판교로가조 포팅 메뉴얼

CD/CI

배포 환경 구성

- nginx 1.18.0
- jenkins(docker image) 2.375 latest
- nohup 8.30
- docker 20.10.23
- snparqube Community Edition Version 9.8

sonarQube 를 통해 코드 정적검사를 수행하고 젠킨스를 통해 빌드 `sshPublisher` 를 활용해 배포를 진행합니다.

sonarQube

sonarQube 의 경우 postgres 디비를 별도로 구성시켜야 하기에 dockerCompose를 통해 구성 했습니다

다음 파일을 `docker-compose up -d` 명령어로 실행해야합니다. 만약 dockerCompose가 없다면 설치하고 수행합니다.

▼ sonarqube/docker-compose.yml

```
version: "3.1"
services:
  sonarqube:
    image: sonarqube:latest
    container_name: sonarqube
    ports:
      - "9000:9000"
      - "9092:9092"
    networks:
      - sonarnet
    environment:
      - SONARQUBE_HOME=/opt/sonarqube
      - SONARQUBE_JDBC_USERNAME=sonar
      - SONARQUBE_JDBC_PASSWORD=sonar
      - SONARQUBE_JDBC_URL=jdbc:postgresql://db:5432/sonar
    volumes:
      - /volums/sonarqube/conf:/opt/sonarqube/conf
      - /volums/sonarqube/data:/opt/sonarqube/data
      - /volums/sonarqube/logs:/opt/sonarqube/logs
      - /volums/sonarqube/extensions:/opt/sonarqube/extensions

  db:
    image: postgres
    container_name: postgres
    networks:
      - sonarnet
    environment:
      - POSTGRES_USER=sonar
      - POSTGRES_PASSWORD=sonar
    volumes:
      - /volumns/sonarqube/postgres:/var/lib/postgresql/data

networks:
  sonarnet:
    driver: bridge
```

jenkins 설정

젠킨스 cd/ci 설정

벡엔드

백엔드에 연결할 mysql 을 aws 에 올리고 백엔드를 실행하는 구조로 진행됩니다.

벡엔드 구성

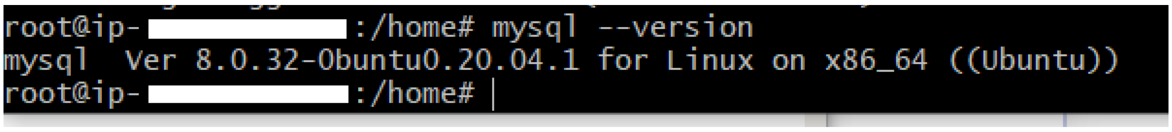
- java 11 버전
- spring boot 2.7
- gradle
- mysql 8.0.32-0ubuntu0.20.04.2
- jpa-hibernate 5.6
- spring security 5.7

설정진행

▼ 1. aws Mysql 설정

```
//su 명령으로 관리자 권한으로 접속 후
apt update
apt install mysql-server
```

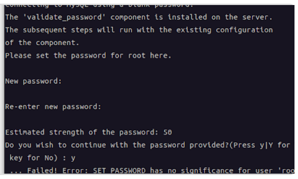
- 설치 확인



```
root@ip-[redacted]:/home# mysql --version
mysql  Ver 8.0.32-0buntu0.20.04.1 for Linux on x86_64 ((Ubuntu))
root@ip-[redacted]:/home# |
```

- 초기설정은 다음 홈페이지 참조

[Ubuntu] 우분투 MySQL 비밀번호 설정 오류 해결하기 Resolving Ubuntu MySQL Password Setting Errors
인데 하다가 sudo mysql_secure_installation 했다가 오류걸리신 분 아래↓↓↓ 일단 왜 오류나는지 설명을 해드리자면 secure_installation에서 비밀번호를 설정하면 root 권한이 아닌 상태로 비밀번호를 설정했기 때문에 root 권한으로 비밀번호를 설정하라는 뜻.. 우분투 내에서 MySQL 비밀번호 설정하는데 아래처럼 오류가 걸리신다고요? 해결해드리지요!! ... Failed!
https://www.bddungsblog.com/2022/11/ubuntu-mysql-resolving-mysql-password.html

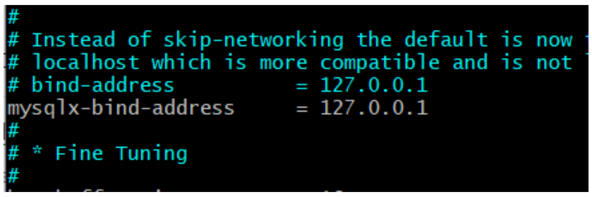


```
The 'validate_password' component is installed on the server.
The subsequent steps will run with the existing configuration
of the component.
Please set the password for root here.

New password:

Re-enter new password:

Estimated strength of the password: 50
Do you wish to continue with the password provided?(Press y|Y for
 key for No) : y
 ... Failed! Error: SET PASSWORD has no significance for user 'roo
```

이제 MySQL을 워크벤치(외부)에서 접속을 하기 위한 설정을 해줘야 한다.

```
cd/etc/mysql/mysql.conf.d
vi mysqld.cnf
```

- bind-address 127.0.0.1 가 적힌 줄 맨앞에 # 를 넣어 주석처리 해주기



```
#
# Instead of skip-networking the default is now
# localhost which is more compatible and is not
# bind-address           = 127.0.0.1
mysqlx-bind-address      = 127.0.0.1
#
# * Fine Tuning
#
```

```
/usr/bin/mysql -u root -p
```

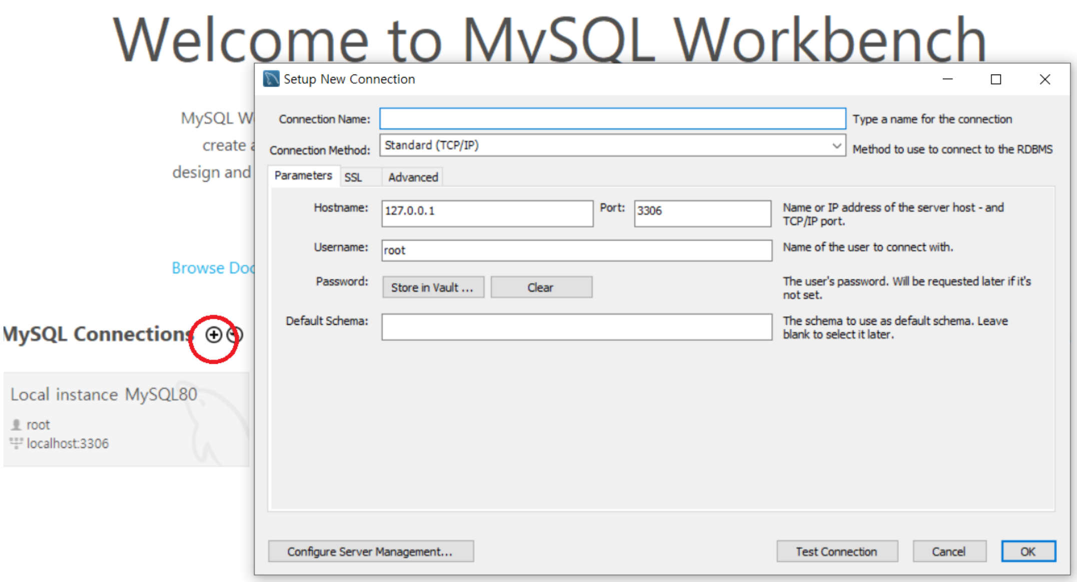
- 외부접속 허용 sql문 입력

```
mysql> create user '유저명'@'%' identified by '비밀번호';
mysql> grant all privileges on *.* to '유저명'@'%' with grant option;
```

- mysql 재시작

```
service mysql restart
ufw allow out 3306/tcp
ufw allow in 3306/tcp
service mysql restart
```

- 이제 생성한 사용자로 외부 개발 환경 워크벤치에서 접속



Welcome to MySQL Workbench

MySQL Connections ⊕ ⊗

Local instance MySQL80
root
localhost:3306

Setup New Connection

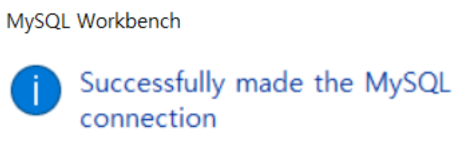
Connection Name:		Type a name for the connection
Connection Method:	Standard (TCP/IP)	Method to use to connect to the RDBMS

Parameters | SSL | Advanced

Hostname:	127.0.0.1 Port: 3306	Name or IP address of the server host - and TCP/IP port.
Username:	root	Name of the user to connect with.
Password:	Store in Vault ... Clear	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.

- + 버튼을 눌러 Connection 추가
- Hostname에 조별로 발급받은 서버도메인 입력

- 위에서 생성한 user의 Username과 password를 각각 Username과 password에 입력
- Test Connection!!


MySQL Workbench

Successfully made the MySQL connection

▼ 2. 개발 환경 파일

application-dev.ym, env.properties 는 누출되어선 안됩니다

env 의 경우 배포 는 op-in git 계정 , 개발은 개인 계정이 사용되었습니다.

개인계정으로 다시 설정할 경우 다음문서를 참조하시면됩니다.

🏁 깃허브 OAuth 계정 설정

▼ .gitignore

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

### STS ###
.apt_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
/src/main/resources/properties/env.properties
/src/main/resources/static/
```

▼ resources/application.yml

```
spring:
  profiles:
    active: 'dev'
  devtools:
    livereload:
      enabled: true
  datasource:
    url:
    username:
    password:
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        use_sql_comments: true
    hibernate:
      ddl-auto: update
```

▼ resources/application-dev.yml

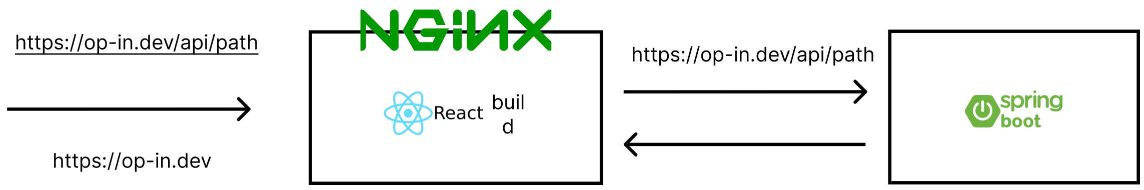
```
spring:
  thymeleaf:
    suffix: .html
    prefix: classpath:/templates/
    check-template-location: true
    cache: false
    mode: HTML
  batch:
    job:
      enabled: false
    jdbc:
      initialize-schema: ALWAYS

  security:
    user:
      name: ssafy
      password: ssafy
  devtools:
    livereload:
      enabled: true
  datasource:
    url: jdbc:mysql://"aws주소"/"mysql디비이름"?serverTimezone=UTC&useUniCode=yes&characterEncoding=UTF-8
    username: "외부접속 등록 아이디"
    password: "외부접속 등록 비밀번호"
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      connection-timeout: 30000
      maximum-pool-size: 10
  mail:
    host: smtp.gmail.com
    port: 587
    username: opinmails
    password: jaoprdilxqwmptlz
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true


  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        use_sql_comments: true
    hibernate:
      ddl-auto: update
server:
  port: 5001


jwt:
  header: Authorization
  algorithm: "HS256"
  #HS512 알고리즘을 사용할 것이기 때문에 512bit, 즉 64byte 이상의 secret key를 사용해야 한다.
  #echo 'spring-boot-security-jwt-tutorial-opensourceCommunity-spring-boot-security-jwt'|base64
  secret: c3ByaW5nLWJvb3Qtc2VjdXJpdHktand0LXR1dG9yaWFsLW9wZW5zb3VyY2VDb21tdW5pdHktc3ByaW5nLWJvb3Qtc2VjdXJpdHktand0
  access-token-validity-in-seconds: 86400 # 1일
  refresh-token-validity-in-seconds: 2592000 # 30일

cloud:
  aws:
    credentials:
      accessKey: AKIA5XUACFAWPCASMRX2
      secretKey: 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
    s3:
      bucket: pangyo
    region:
      static: ap-northeast-2
    stack:
      auto: false
githubToken1: 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
githubToken2: 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
githubToken3: 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
```

▼ opInBackEnd/src/main/resources/properties/env.properties

```
security.oauth.github.client-id=6eba3453001cf6a9b2bb
security.oauth.github.client-secret=4195d5677992d3ac6b5798c2f13acd4e191499e7
```

▼ 3. 배포환경

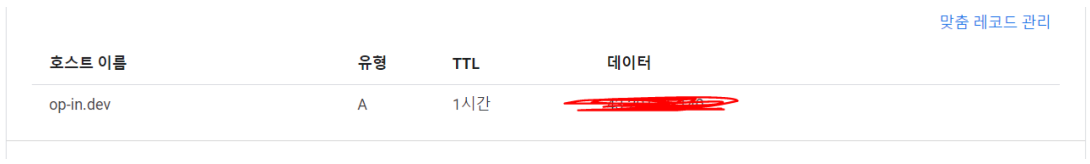
- ec2 443 (https) 포트는 프론트 빌드파일을 바라보게 설정했습니다.
- 배포환경의 경우 nginx 가 프론트의 `WebServer` 역활을합니다
- api 요청은 nginx의 `proxy pass` 를 통해 `api/` 라고 명시된주소를 내부 백엔드로 프록시걸어 주었습니다.



도메인 구매

해당 프로젝트는 구글 도메인을 통해 도메인을 구매 했으며

다음과 같이 등록 하면됩니다


맞춤 레코드 관리

호스트 이름	유형	TTL	데이터
op-in.dev	A	1시간	[redacted]

NGINX

nginx 를 설치하고 랫츠 인크립트 인증을 받고 다음 두가지파일을 수정하면됩니다.

랫츠 인크립트 서트봇을 통해 도메인에 ssl을 인증 받고 다음 두가지과정을 수행해야합니다.

▼ certbot ssl 발급 과정

```
sudo add-apt-repository ppa:certbot/certbot
```

```
sudo apt install python-certbot-nginx
```

```
sudo certbot --nginx -d 도메인이름
```

이메일 주소→ agree→yes→2

▼ /etc/nginx/conf.d/opinFront.conf

```
server{
    listen 443;
    ssl on;
    listen [::]:443;

    # listen 80;
    # listen [::]:80;
    server_name i8c211.p.ssafy.io;
    # server_name op-in.dev www.op-in.dev;

    ssl_certificate /etc/letsencrypt/live/op-in.dev/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/op-in.dev/privkey.pem;

    # root /var/www/opin;
    root /home/ubuntu/jenkins_build/opinFrontBuild;
#    index index.html index.htm;

    location / {
        try_files $uri /index.html;
    }
    location /api {
        proxy_pass http://i8c211.p.ssafy.io:5001;
    }

   # location ~ ^/(assets)/ {
# alias /var/www/opin/assets/;
 #  }

}
```

▼ /etc/nginx/nginx.conf

```
user root;
#user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
  worker_connections 768;
  # multi_accept on;
}

http {

  ##
  # Basic Settings
  ##
  client_max_body_size 5M;
  sendfile on;
  tcp_nopush on;
  tcp_nodelay on;
  keepalive_timeout 65;
  types_hash_max_size 2048;
  # server_tokens off;

  server_names_hash_bucket_size 64;
  # server_name_in_redirect off;

  include /etc/nginx/mime.types;
  default_type application/octet-stream;

  ##
  # SSL Settings
  ##

  ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
  ssl_prefer_server_ciphers on;

  ##
  # Logging Settings
  ##

  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;

  ##
  # Gzip Settings
  ##

  gzip on;

  # gzip_vary on;
  # gzip_proxied any;
  # gzip_comp_level 6;
  # gzip_buffers 16 8k;
  # gzip_http_version 1.1;
  # gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/

  ##
  # Virtual Host Configs
  ##

  include /etc/nginx/conf.d/*.conf;
  # include /etc/nginx/sites-enabled/*;
}


#mail {
# # See sample authentication script at:
# # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
# # auth_http localhost/auth.php;
# # pop3_capabilities "TOP" "USER";
# # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
# server {
#   listen     localhost:110;
#   protocol   pop3;
#   proxy      on;
# }
#
# server {
#   listen     localhost:143;
#   protocol   imap;
#   proxy      on;
# }
#}
```

▼ 4. 배포 환경 파일

▼ resources/application-dev.yml

```
spring:
  batch:
    job:
      enabled: false
    jdbc:
      initialize-schema: ALWAYS
  mvc:
    throw-exception-if-no-handler-found: true
  web:
    resources:
      add-mappings: false
  security:
    user:
      name: ssafy
      password: ssafy
  devtools:
    livereload:
      enabled: true
  datasource:
    url: jdbc:mysql://localhost/dev?serverTimezone=UTC&useUniCode=yes&characterEncoding=UTF-8
    username: ssafy3
    password: ssafy
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      connection-timeout: 30000
      maximum-pool-size: 10
  mail:
    host: smtp.gmail.com
    port: 587
    username: opinmails
    password: jaoprdilxqwmptlz
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true

  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        use_sql_comments: true
    hibernate:
      ddl-auto: update
```

```
server:
  port: 5001
  # ssl:
  #   key-store-type: PKCS12
  #   key-store-password: ssafyssafy
  #   key-store: classpath:keystore.p12
  #   key-alias: tomcat


jwt:
  header: Authorization
  algorithm: "HS256"
  #HS512 알고리즘을 사용할 것이기 때문에 512bit, 즉 64byte 이상의 secret key를 사용해야 한다.
  #echo 'spring-boot-security-jwt-tutorial-opensourceCommunity-spring-boot-security-jwt'|base64
  secret: c3ByaW5nLWJvb3Qtc2VjdXJpdHktand0LXR1dG9yaWFsLW9wZW5zb3VyY2VDb21tdW5pdHktc3ByaW5nLWJvb3Qtc2VjdXJpdHktand0
  access-token-validity-in-seconds: 86400 # 1일
  refresh-token-validity-in-seconds: 2592000 # 30일



cloud:
  aws:
    credentials:
      accessKey: AKIA5XUACFAWPCASMRX2
      secretKey: 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
    s3:
      bucket: pangyo
    region:
      static: ap-northeast-2
    stack:
      auto: false


githubToken1 : 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
githubToken2 : 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
githubToken3 : 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
```

▼ opInBackEnd/src/main/resources/properties/env.properties

```
security.oauth.github.client-id=b61565834227668e9069
security.oauth.github.client-secret=3962de5e1e6492bdde479fc63725da0d4191fa0e
```

프론트엔드

사용 도구

- Visual Studio Code(v1.74.2)

환경

- node (v16.19)
- npm (v8.19.3)

빌드도구

- Vite (v4)

패키지 구성

- React 18
- Recoil
- tailwind css
- 기타 라이브러리(package.json 참고)

패키지 설치

- `npm install --legacy-peer-deps`
- `--legacy-peer-deps` 는 왜 사용하나요?
    - `toast-ui/editor` 에서 React 18에 대한 호환성을 설정해주지 않았기 때문입니다.

빌드 방법

- `npm run build`

작성자

김명진 — krocd@naver.com