너를 호스트 컴퓨터와 동일한 네트워크에서 돌리겠다는 의미이다.

```
docker run --rm \-p 8888:8888/tcp \-p 5000-5050:5000-5050/udp \-e KMS_MIN_PORT=5000 \-e KMS_MAX_PORT=5050 \kurento/kurento-media-serve
```

• 마지막으로 실행 중인 미디어 서버의 포트 설정

🌈Nginx 설치 - 로컬 환경

- **Nginx 설치**

```
sudo apt install nginx
```

- **실행**

```
sudo service start nginx
sudo service status nginx
# 현재 nginx 동작 상태 확인
systemctl status nginx.service
```

```
root@ubuntu:/# service nginx status
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2021-06-09 23:36:02 PDT; 3min 32s ago
       Docs: man:nginx(8)
    Process: 97090 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=ex>
    Process: 97091 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, sta>
   Main PID: 97092 (nginx)
      Tasks: 3 (limit: 4618)
     Memory: 3.9M
     CGroup: /system.slice/nginx.service
             ├─97092 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             ├─97093 nginx: worker process
             └─97094 nginx: worker process
```

- **Nginx Version 확인**

```
sudo dpkg -l nginx
```

- **기본 경로 nginx.conf 설정 /etc/nginx/**

```
cd /etc/nginx
```

```
ubuntu@ip-172-26-15-64:/etc$ cd nginx/
ubuntu@ip-172-26-15-64:/etc/nginx$ ls
 conf.d          fastcgi_params   koi-win      modules-available   nginx.conf     scgi_params       sites-enabled  'udo nginx -t'   win-utf
 fastcgi.conf    koi-utf          mime.types   modules-enabled     proxy_params   sites-available   snippets        uwsgi_params
```

/etc/nginx/conf.d 이동 후 .conf파일 생성(설정 파일)

```
cd /etc/nginx/conf.d
sudo nano front.conf
```

```
# 443 포트로 접근시 ssl을 적용한 뒤 3000포트로 요청을 전달해주도록 하는 설정.
server {
        server_name live-live.store;
```

```
        location / {
                proxy_pass http://live-live.store:3000;
        }

        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/live-live.store/fullchain.pem; # managed by Cert>
        ssl_certificate_key /etc/letsencrypt/live/live-live.store/privkey.pem; # managed by Ce>

        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

# 80 포트로 접근시 443 포트로 리다이렉트 시켜주는 설정
server {
        if ($host = idu-market.shop) {
                return 301 https://$host$request_uri;
        } # managed by Certbot


        listen 80;
        server_name live-live.store;
        return 404; # managed by Certbot
}
```

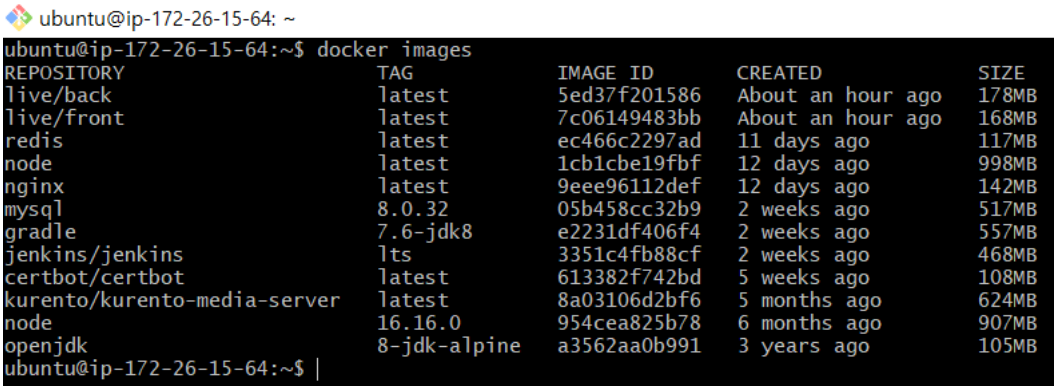
🌈DB 설치 - Docker File

Mysql

- **Mysql 8.0.32버전 이미지 파일 내려받기**

```
docker pull mysql:8.0.32
```

- **이미지 설치 확인**

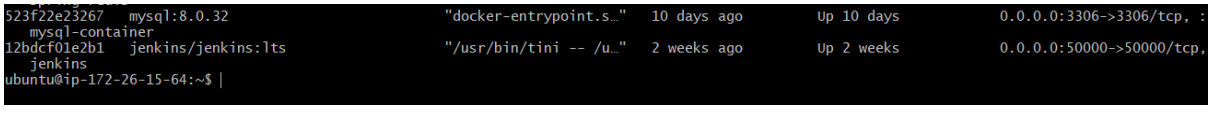


- **Docker 이미지 실행**

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=liveB104 -d -p 3306:3306 mysql:8.0.32
```

- 포트 3306 바인딩, mysql root 비밀번호 설정
- **Docker 컨테이너 리스트 출력**

```
docker ps -a
```



- **MySQL Docker 컨테이너 시작&중지&재시작**

```
# MySQL Docker 컨테이너 중지
$ docker stop mysql-container

# MySQL Docker 컨테이너 시작
$ docker start mysql-container

# MySQL Docker 컨테이너 재시작
$ docker restart mysql-container
```

- **MySQL Docker 컨테이너 접속 및 Database 확인**

```
$ docker exec -it mysql-container bash
root@dc557b92f573:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)
```

- 사용할 Database 생성

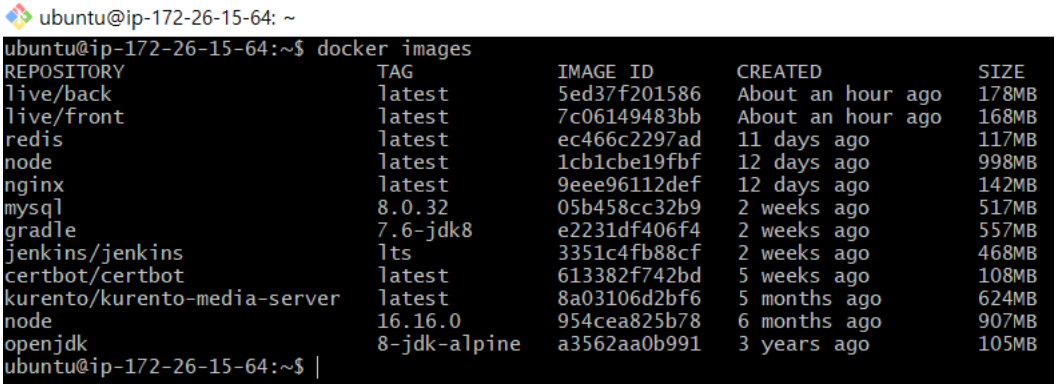
```
mysql> create database live;
| Database           |
+--------------------+
| information_schema |
| live               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

Redis

- **Redis Docker 이미지 설치하기 - 최신버전**

```
docker pull redis
```

- **이미지 설치 확인**



- **Docker 이미지 실행**

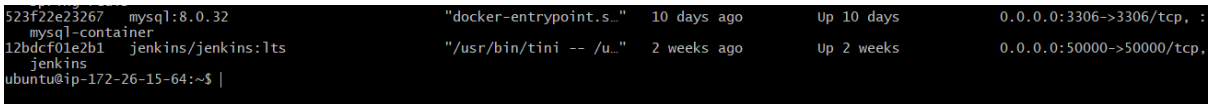
```
docker run --name redis -d -p 6379:6379 -v ./redis/redis.conf:/redis/redis.conf spring-redis
```

- 포트 3306 바인딩, mysql root 비밀번호 설정
- -v ./redis/redis.conf:/redis/redis.conf : 볼륨 바인드를 통해 conf 설정파일을 지정
- **Redis.conf 설정**

```
cd ../..
mkdir redis
cd redis
sudo vi(or nano) redis.conf
// nano 설정 코드
  port 6379  // 포트 설정
bind 0.0.0.0 // 전부 열림 설정
requirepass liveB104 // redis 비밀번호

//나가기 vi : esc -> :wq  nano : ctrl + O
```

- **Docker 컨테이너 리스트 출력**

```
docker ps -a
```



- **Redis Docker 컨테이너 시작&중지&재시작**

```
# Redis Docker 컨테이너 중지
$ docker stop spring-redis

# Redis Docker 컨테이너 시작
$ docker start spring-redis

# Redis Docker 컨테이너 재시작
$ docker restart spring-redis
```

- **Redis Docker 컨테이너 접속 및 redis-cli 비밀번호 설정 확인**

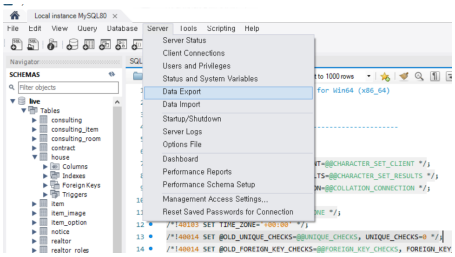
```
docker exec -it spring-redis bash
redis-cli
auth liveB104
ping
pong OK
```

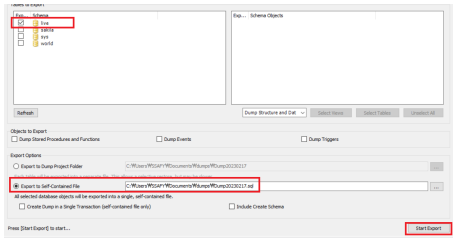
📃Dummy data

- 최신 Dump File

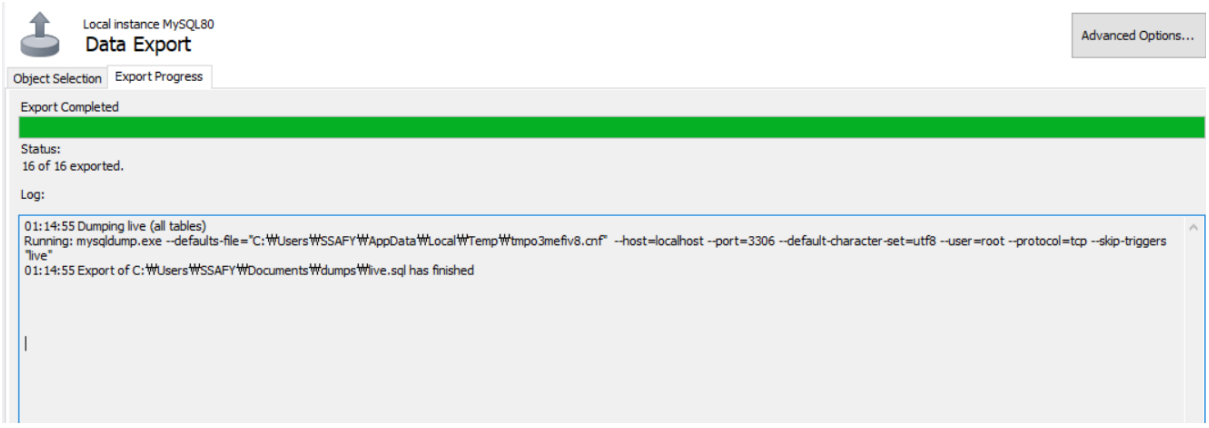
```
-- MySQL dump 10.13  Distrib 8.0.32, for Win64 (x86_64)
--
-- Host: localhost    Database: live
-- ------------------------------------------------------
-- Server version 8.0.32

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `consulting`
--

DROP TABLE IF EXISTS `consulting`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `consulting` (
  `consulting_no` bigint NOT NULL AUTO_INCREMENT,
  `created_date` datetime DEFAULT NULL,
  `last_modified_date` datetime DEFAULT NULL,
  `consulting_date` datetime DEFAULT NULL,
  `link` varchar(255) DEFAULT NULL,
  `requirement` varchar(255) DEFAULT NULL,
  `status` int DEFAULT NULL,
  `realtor_no` bigint DEFAULT NULL,
  `user_no` bigint DEFAULT NULL,
  PRIMARY KEY (`consulting_no`),
  KEY `FKnmtg66b7ovo3m7ygr8c6vtm0q` (`realtor_no`),
  KEY `FKfc8l2pixd2jh0myc2lc8yeufs` (`user_no`),
  CONSTRAINT `FKfc8l2pixd2jh0myc2lc8yeufs` FOREIGN KEY (`user_no`) REFERENCES `users` (`user_no`),
  CONSTRAINT `FKnmtg66b7ovo3m7ygr8c6vtm0q` FOREIGN KEY (`realtor_no`) REFERENCES `realtor` (`realtor_no`)
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
...
...
```

- **Data 추출**





- **추출 및 확인**




```
1 • SELECT * FROM live.notice;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

notice_no	created_date	last_modified_date	link	notice_info	notice_writer	realtor_no	user_no
1	2023-02-17 00:00:00	2023-02-17 00:00:00	NULL	상담이 시작됩니다. 상담실로 입장해 주세요.	두기철	1	1
2	2023-02-07 00:00:00	2023-02-07 00:00:00	NULL	상담 매물 등록이 완료되었습니다.	김순자	2	1
3	2023-02-02 00:00:00	2023-02-02 00:00:00	NULL	상담이 시작됩니다. 상담실로 입장해 주세요.	박창호	3	1
4	2023-02-20 00:00:00	2023-02-20 00:00:00	NULL	상담 매물 등록이 완료되었습니다.	김대희	4	1
5	2023-02-23 00:00:00	2023-02-23 00:00:00	NULL	상담이 시작됩니다. 상담실로 입장해 주세요.	오승준	5	1
6	2023-02-17 00:00:00	2023-02-17 00:00:00	NULL	상담 매물 등록이 완료되었습니다.	황준희	6	1
7	2023-01-24 00:00:00	2023-01-24 00:00:00	NULL	상담 매물 등록이 완료되었습니다.	이은주	7	1
8	2023-02-19 00:00:00	2023-02-19 00:00:00	NULL	상담이 취소되었습니다.	조민성	8	1
9	2023-02-21 00:00:00	2023-02-21 00:00:00	NULL	상담이 시작됩니다. 상담실로 입장해 주세요.	박창호	9	1
10	2023-02-17 00:00:00	2023-02-17 00:00:00	NULL	상담 신청입니다. 목록을 확인해 주세요.	김싸피	1	1
11	2023-01-29 00:00:00	2023-01-19 00:00:00	NULL	사용자가 상담을 취소하였습니다.	김태수	1	2
12	2023-02-07 00:00:00	2023-02-07 00:00:00	NULL	상담 신청입니다. 목록을 확인해 주세요.	이지우	1	3
13	2023-02-02 00:00:00	2023-02-02 00:00:00	NULL	사용자가 상담을 거절하였습니다.	신현광	1	4
14	2023-02-17 00:00:00	2023-02-17 00:00:00	NULL	상담 신청입니다. 목록을 확인해 주세요.	김영주	1	6

🔑SSL 설정

- **Certbot - 패키지 설치**

```
sudo add-apt-repository ppa:certbot/certbot
// enter
sudo apt-get install python-certbot-nginx
```

- 특정 도메인 SSL 발급하기

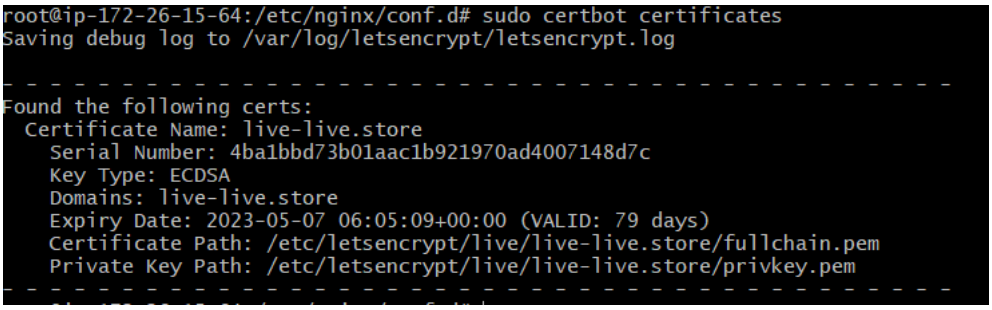
```
sudo certbot --nginx -d live-live.store --email sxott94@gmail.com --agree-tos
```

- 설치 설정 지정하기

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about our work
encrypting the web, EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: n
<== 이메일로 news 같은 소식을 받을 건지를 물어보는 것입니다 N 를 해도 됩니다.

Which names would you like to activate HTTPS for?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: h2code.cf
2: mydomain.h2code.cf
3: opcache.h2code.cf
4: www.h2code.cf
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel): // enter
<== nginx에 등록된 4개의 호스트가 모두 리스트 업되며 각각 받을 것인지 아님 모두 한 꺼번에 받을 것인지를 선택 하게 됩니다.

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
<== 이부분은 HTTPS 로 Redirect 로의 설정을 대신 certbot이 해줄것인지 아닌 지를 선택하는 부분입니다.
2번을 선택하면 certbot이 기존의 conf 파일에서 redirect 구문을 추가하고 포트 변경 ,ssl_ciphers,ssl_protocols 정보를 모두 다 설정해주게 됩니다 포스팅에서는 2번
```

- **인증서 발급확인**

```
sudo certbot certificates
```



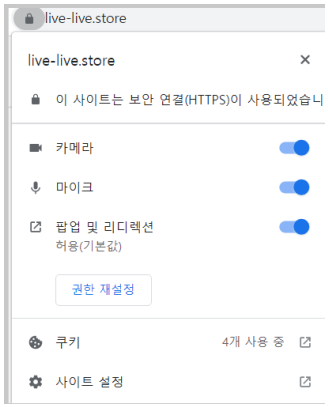
- 기본경로 front.conf 파일 확인 후 Nginx 재시작

```
# conf(설정파일) 수정 후에도 반드시 재가동 해주어야한다.
sudo systemctl restart nginx
```

- **인증서 기한 (90일 발급일로부터) → 자동화**

```
sudo certbot renew --dry-run
```

- HTTPS 확인

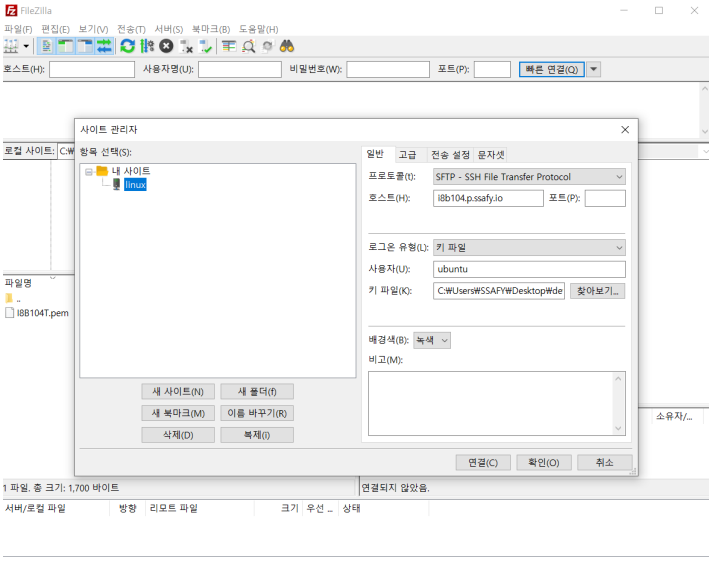


🌈Back 서버 WS 설정

- **SSL인증서를 spring boot에서 필요한 형식(PKCS12)로 변환**
  - pem 파일이 위치한 경로에서 진행 (Certbot로 발급받은 .pem파일)

```
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem
-out keystore.p12 -name tomcat -CAfile chain.pem -caname root
```

- password 설정 → 원하는 비밀번호 설정 후 저장
- **keystore.p12 파일을 /src/main/resources에 이동**
  - 로컬환경에서 가져오는 방법
  - FileZilla 이용
  - 파일 → 사이트 관리자 클릭

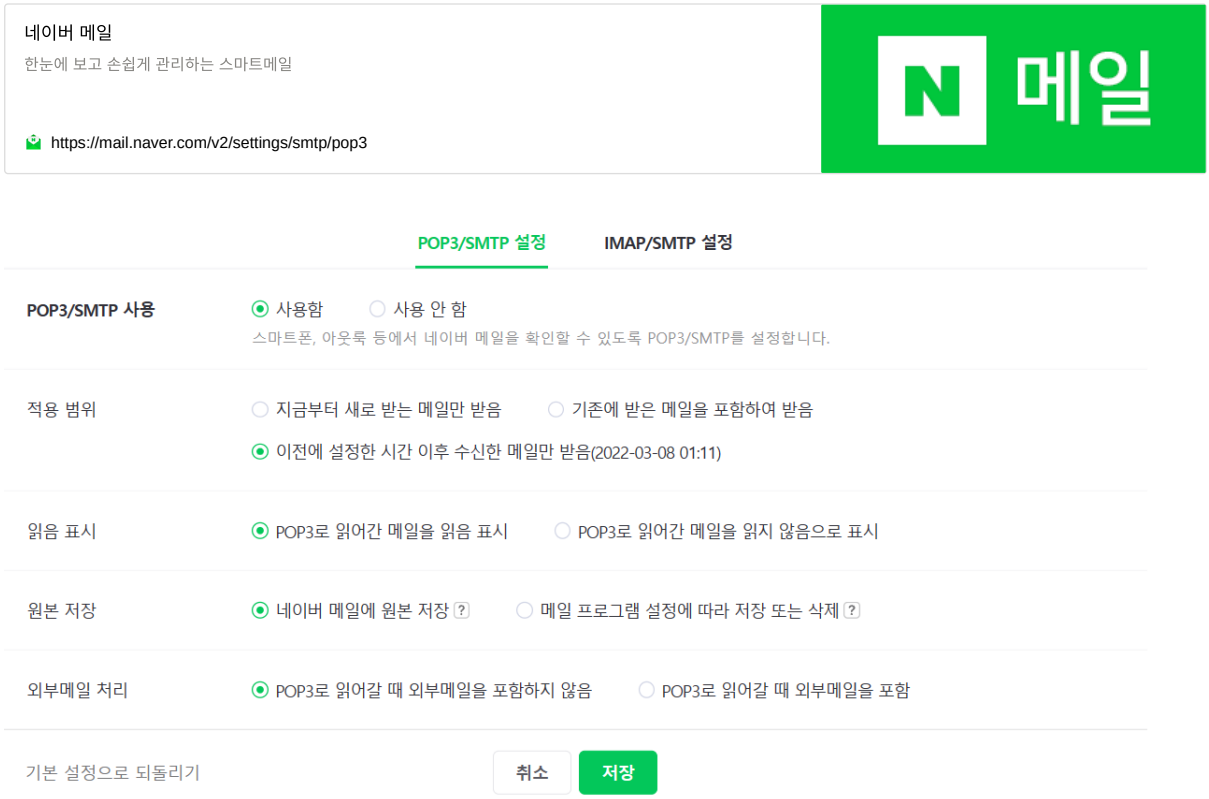


- 연결 후 로컬환경의 **keystore.p12 파일을 /src/main/resources에 이동**
- **.yml에 ssl 설정 등록 → 배포버전에서는 비밀번호 변수로 지정**

```
ssl:
  key-store: classpath:ssl/keystore.p12
  key-store-type: PKCS12
  key-store-password: liveB104key
```

외부 서비스 문서

- **SMTP**



- **공공 데이터 포털(국토교통부 부동산 중개업 정보 서비스, 사업자 등록번호 진위확인 서비스)**

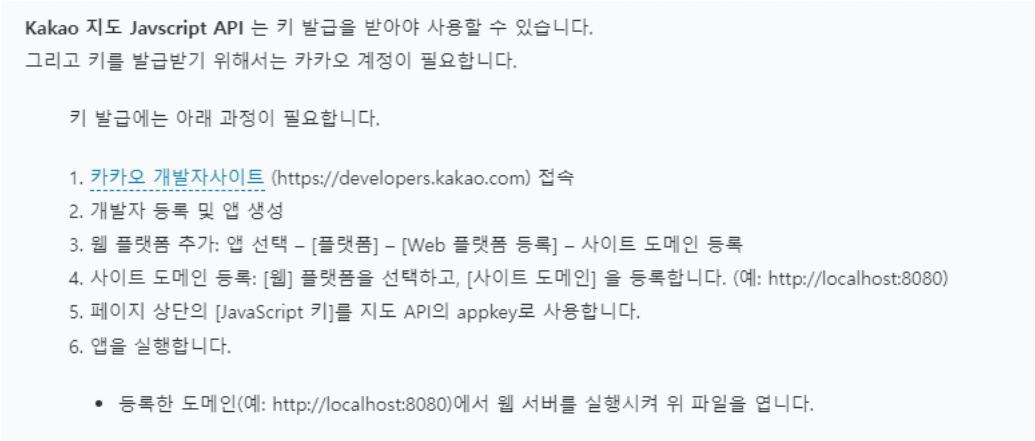
공공 데이터를 이용한 중개사무소 검색 및 중개인 검증 기능

사업자 번호 확인 등 신분 확인 검증 기능 제공

* 활용신청 시 '**위치기반서비스사업신고필증**'이 등록되지 않으면 반려가 될 수 있으니 참고 하시기 바랍니다.

- **Kakao 지도 API**

건물(매물) 위치 확인 및 표시 API 제공

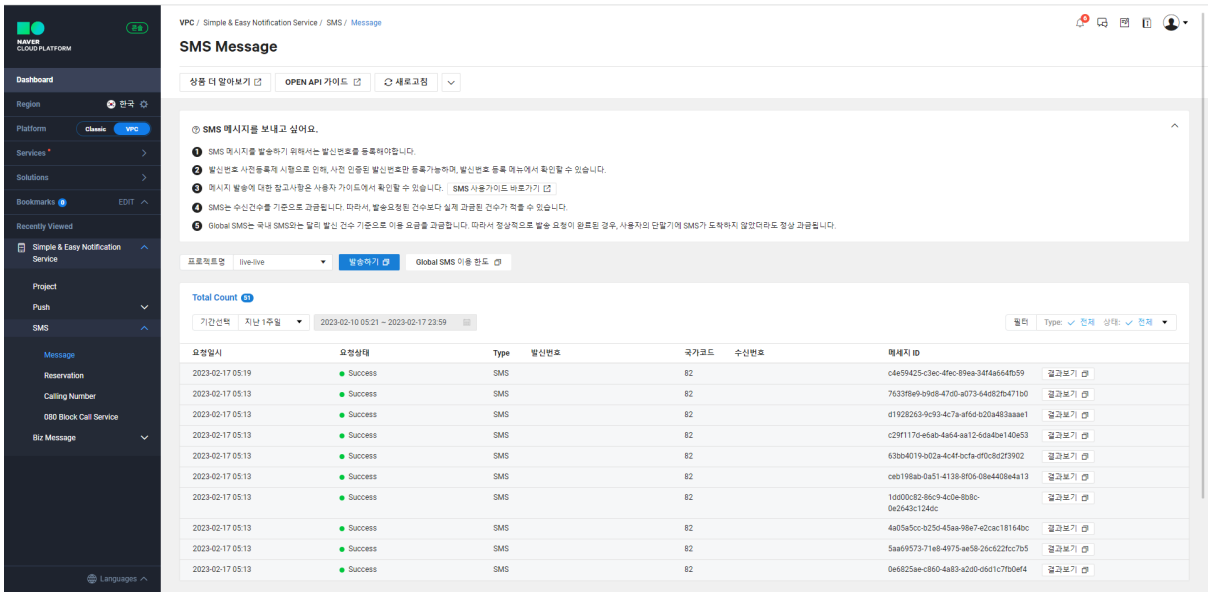


- **Kakao(다음) 우편번호 API**

주소 검색 및 확인 API 제공

- **SMS Message Service**

고객에게 알람용 문자 메시지 전송하는데 사용



- AWS S3

사용자의 프로필 이미지, 매물의 사진을 저장하기 위해 사용한 스토리지 서비스