SELLOG

포팅 매뉴얼

포팅 매뉴얼

개발환경

1. Front-End
    a. Visual Studio Code 1.75.1
    b. React 18.2.0
        - Recoil
        - styled-components 5.3.6
        - axios 1.2.3
    c. Modeling
        i. Blender
        ii. three.js

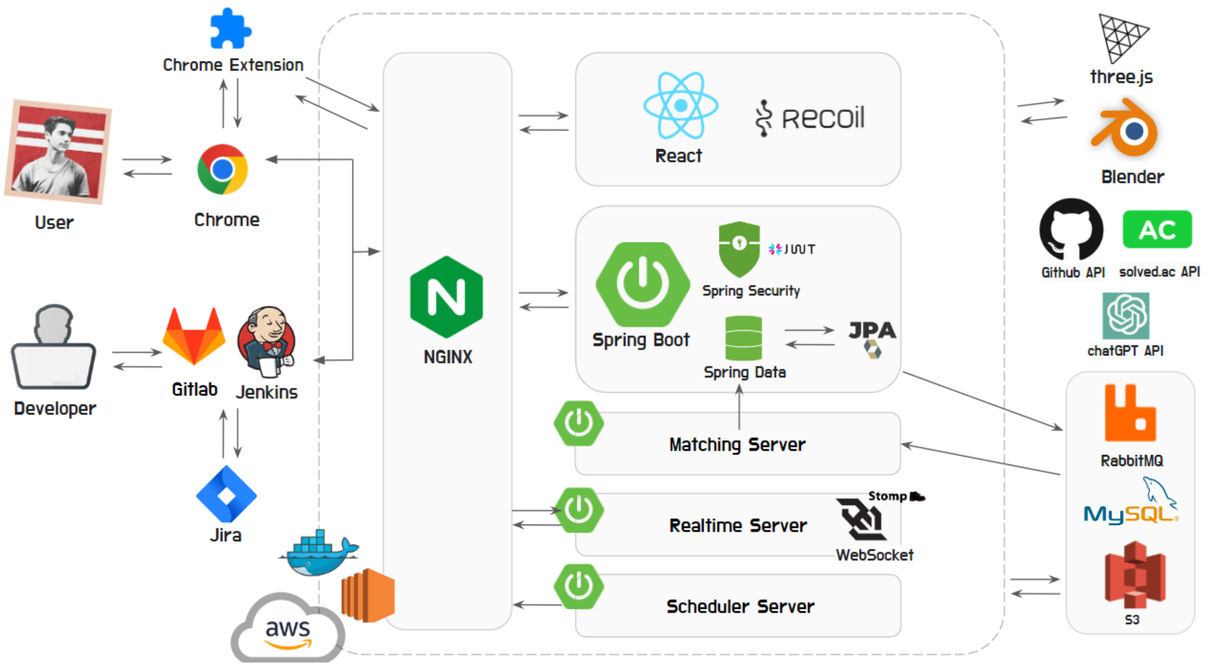
2. Back-End
    a. IntelliJ IDEA 2022.3.1
    b. SpringBoot Gradle 2.7.10
        - JAVA 11.0.16.14
        - Spring Security
        - Spring Data JPA
        - JWT 0.11.5
        - QueryDSL
        - HikariCP 4.0.3
    c. Test
        a. JUnit5
        b. Mockito
        c. Jacoco toolVersion0.8.7

3. DataBase
    a. MySQL 8.0.31
    b. RabbitMQ

4. CI/CD
    a. AWS EC2
        - Ubuntu 20.04
        - Docker 20.10.23
        - Nginx 1.18.0
        - Jenkins 2.378.1
5. Chrome Extension
    - Html, Css, Javascript

5. 공통

- Gitlab
- Jira
- Mattermost, Notion
- Postman 10.9.4



EC2 세팅

1. Docker

1-1. Docker 설치

apt-get 업데이트, 관련 패키지 설치

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

Docker 공식 GPG-Key 추가

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Repostory 설정

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

JDK 설치

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

2. Jenkins

2-1. Jenkins container 설치

```
docker run -d -p 8999:8080 -p 50500:50000 -v jenkins-data:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -u root --nam
```

2-2. Jenkins 컨테이너 내부에 Docker 설치

```
sudo docker exec -u 0 -it jenkins bash

# 패키지 업데이트
apt-get update

# sudo 패키지 설치
apt-get install sudo

# Docker 설치
sudo apt-get install docker.io
```

2-3. Jenkins 플러그인 설치

Jenkins 관리 → 플러그인 관리 → Available plugins → **NodeJs Plugin 설치 & GitLab Plugin 설치**

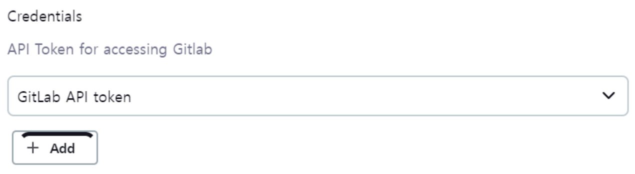
2-4. GitLab 연결 설정

Jenkins 관리 → 시스템 설정

Gitlab 항목에서 **Enable authentication for '/project' end-point** 체크

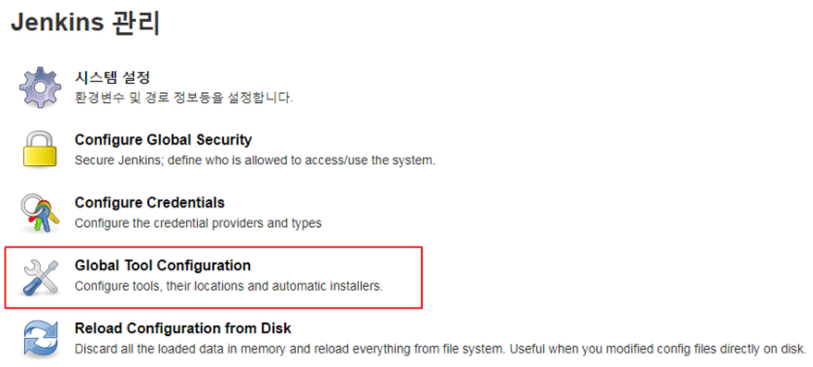
- Connection name : gitlab-connection
- Gitlab host URL : https://lab.ssafy.com
- Credentials
    - +ADD 클릭하여 Jenkins 클릭
        - Kind : GitLab API Token
        - Scope : Global
        - API token :
        - ID : gitlab-access-token

    - 드롭박스에서 GitLab API token 선택하여 적용


Credentials
API Token for accessing Gitlab
GitLab API token

2-5. Node 설치

Jenkins 관리 → Global Tool Configuration


Jenkins 관리

시스템 설정
환경변수 및 경로 정보등을 설정합니다.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Global Tool Configuration
Configure tools, their locations and automatic installers.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

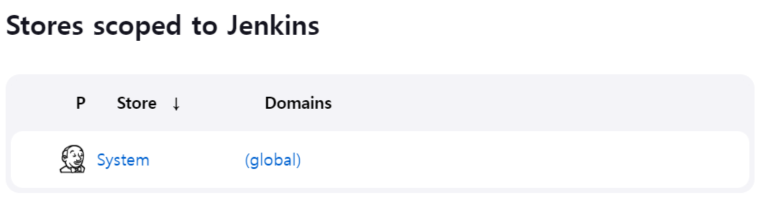
NodeJS 항목에서 **NodeJS installations** 클릭

- name : 18.12.1-LTS
- version : NodeJs 18.12.1

2-6. Username with password Credentials 추가

Jenkins 관리 → Manage Credentials

Domains의 (global) 클릭하여 +Add Credentials


Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

- Kind : Username with password
- Scope : Global
- Username :
- Password :
- ID : gitlab-access-account

2-7. Item 생성

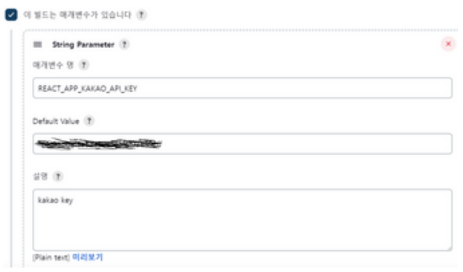
Enter an item name :

Pipeline 으로 생성

Configure 항목 클릭

- Git Repository와 연결 설정
    - GitLab Connetion 설정
        - gitlab-connection 선택
        - Use alternative credential 체크
            - Credential을 GitLab API token 선택
    - Build Trigger
        - **Build when a change is pushed to GitLab. GitLab webhook URL: http://j8a205.p.ssafy.io:8999/project/blommer** 체크

- 고급 버튼 클릭
    - 하단의 Secret token 부분에서 Generate 버튼 클릭하여 토큰 생성
- webhook URL 과 Secret token은 GitLab에서 Webhook 설정 시 필요
- 매개변수 설정해서 .env 파일 shell script를 생성



[✓] 이 빌드는 매개변수가 있습니다

String Parameter

매개변수 명: REACT_APP_KAKAO_API_KEY

Default Value: [redacted]

설명: kakao key

[Plain text] 미리보기

2-8. 스크립트 작성

Pipeline

```
pipeline {
   agent any

   tools {nodejs "18.12.1-LTS"}

   stages {
       stage('Gitlab') {
           steps {
               git branch: 'main', credentialsId: 'gitlab-access-account', url: 'https://lab.ssafy.com/s08-final/S08P31A404.git'
           }
       }
       stage('SpringBootBuild') {
           steps {
               dir('back-end') {
                   sh "sudo chmod 755 gradlew"
                   sh "./gradlew bootJar"
               }
           }
       }
       stage('SpringScheduleBuild') {
           steps {
               dir('back-schedule') {
                   sh "sudo chmod 755 gradlew"
                   sh "./gradlew bootJar"
               }
           }
       }
       stage('SpringRealTimeBuild') {
           steps {
               dir('back-realtime') {
                   sh "sudo chmod 755 gradlew"
                   sh "./gradlew bootJar"
               }
           }
       }
       stage('SpringMatchingBuild') {
           steps {
               dir('back-matching') {
                   sh "sudo chmod 755 gradlew"
                   sh "./gradlew bootJar"
               }
           }
       }
       stage('ReactBuild') {
           steps {
               dir('front-end') {
                   sh'''
                    ls -al
                    npm install --legacy-peer-deps
                    rm -f .env
                    printf "VITE_API_BASE_URL=${VITE_API_BASE_URL}\n" >> .env
                    cat .env
                    CI=false npm run build
                    '''
               }
```

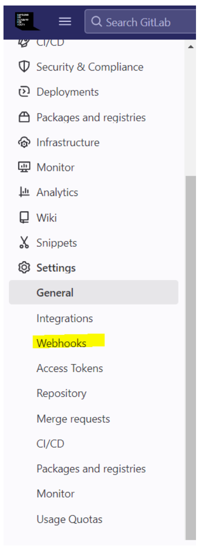
```
            }
        }
        stage('Build') {
             steps {
                 sh 'docker build -t estable-front ./front-end/'
                 sh 'docker build -t estable-back ./back-end/'
                 sh 'docker build -t estable-schedule ./back-schedule/'
                 sh 'docker build -t estable-realtime ./back-realtime/'
                 sh 'docker build -t estable-matching ./back-matching/'
             }
         }
         stage("Stop and Remove Front Container") {
             steps {
                 script {
                     def result = sh(returnStdout: true, script: "docker ps -q --filter name=front")
                     if (result.trim().length() > 0) {
                         sh "docker stop front"
                         sh "docker rm front"
                         echo "Container named 'front' has been stopped and removed."
                     } else {
                         echo "No container named 'front' was found."
                     }
                 }
             }
         }
         stage("Stop and Remove Back Container") {
             steps {
                 script {
                     def result = sh(returnStdout: true, script: "docker ps -q --filter name=back")
                     if (result.trim().length() > 0) {
                         sh "docker stop back"
                         sh "docker rm back"
                         echo "Container named 'back' has been stopped and removed."
                     } else {
                         echo "No container named 'back' was found."
                     }
                 }
             }
         }
         stage("Stop and Remove Schedule Container") {
             steps {
                 script {
                     def result = sh(returnStdout: true, script: "docker ps -q --filter name=back-schedule")
                     if (result.trim().length() > 0) {
                         sh "docker stop back-schedule"
                         sh "docker rm back-schedule"
                         echo "Container named 'back-schedule' has been stopped and removed."
                     } else {
                         echo "No container named 'back-schedule' was found."
                     }
                 }
             }
         }
         stage("Stop and Remove realtime Container") {
             steps {
                 script {
                     def result = sh(returnStdout: true, script: "docker ps -q --filter name=back-realtime")
                     if (result.trim().length() > 0) {
                         sh "docker stop back-realtime"
                         sh "docker rm back-realtime"
                         echo "Container named 'back-realtime' has been stopped and removed."
                     } else {
                         echo "No container named 'back-realtime' was found."
                     }
                 }
             }
         }
         stage("Stop and Remove matching Container") {
             steps {
                 script {
                     def result = sh(returnStdout: true, script: "docker ps -q --filter name=back-matching")
                     if (result.trim().length() > 0) {
                         sh "docker stop back-matching"
                         sh "docker rm back-matching"
                         echo "Container named 'back-matching' has been stopped and removed."
                     } else {
                         echo "No container named 'back-matching' was found."
                     }
                 }
             }
         }
         stage('Deploy') {
             steps{
                 sh 'docker run -d -p 5173:5173 --name front estable-front'
                 sh 'docker run -d -p 8080:8080 -e TZ=Asia/Seoul --name back estable-back'
                 sh 'docker run -d -p 8087:8087 -e TZ=Asia/Seoul --name back-schedule estable-schedule'
                 sh 'docker run -d -p 8083:8083 -e TZ=Asia/Seoul --name back-realtime estable-realtime'
```

```
                  sh 'docker run -d -p 8091:8091 -e TZ=Asia/Seoul --name back-matching estable-matching'
              }
          }
         stage('Finish') {
              steps{
                  sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
              }
          }
     }
  }
```

2-9. Webhook 설정

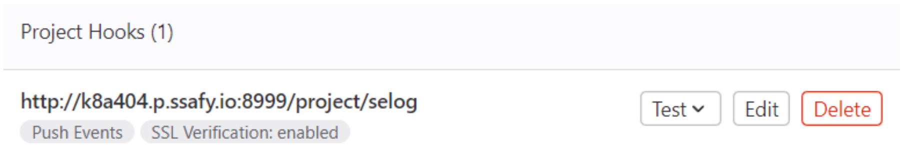
gitlab 접속하여 **S08P22A205** 프로젝트 클릭

`Settings` → `Webhook` 클릭



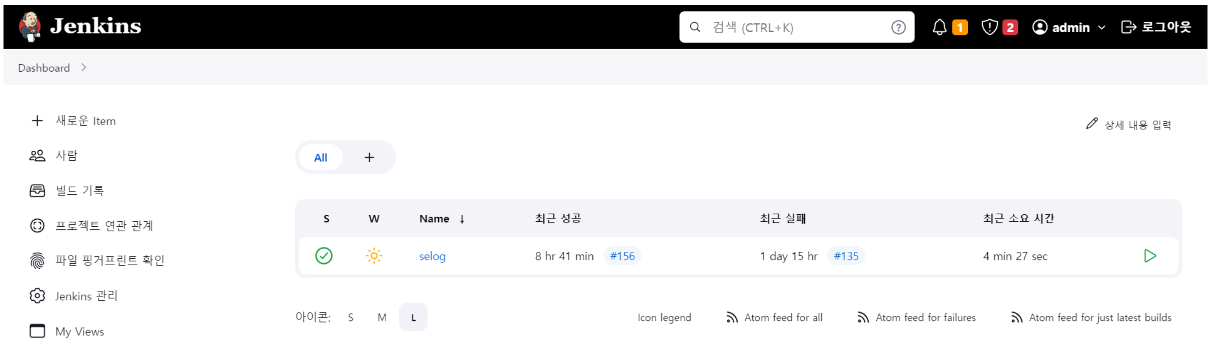
- `URL` : http://k8a404.p.ssafy.io:8999/project/selog
- `Secret token` : Configure에서 generate한 토큰
- `Trigger`
    - `Push events` : main

누르고 하단에 생성된 Webhook 확인


Project Hooks (1)

http://k8a404.p.ssafy.io:8999/project/selog
Push Events SSL Verification: enabled

2-10. Jenkins에서 빌드테스트



S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간
✓	☀	selog	8 hr 41 min #156	1 day 15 hr #135	4 min 27 sec

3. RabbitMQ

3-1. 설치

Docker 이미지 다운로드

```
docker pull rabbitmq:management
```

Docker Container 생성

```
docker run -d --network=host --name rabbitmq -p 5672:5672 -p 15672:15672 -d rabbitmq:management
```

3-2. 메시지 큐 생성

Bash 접속

```
docker exec -it rabbitmq bash
```

Queue 생성

```
rabbitmqadmin declare queue name=sellog.bad.queues
rabbitmqadmin declare queue name=sellog.queues
```

Exchange 생성

```
 rabbitmqadmin declare exchange name=sellog.exchange type=direct
```

Binding 생성

```
 rabbitmqadmin declare binding source="sellog.exchange" destination_type="queue" destination="sellog.bad.queues" routing_key="sellog.b
 rabbitmqadmin declare binding source="sellog.exchange" destination_type="queue" destination="sellog.queues" routing_key="sellog.routi
```

application.properties 설정

```
# RabbitMQ
spring.rabbitmq.host =
spring.rabbitmq.port = 5672
spring.rabbitmq.username =
spring.rabbitmq.password =
```

4. MYSQL

4-1. MYSQL 설치

도커 이미지 다운로드

```
docker pull mysql
```

컨테이너 생성

```
docker run --network=host --name mysql -e MYSQL_ROOT_PASSWORD=root -d -p 3306:3306 mysql:latest
```

4-2. MYSQL 환경설정

Bash 실행

```
docker exec -it mysql bash
```

Admin 접속

```
mysql -u root -p
// 입력 후 컨테이너 실행 시 사용했던 Password 입력
```

유저 생성 (예시 : ssafy)

```
# USER 생성, '%'는 모든 IP에서 접속 가능
mysql> CREATE USER ssafy@'%' identified by {비밀번호};
# 생성한 USER에 모든 권한 부여
mysql> GRANT ALL PRIVILEGES ON *.* to ssafy@'%';
# 변경 사항 적용
mysql> FLUSH PRIVILEGES;
mysql> exit;
```

ssafy 유저로 접속

```
mysql -u ssafy -p
// Enter password: {비밀번호}
```

Bloomer DB 생성

```
CREATE DATABASE bloomer;
SHOW DATABASES; // 확인
```

application.properties설정

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://j8a205.p.ssafy.io:3301/bloommer?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul
spring.datasource.username=
spring.datasource.password=
#spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
#spring.jpa.properties.hibernate.format_sql=true
```

Nginx Default값

/etc/nginx/sites-enabled/default

```
server {
        listen 80 default_server;
        listen [::]:80 default_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 _;
        location / {
                        # First attempt to serve request as file, then
                        # as directory, then fall back to displaying a 404.
                        try_files $uri $uri/ =404;
                }

                # pass PHP scripts to FastCGI server
                #
                #location ~ \.php$ {
                #       include snippets/fastcgi-php.conf;
                #
                #       # With php-fpm (or other unix sockets):
                #       fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
                #       # With php-cgi (or other tcp sockets):
                #       fastcgi_pass 127.0.0.1:9000;
                #}

                # deny access to .htaccess files, if Apache's document root
                # concurs with nginx's one
                #
                #location ~ /\.ht {
                #       deny all;
                #}
}
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 k8a404.p.ssafy.io; # managed by Certbot
        client_max_body_size 10M;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                proxy_pass http://localhost:5173;
        }

        location /api {
                proxy_pass http://localhost:8080;
        }
        location /oauth2 {
                proxy_pass http://localhost:8080;
        }
        location /login/oauth2 {
                        proxy_pass http://localhost:8080;
                }
        location /real-time {
                        proxy_pass http://localhost:8083;
                        proxy_http_version 1.1;
                        proxy_send_timeout 200;
                        proxy_set_header Upgrade $http_upgrade;
                        proxy_set_header Connection "Upgrade";
                        proxy_set_header Host $host;
                }
        location /matching {
                        proxy_pass http://localhost:8083;
                        proxy_http_version 1.1;
                        proxy_set_header Upgrade $http_upgrade;
                        proxy_set_header Connection "Upgrade";
                        proxy_set_header Host $host;
                }
                # pass PHP scripts to FastCGI server
                #
                #location ~ \.php$ {
                #       include snippets/fastcgi-php.conf;
                #
                #       # With php-fpm (or other unix sockets):
                #       fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
                #       # With php-cgi (or other tcp sockets):
                #       fastcgi_pass 127.0.0.1:9000;
                  #}
listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/k8a404.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/k8a404.p.ssafy.io/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = k8a404.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name k8a404.p.ssafy.io;
    return 404; # managed by Certbot
```

빌드 및 배포

1. FrontEnd

.env 파일 추가

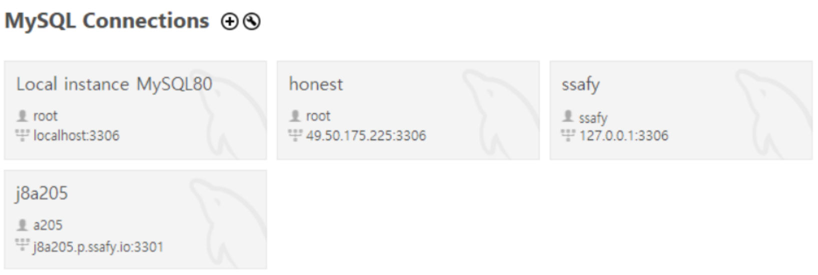
```
VITE_API_BASE_URL=
```

MYSQL WorkBench 사용방법

MySQL 8.0.32 설치 진행

https://dev.mysql.com/downloads/installer/

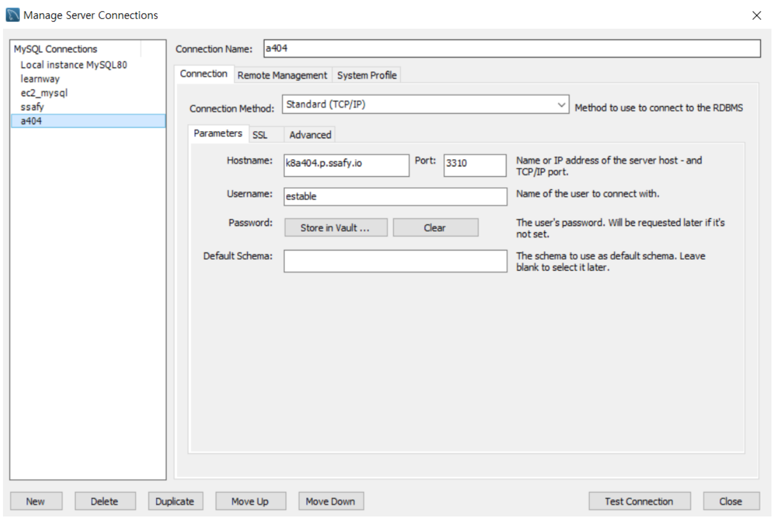
Workbench에서 Connection 생성



MySQL Connections

Local instance MySQL80 root localhost:3306	honest root 49.50.175.225:3306	ssafy ssafy 127.0.0.1:3306
j8a205 a205 j8a205.p.ssafy.io:3301		

k8a404.p.ssafy.io:3310



Manage Server Connections

MySQL Connections: Local instance MySQL80, learnway, ec2_mysql, ssafy, a404

Connection Name: a404

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

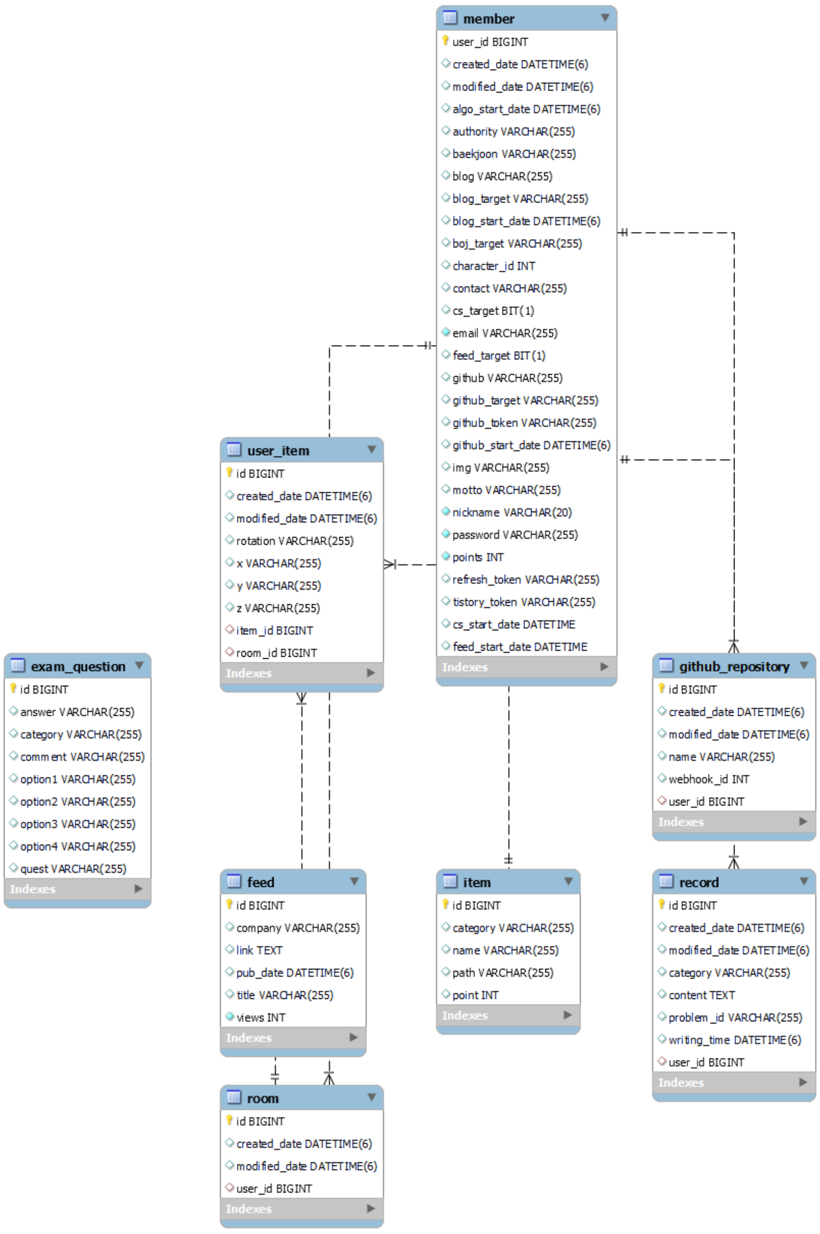
Hostname: k8a404.p.ssafy.io Port: 3310 — Name or IP address of the server host - and TCP/IP port.

Username: estable — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

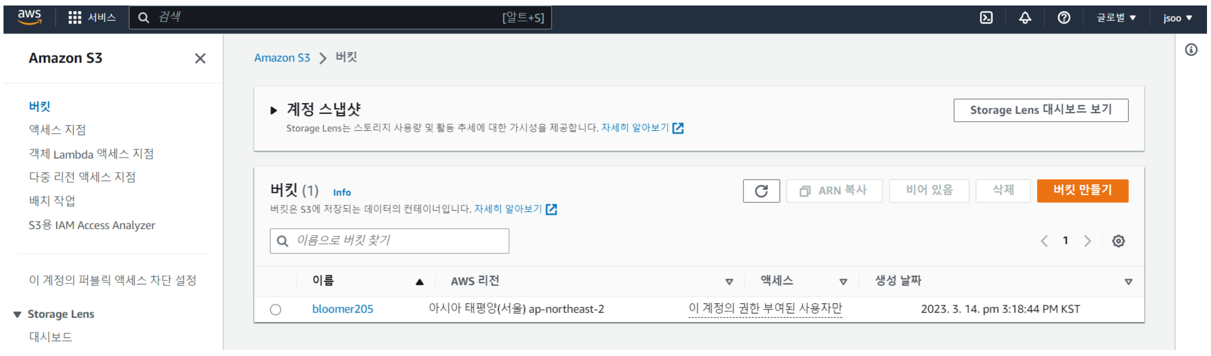
→ 계정 입력 완료 후 Test Connection에 성공하면 연결 성공



외부 서비스

1. AWS S3 Bucket

계정 생성 및 bucket 추가



계정 스냅샷

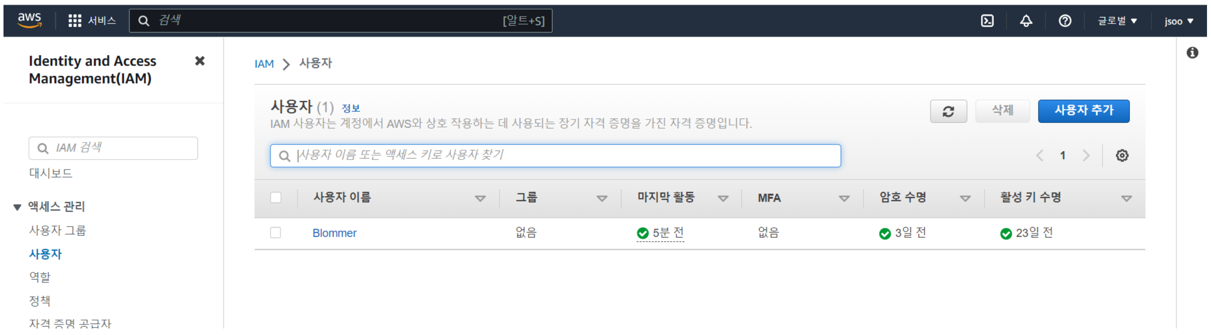
Storage Lens는 스토리지 사용량 및 활동 추세에 대한 가시성을 제공합니다. 자세히 알아보기

버킷 (1)

버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기

이름	AWS 리전	액세스	생성 날짜
bloomer205	아시아 태평양(서울) ap-northeast-2	이 계정의 권한 부여된 사용자만	2023. 3. 14. pm 3:18:44 PM KST

보안 자격 증명 > 엑세스키 생성



Identity and Access Management(IAM)

IAM 검색

대시보드

▾ 액세스 관리

사용자 그룹

사용자

역할

정책

자격 증명 공급자

IAM > 사용자

사용자 (1) 정보

IAM 사용자는 계정에서 AWS와 상호 작용하는 데 사용되는 장기 자격 증명을 가진 자격 증명입니다.

삭제 | 사용자 추가

사용자 이름 또는 액세스 키로 사용자 찾기

사용자 이름	그룹	마지막 활동	MFA	암호 수명	활성 키 수명
Blommer	없음	✓ 5분 전	없음	✓ 3일 전	✓ 23일 전

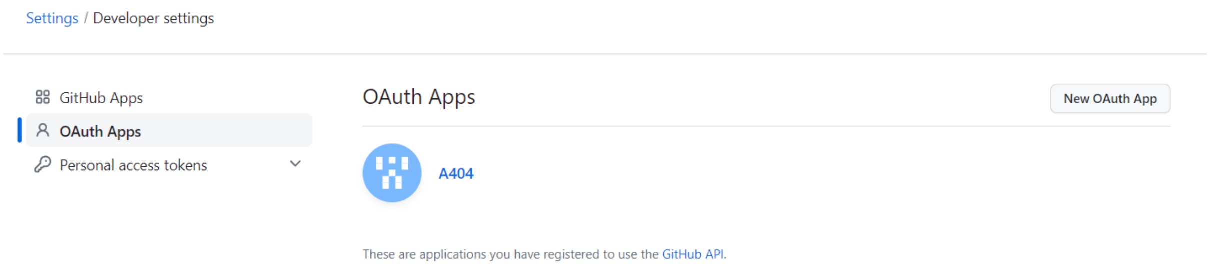
BackEnd - application.properties 추가

```
cloud.aws.credentials.access-key=
cloud.aws.credentials.secret-key=
cloud.aws.s3.bucket=
cloud.aws.stack.auto=false
logging.level.com.amazonaws.util.EC2MetadataUtils=error

spring.servlet.multipart.maxFileSize=10MB
spring.servlet.multipart.maxRequestSize=10MB
```

2. Github

- 애플리케이션 추가


Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

OAuth Apps

New OAuth App

A404

These are applications you have registered to use the GitHub API.

```
spring.security.oauth2.client.registration.github.client-id=ENC(vbvkZdN7toXnGUb6HkZps3PrnQ2oZ6l89meNWgxMg6Oso0chnn2ibuFcJjldtylAz3uhTz
spring.security.oauth2.client.registration.github.client-secret=ENC(xNvmU9gxOvN1Vx8WUBfxG9H1rz1IVTB3eg/oAM3IzrgBCLQkcAn133+4h5tbMval3t
spring.security.oauth2.client.registration.github.scope=repo,user:email,read:user
spring.security.oauth2.client.registration.github.redirect-uri=http://k8a404.p.ssafy.io:8080/login/oauth2/code/github
```

3. ChatGPT

```
OPEN_AI_KEY=ENC(1cB68S1hpUWz3FDLrUwb3wBVlB5cqEiIuMLm44lAz0FTkG5fMlTUT84Vcxvph66rfuwtHcUXEhbF80LxrNMu7IVHjU1zlDbtSQGwYXrcUaNVRiKNfRrU9j
```

배포 명령어

기본 명령어

```
// 실행중인 컨테이너
sudo docker ps -a

// 다운받은 이미지 목록
sudo docker image ls

// 이미지 생성
sudo docker build -t {이미지이름} .

// 이미지 삭제
sudo docker rmi {이미지이름}

// 네트워크
sudo docker network create
sudo docker network connect {network_name} {container_name}

// 컨테이너 생성 & 실행
sudo docker run --name={컨테이너이름} {hostPort}:{containerPort} {이미지이름}:{버전}

// 컨테이너 삭제
sudo docker rm {컨테이너아이디}
```