RecipeNav 배포 매뉴얼

# 개요

본 문서에서는 특화 프로젝트간 D105 팀이 제작한 Recipe Nav 배포 방식에 대해 소개합니다.

프론트엔드에서의 React, 백엔드 Spring boot, Redis, Naver Oauth 사용방식에 대해 설명합니다.  
  
React 배포 방법

1. 존재하는 깃 리포지토리로 이동합니다.

Ex) <https://lab.ssafy.com/s07-bigdata-dist-sub2/S07P22D105>

1. 깃 허브 Root 디렉터리에서 FE로 이동하면 recipe-nav라는 폴더가 존재하는데 그곳이 react workspace의 루트입니다.
2. npm build를 실행, build 디렉터리에 생성되는 것을 확인
3. 이제 nginx 설정을 하도록 하겠습니다.site-enable로 이동하여 설정합니다.

|  |
| --- |
| server {  listen 80;  server\_name j7d105.p.ssafy.io.redirect;  return 301 https://j7d105.p.ssafy.io$request\_uri;  }  server {  listen 443 ssl http2;  server\_name j7d105.p.ssafy.io;  ssl\_certificate /etc/letsencrypt/live/j7d105.p.ssafy.io/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/j7d105.p.ssafy.io/privkey.pem;  location /api {  proxy\_pass http://localhost:8081;  proxy\_set\_header Host $http\_host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-Forwarded-Proto $scheme;  }  location /{  root /home/ubuntu/react/recipenav\_app;  index index.html;  try\_files $uri $uri/ /index.html;  }  }  ~  ~ |

1. 노란색으로 표시한 root로 지정한 위치에 npm build로 생성한 build 디렉터리에 생성된 모든 것들을 복사합니다.  
   root는 정적 리소스의 위치를 가리키며 index는 index.html이라고 지정, try\_files는 서버에서 정적리소스를 찾을 떄, uri 주어진 대로 먼저 찾고 실패하면 그 뒤에 / 붙이고, 그것도 실패하면 index.html을 반환합니다.
2. –끝—

## 도커를 통한 자동배포 필요한 전처리 작업

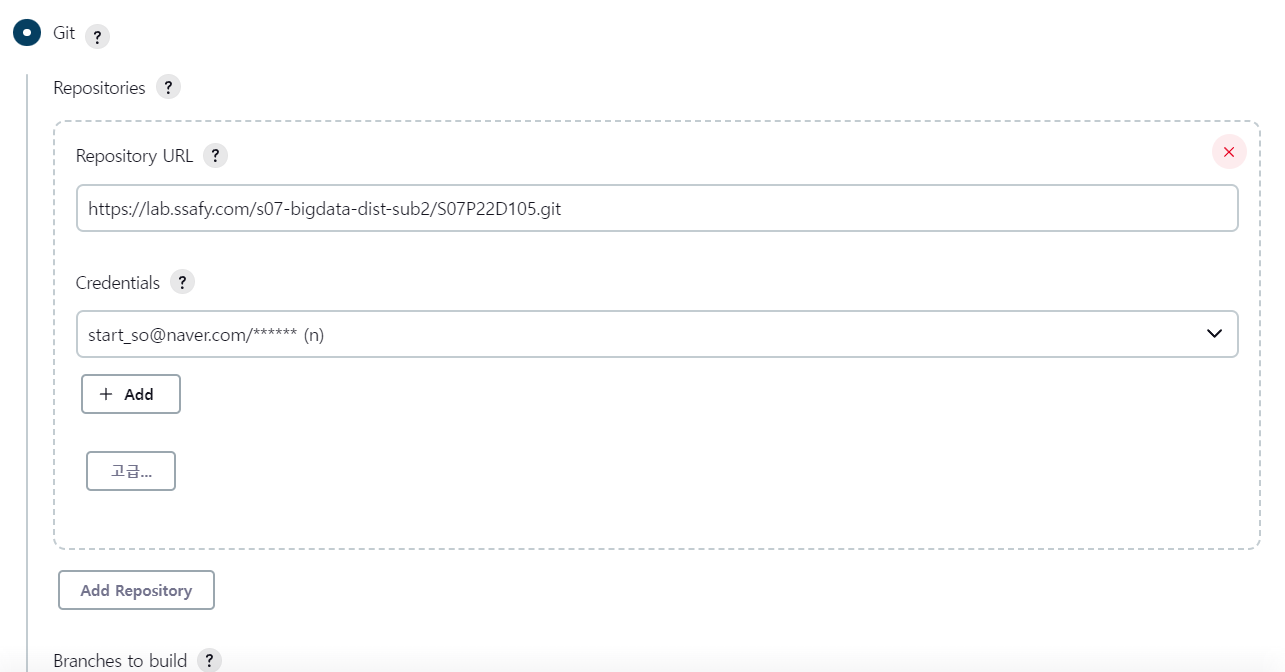
a. 도커 설치  
주의점 : 도커를 설치하면 , 도커 명령어를 사용할 수 있는 그룹,사용자는 root 밖에 없기 때문에 다른 유저를 사용 중이라면 group을 추가하거나 해야한다.

b. 젠킨스 설치, 젠킨스에 openSSH , node 를 위한 패키지 설치  
 인터넷에서 검색하자.

c. 젠킨스에서 host로 비밀번호 없이 ssh접속을 위한 설정  
인터넷에서 하자. Authorized\_keys 설정 시에 키가 여러 개라면 엔터를 입력 후 그밑에 작성, 넷 상에서는 enter 입력 시 특정 환경에서 오류가 나니 space를 입력해야 한다는데, 본인의 환경에서는 enter를 입력 해야했다.  
ex) <key종류>/spcae<공개키>/spcae <호스트>/enter

<key종류>/spcae<공개키>/spcae <호스트>  
위 방식이 통하는 버전은 다음과 같다.

Operating System: Ubuntu 20.04 LTS

Kernel: Linux 5.4.0-1018-aws  


1. 젠킨스를 통하여 gitlab의 특정 브랜치에 merge가 일어날 때, 빌드를 시키기 위하여 우선 Repository Url에 해당 깃 리포지토리의 uri를 적어 줍니다.

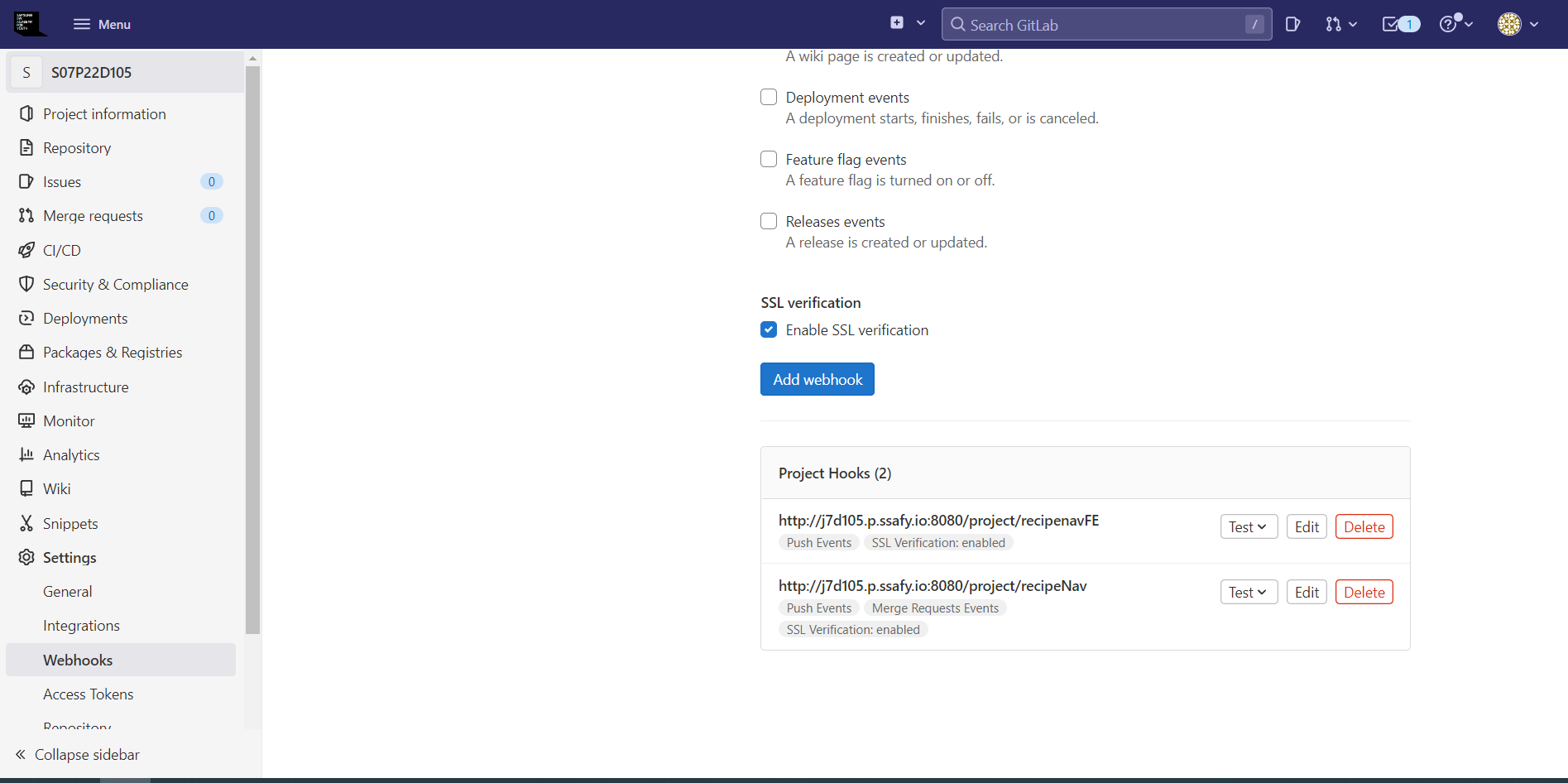
2. 인증이 필요하기 때문에 Credentials를 정하고 입력해주어야 합니다. 이번 프로젝트의 경우

3. AccessToken 지정에서 자주 발생하는 에러가 발생하여 계정을 통하여 진행하였습니다. Credentials 선택 후 단순히 ID와 PASSWORD를 적어주면 완료됩니다.

4.고급 탭을 클릭하면 Secret Token 영역이 있는데 이를 Generate 버튼을 눌러 생성하고 복사합니다. Ex ) ef107e1606fd74bc2b9a0c4d7a3e85df

5. 해당 리포지 토리로 이동하여 Web Hook을 설정합니다.

URL 예시 : <http://i7d101.p.ssafy.io:18080/github-webhook/>

시크릿 토큰 : 4번과정에서의 토큰 복사  
push 이벤트 클릭 : 브랜치 이름 작성  


6. 다음과 같이 빌드 스텝을 작성합니다.

|  |
| --- |
| cd /var/jenkins\_home/workspace/recipenavFE/FE/recipe-nav/  cat > .env <<EOF  REACT\_APP\_NAVER\_CLIENT\_ID=my\*\*\*\*\*CDFTNG6amzZ  REACT\_APP\_NAVER\_AUTH\_URL=https://nid.naver.com/oauth2.0/authorize  REACT\_APP\_NAVER\_REDIRECT\_URL=https://j7d105.p.ssafy.io/oauth/naver  EOF  ls  npm i  echo "빌드 시작 합니다.."  CI=false npm run build --mode deploy  echo "빌드 정상 종료" |

cat > .env <<EOF

…

EOF

의 경우 환경 변수 입력을 위하여 실행하였다.

Redis Cluster 구성 방법

[5분 안에 구축하는 Redis-Cluster (tistory.com)](https://co-de.tistory.com/24)  
위의 링크 통해 그대로 구성하면된다. 하지만 도커 사용 시 주의점이 있다.

레디스 클러스터로 master를 3개로하여 구성 시, Spring boot에서 조회를 할 떄,

x라는 임의의 키를 조회한다 하고 하자. 그러면 Spring boot에서는 클러스터 노드로 구성된 노드에게 get key x를 할 텐데, 해당 키가 x에 있다면 무사히 동작을 할 것이다. 하지만 x의 키가 다른 노드에 있을 경우 MOVED <node> 라는 메시지를 통해 이쪽 노드를 참고하라고 하며 다시 요청을 보내게 된다. 이 경우 클러스트 구축 시 사용된 localhost:7002 와 같은 IP 형식으로 나오게 되어 오류가 나게된다. 따라서 위 블로그를 따라할 시, host os의 ip를 사용하고 사용되는 모든 포트를 –p 옵션을 통하여 포워딩 해줘야할 것이다.

# Spring Boot 배포 방법

1.”React 배포 방법”의 5번까지 BE에 맞춰 설정하면 된다.  
2.젠킨스에서 아래와 같이 build 스크립트를 작성한다.

|  |
| --- |
| pwd  cd /var/jenkins\_home/workspace/recipeNav/BE/recipenav/  echo "gradle start"  chmod 777 ./gradlew  ./gradlew build  echo "update before sequnce"  ssh -t -t ubuntu@host.docker.internal<<EOF  cd /home/ubuntu/spring\_compose\_folder  docker-compose up --build -d  exit  EOF |

3. docker compose –up –build –d에 사용되는 docker-compose.yml 내용

|  |
| --- |
| version: "3"  services:  spring:  container\_name: spring  build:  context: /jenkins/workspace/recipeNav/BE/recipenav  dockerfile: Dockerfile  ports:  - 8081:8081  volumes:  - /home/ubuntu/spring\_compose\_folder/springv:/deploy  networks:  default:  external:  name: recipenav-net |

4. 사용되는 도커 이미지 파일, /jenkins/workspace/recipeNav/BE/recipenav/Dockerfile

|  |
| --- |
| FROM openjdk:11-jdk  ENTRYPOINT java -jar /deploy/\*.jar  EXPOSE 8081 |

# 그외 하둡을 위한 소스파일

1. 농수산물 가격 csv 파일 UTF-8로 변환

|  |
| --- |
| $sourcePath = "C:\data\dayselling" //엑셀 파일들이 있는 디렉토리  $destinationPath = "C:\data\dayselling\format" //결과 쓸 디렉토리  if (!(Test-Path $destinationPath)) {  New-Item -ItemType Directory -Path $destinationPath  }  Get-ChildItem -Path $sourcePath -File | ForEach-Object {  Write-Host "Converting $\_"  $content = Get-Content $\_.FullName  Set-content (Join-Path -Path $destinationPath -ChildPath $\_) -Encoding utf8 -Value $content  } |

2. NSCSMCsvCalc.java(농수산물 가격정보), IngredientCode.java(식자재 정보) 등등의 mapreduce 파일을 만들고 Driver.java에 등록한다.(관련 파일 첨부)

3. Hadoop 실행 예시 코드

|  |
| --- |
| - Projects 폴더에서  ant  - 디렉토리 삭제(소스코드 상에 자동 삭제 설정했으면 안 해도됨)  hdfs dfs -rm -r [wordcount\_test\_out]  - 데이터 파일 넣기  hdfs dfs -put [data/wordcount-data.txt] [wordcont\_test]  -하둡 실행  hadoop jar ssafy.jar [wordcount] [wordcount\_test] [wordcount\_test\_out]  - 결과 보기(리듀서 갯수에 따라서 숫자 늘려감)  hdfs dfs -cat [wordcount\_test\_out]/part-r-00000 | more hdfs dfs -cat [wordcount\_test\_out]/part-r-00001 | more |

4. sqoop 기본 명령어

|  |
| --- |
| 다운 버전 [Index of /dist/sqoop/1.4.7 (apache.org)](http://archive.apache.org/dist/sqoop/1.4.7/)  참고한 블로그 [[Sqoop] Sqoop 설치 및 MySQL과 Import, Export하기 (tistory.com)](https://earthconquest.tistory.com/241)  - 확인 명령어  sqoop list-databases --connect jdbc:mysql://j7d105.p.ssafy.io:3306/recipenav?serverTimezone=Asia/Seoul --username root --password \*\*\*\*\*  - Import 명령어  sqoop import --connect jdbc:mysql://j7d105.p.ssafy.io:3306/recipenav?serverTimezone=Asia/Seoul --table t\_recipe --target-dir /user/j7d105/t\_recipe --username root --passwrod \*\*\*\*\* -m 1 |

4-1. 농수축산물 일자별 가격 데이터 가공해서 export

|  |
| --- |
| hadoop jar ssafy.jar nscsm\_csv\_calc day\_sell\_product day\_sell\_product\_out  sqoop export --connect jdbc:mysql://j7d105.p.ssafy.io:3306/recipenav?serverTimezone=Asia/Seoul \  --username root \  --password recipenav123 \  --table t\_ingredient\_price\_log \  --export-dir /user/j7d105/day\_sell\_product\_out/\* \  --columns ing\_date,ing\_pcode,ing\_pscode,unit,ing\_min\_cost,ing\_max\_cost,ing\_avg\_cost \  --input-fields-terminated-by "\t" |

4-2. 레시피와 레시피 재료를 계산하기 위해 join해서 import

|  |
| --- |
| sqoop import  --connect jdbc:mysql://j7d105.p.ssafy.io:3306/recipenav?serverTimezone=Asia/Seoul \  --username root \ --password recipenav123 \  --target-dir /user/j7d105/t\_recipe\_ing\_manifu \  --split-by ti.rec\_seq \  --query 'select ti.rec\_seq,rec\_code,ing\_name,ing\_amount,if(ing\_type = "주재료",1,0) from t\_recipe\_ingredient ti inner join t\_recipe tr on ti.rec\_seq = tr.rec\_seq WHERE $CONDITIONS' |

4-3. 레시피 유사도 계산 export

|  |
| --- |
| hadoop jar ssafy.jar recipe\_simil\_calc 1 t\_recipe\_ing\_manifu t\_recipe\_ing\_manifu\_out  sqoop export  --connect jdbc:mysql://j7d105.p.ssafy.io:3306/recipenav?serverTimezone=Asia/Seoul \  --username root \ --password recipenav123 \  --table t\_recipe\_simil \  --export-dir /user/j7d105/t\_recipe\_ing\_manifu\_out/\* \  --columns recipe\_seq\_first,recipe\_seq\_second,value \  --input-fields-terminated-by "\t" |

4-4. 사용자 찜하기 기록 import

|  |
| --- |
| sqoop import --connect jdbc:mysql://j7d105.p.ssafy.io:3306/recipenav?serverTimezone=Asia/Seoul \  --username root \  --password recipenav123 \  --target-dir /user/j7d105/t\_user\_recipe\_like \  --split-by tur.userRecipeSeq \  --query 'select userRecipeSeq,rec\_seq,user\_seq from t\_user\_recipe tur WHERE $CONDITIONS' |

4-5. 한번 처리된 거래내역 import

|  |
| --- |
| sqoop import --connect jdbc:mysql://j7d105.p.ssafy.io:3306/recipenav?serverTimezone=Asia/Seoul \  --username root \  --password recipenav123 \  --target-dir /user/j7d105/tip \  --split-by tip.ing\_seq\  --query 'SELECT ing\_seq, ing\_date, ing\_avg\_cost FROM recipenav.t\_ingredient\_price\_log tip WHERE $CONDITIONS' |