포팅메뉴얼

태그	김광표
다중 선택	Not started

버전 정보

백엔드

Name	Tags	version
Java 열기	language	JDK 17
Spring Boot	framework	3.1.x
SpringSecurity	library	
Gradle		
JPA	framework	
QueryDSL	framework	
Redis	NoSQL	
MYSQL	RDBMS	
S3	DB	
Elasticsearch		
EC2		
JENKINS		JDK 17
Nginx		
Docker		

프론트엔드

Name	Tags	version
React 열기		18.2.0
prettier	formatter	
react-router-dom	library	6.14.2
react-dom	library	18.2.0
ant-design	library	5.10.1
axios	library	1.4.0
react-cookies	library	

사용 도메인

https://ssafywiki.info

프론트엔드 주소 : ssafywiki.info
스프링 스웨거 주소 :ssafywiki.info:8080/swagger-ui.html
젠킨스 주소 : antoday.site:8000

엘라스틱서치 : https://k9e202a.p.ssafy.io/test_v1.0.0/

배포 매뉴얼

도커 환경 설정

도커 설치

```
$ sudo apt-get update
$ sudo apt-get -y upgrade
$ curl -fsSL https://get.docker.com/ | sudo sh

...

$ docker --version
# 권한설정
$ sudo usermod -aG docker $USER
$ sudo service docker restart

# 아래는 재로그인(연결을 종료했다 다시 연결해도 된다)
$ sudo su
$ sudo su ubuntu

$ docker ps
```

도커 컴포즈 설치

```
$ sudo curl -L https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/doc

$ sudo chmod +x /usr/local/bin/docker-compose

$ docker-compose --version
```

젠킨스

젠킨스 설치 및 도커 컴포즈 설정

```
$ mkdir compose && cd compose
$ mkdir jenkins-dockerfile && cd jenkins-dockerfile
$ vim Dockerfile

...
FROM jenkins/jenkins:lts

USER root
RUN apt-get update &&\
    apt-get upgrade -y &&\
    apt-get install -y openssh-client
...

$ cd ..
$ vim docker-compose.yml

...
version: "3"
services:
  jenkins:
    container_name: jenkins-compose
    build:
      context: jenkins-dockerfile
      dockerfile: Dockerfile
    user: root
    ports:
      - 8000:8080
      - 8888:50000
    volumes:
      - /home/ubuntu/compose/jenkins:/var/jenkins_home
      - /home/ubuntu/compose/.ssh:/root/.ssh
...

$ mkdir jenkins
$ mkdir .ssh
```

젠킨스 도커 이미지 빌드

```
docker-compose up --build -d

$ docker logs jenkins-compose # 비밀번호 출력
```

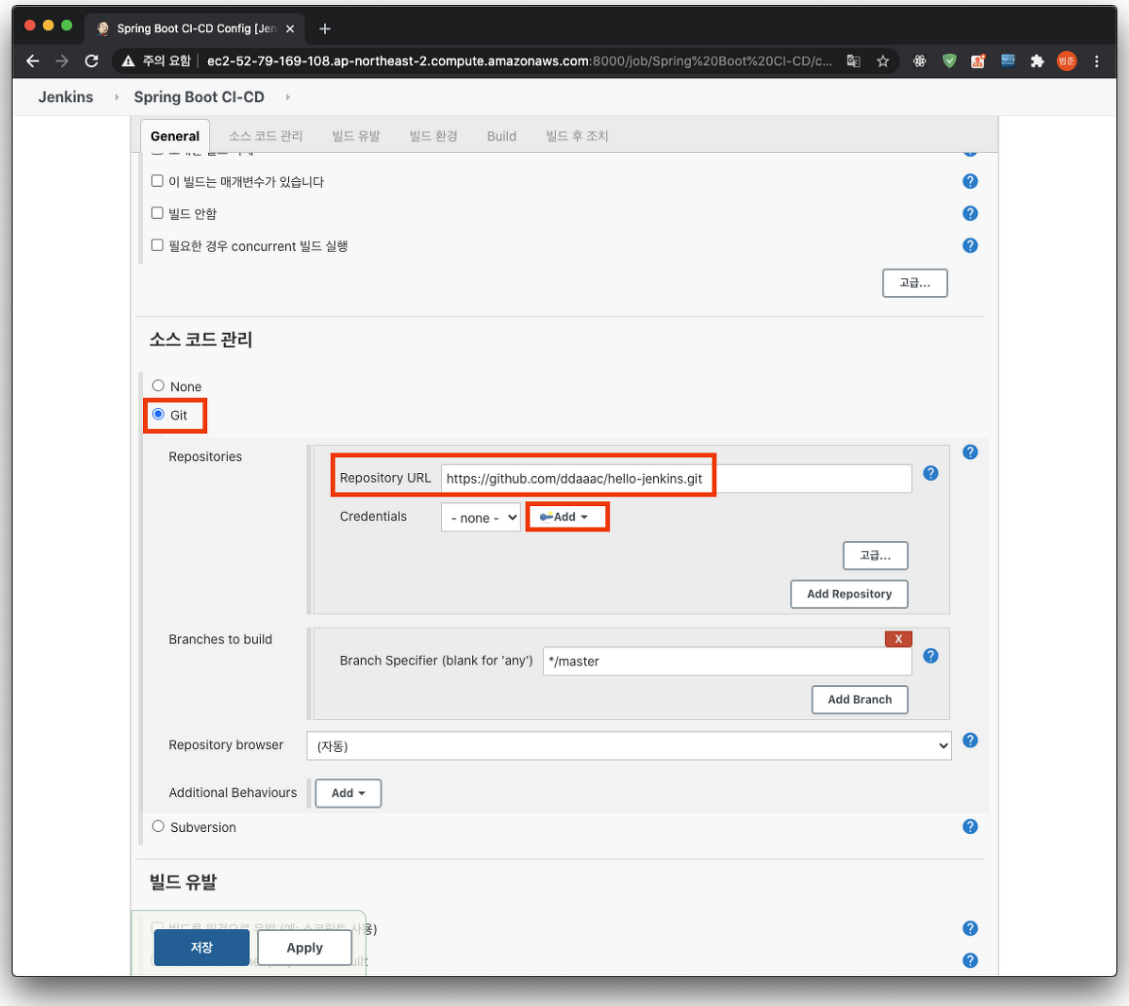
깃랩 설정

젠킨스에 gitlab 플러그인 설치

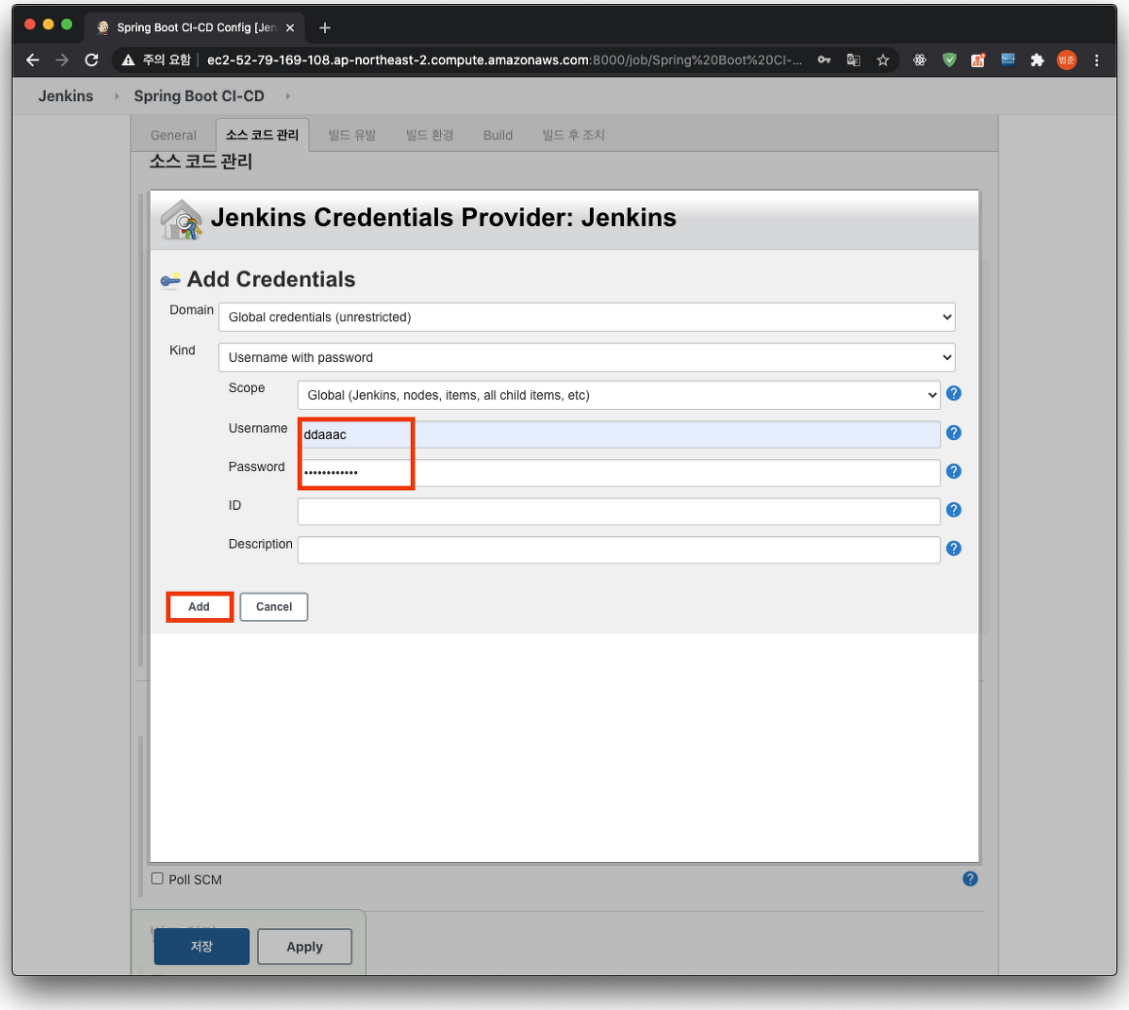
gitlab credential 추가

아이템 생성

프리스타일 생성



깃랩 저장소 등록



계정 등록

깃랩 토큰 등록

깃랩 웹훅 등록

```
FROM openjdk:17-jdk
ENTRYPOINT java -jar /deploy/antoday-0.0.1-SNAPSHOT.jar
ENV SPRING_PROFILES_ACTIVE=prod // 환경변수 등록
EXPOSE 8080
```

jenkins 컨테이너 키 등록

```
$ ssh-keygen -t rsa

$ cat /root/.ssh/id_rsa.pub

$ exit
```

```
$ vim ~/.ssh/authorized_keys // 이후 위에서 얻은 키 붙여넣기

$ docker exec -it jenkins-compose bash

$ ssh ubuntu@우분투주소 // 접속 확인
```

최종 Docker-compose.yml

```
version: "3"
services:
        redis:
                image: redis:latest
                container_name: redis-compose
                user: root
                ports:
                        - 6379:6379
                volumes:
                        - ./redis/data:/data
                        - ./redis/conf/redis.conf:/usr/local/conf/redis.conf
                labels:
                        - "name=redis"
                        - "mode-standalone"
                restart: always
                command: redis-server /usr/local/conf/redis.conf
                environment:
                        - TZ=Asia/Seoul
        jenkins:
                container_name: jenkins-compose
                build:
                        context: jenkins-dockerfile
                        dockerfile: Dockerfile
                user: root
                ports:
                        - 8000:8080
                        - 8888:50000
                volumes:
                        - /home/ubuntu/compose/jenkins:/var/jenkins_home
                        - /home/ubuntu/compose/.ssh:/root/.ssh
                environment:
                        - TZ=Asia/Seoul
        spring:
                container_name: spring-compose
                build:
                        context: spring-dockerfile
                        dockerfile: Dockerfile
                ports:
                        - 8080:8080
                volumes:
                        - /home/ubuntu/compose/jenkins/workspace/SSAFYWIKIBackEnd/ssafywiki_back/ssafywiki/build/libs:/deploy
                depends_on:
                        - redis
                environment:
                        - TZ=Asia/Seoul
        #react:
        #       container_name: react-compose
        #       build:
        #               context: react-dockerfile
        #               dockerfile: Dockerfile
        #       ports:
        #               - 3000:3000
        #       volumes:
        #               - /home/ubuntu/compose/jenkins/workspace/SSAFYWIKIFRONT/ssafywiki_front/build:/deploy
~
```

도메인 설정 및 SSL 인증

SSLForFree, CertBot 등의 서비스를 사용해 도메인에 SSL 인증

yml, .env 파일

- yml

```
server:
  port: 8080

spring:
  mail:
    host: smtp.gmail.com
    port: 587
    username: bdose202@gmail.com
    password: vnwe ejis wrpd emog
    properties:
      mail:
```

```
        smtp:
          auth: true
          timeout: 5000
          starttls:
            enable: true
  redis:
    host: localhost
    port: 6739
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        show_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect

  datasource:
    url: jdbc:mysql://database-ssafywiki.cnbro2eo1odc.ap-northeast-2.rds.amazonaws.com/ssafywiki?createDatabaseIfNotExist=true&useSSL=f
    username: admin
    password: ssafywiki!
    driver-class-name: com.mysql.cj.jdbc.Driver

  springdoc:
    default-consumes-media-type: application/json;charset=UTF-8
    default-produces-media-type: application/json;charset=UTF-8
    paths-to-match: /**
    swagger-ui:
      path: /
      display-request-duration: true
      groups-order: desc
      operations-sorter: alpha
      disable-swagger-default-url: true
    api-docs:
      groups:
        enabled: true

  logging:
    level:
      org.hibernate.sql: debug
      org.hibernate.type: trace
```

- .env

```
REACT_APP_SERVER_API_URL="https://ssafywiki.info/"
REACT_APP_SERVER_WS_URL="wss://ssafywiki.info/ws"
REACT_APP_AWS_ACCESS_KEY="AKIA2NHZIQWCC4JGFZBM"
REACT_APP_AWS_SECRET_KEY="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"
AWS_SDK_LOAD_CONFIG=1
```

nginx 설정

```
autoindex_localtime on;
server {


        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        server_name ssafywiki.info www.ssafywiki.info;
        location / {
                root    /home/ubuntu/compose/jenkins/workspace/SSAFYWIKIFRONT/ssafywiki_front/build;
                index   index.html index.htm;
                try_files $uri $uri/ /index.html;

        }
        location /redis {
                proxy_pass http://localhost:6379;
                proxy_set_header X-Real_IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }
        location /api {
                proxy_pass http://localhost:8080;
                proxy_set_header X-Real_IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
```

```
        }

        location /ws {
            # WebSocket 서버로 프록시 설정
                proxy_pass http://localhost:8080;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header Host $host;
        }
        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/ssafywiki.info/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/ssafywiki.info/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 ($host = www.ssafywiki.info) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = ssafywiki.info) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 default_server;
        listen [::]:80 default_server;

        server_name ssafywiki.info www.ssafywiki.info;
    return 404; # managed by Certbot

}
```

엘라스틱 서치 메뉴얼

- 엘라스틱 캐시의 인덱스를 만드는데에 필요한 설정 파일 입니다.
  - Nori : 한글 형태소 분석기
    - 엘라스틱서치에 노리 플러그인을 설치합니다.

```
./bin/elasticsearch-plugin install analysis-nori
```

  - Ngram : 단어를 N개씩 분리하여 찾게 해줍니다. 1개로 설정하여 3글자 이름을 잘 찾게 만듧니다.

이게 없으면 “권선근” 을 찾기위해 “권선” , “권선근” 까지 검색해야 할 가능성이 높습니다.

  - docs_id : docs_title 두 컬럼을 사용합니다. 제목으로 검색해서 id를 찾는 목적으로 사용합니다.

```
cd /usr/share/elasticsearch
// 여기에 mapping.json 파일이 있어야 한다.
curl -XPUT localhost:9200/test_v1.0.0?pretty=true -d @mapping.json -H "Content-Type:application/json"
// mapping.json으로 인덱스 test_v1.0.0를 생성한다.
elasticsearch-setup-passwords auto
// 엘라스틱서치에 사용될 비밀번호를 자동생성한다.
etc/logstash 에
xpack.monitoring.elasticsearch.username: "elastic"
xpack.monitoring.elasticsearch.password: "생성된 비밀번호"
추가할것
```

  -

▼ ssafy.conf

```
cd /etc/logstash/conf.d
sudo vim ssafy.conf


input {
```

```
  jdbc {
    jdbc_driver_library => "/opt/logstash/vendor/jar/jdbc/mysql-connector-java-8.0.18.jar"
    jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://database-ssafywiki.cnbro2eo1odc.ap-northeast-2.rds.amazonaws.com:3306/ssafywiki"
    jdbc_user => "admin"
    jdbc_password => "ssafywiki!"
    jdbc_paging_enabled => true
    tracking_column => "unix_ts_in_secs"
    use_column_value => true
    clean_run => true
    tracking_column_type => "numeric"
    schedule => "*/5 * * * * *"
    statement => "SELECT *, UNIX_TIMESTAMP(docs_modified_at) AS unix_ts_in_secs FROM ssafywiki.documents WHERE (UNIX_TIMESTAMP(doc
  }
}

filter {

}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "test_v1.0.0"
    user => "elastic"
    password => "g0UgjA6o5fh2PIFCGhrw"
    document_id => "%{docs_id}"
  } stdout {
    codec => rubydebug
  }
}
```

- Logstash 설정 파일입니다.
  - DB에서 SELECT문을 통해 문서 데이터를 가져오고 있습니다.
  - 문서 ID로 중복을 방지합니다.
  - SELECT 하여 가져오는 문서들중 최신 수정시간 보다 이후인 것만 가져옵니다.
  - 로그스태시를 재실행 하면 다 가져옵니다
  - jdbc에는 대상 DB 정보를, output에는 엘라스틱 서치의 인덱스 정보를 씁니다.
  - output의 document_id는 DB의 문서 테이블의 아이디를 뜻하는게 아니라 엘라스틱 서치가 관리하는 테이블의 ID라고 생각하면 됩니다. 이것을 우리 문서테이블의 docs_id로 할당시킵니다.

  명령어

  sudo systemctl [restart,status,stop] [elasticsearch , logstash, kibana]

  문제 발생시 대처법:

```
cd /usr/share/elasticsearch
// 여기에 mapping.json 파일이 있어야 한다.
curl -XPUT localhost:9200/test_v1.0.0?pretty=true -d @mapping.json -H "Content-Type:application/json"
// 인덱스 생성완료
sudo systemctl restart logstash
// 시간 1~2분 걸림... 기다릴것.....재시작 반복 하면 엿됨.
sudo systemctl status logstash
// 이걸로 Select문 가져오는거 체크 할것... 나오는데 좀 걸릴것임

//Nginx , logsatsh, elasticsearch status 확인해서 문제 있는거 해결할것
//sudo ufw status 에서 막아놓은거 풀어 볼것
//키바나 포트 풀면 웹에서 접근 => 검색창에 index 치면 인덱스 목록, 설정 확인 가능

// 다 정상인데 검색이 안된다????
// 인덱스가 제대로 생성 되지 않았을 가능성 있음
// 키바나에서 인덱스 매핑 확인하면 우리의 mapping.json 처럼 안되있을 가능성 있음
// 추측으로는 로그스태시가 자동으로 생성하는것.
// 따라서 로그스태시를 끄고 인덱스를 삭제 후 생성 , 로그스태시 재시작 할것
```

  NGINX 에서 / 경로로 오는거 엘라스틱서치 포트로 포워딩 할 것.

```
https://k9e202a.p.ssafy.io/test_v1.0.0/
```