1. Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서

1) 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정값, 버전(IDE버전 포함) 기재

- IDE

  IntelliJ IDEA Ultimate 2022.2.2

- Node.js

  node-v14.17.0

- Vue.js

  v2.7.10

- JDK

  openjdk 11.0.16 2022-07-19
  OpenJDK Runtime Environment (build 11.0.16+8-post-Ubuntu-0ubuntu120.04)
  OpenJDK 64-Bit Server VM (build 11.0.16+8-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)

- Spring Boot

  v2.5.2

- Apache Tomcat (내장 아파치 톰캣 WAS)

  org.apache.tomcat.embed
  » tomcat-embed-core

  v9.0.48

- 웹서버 및 리버스 프록시 서버

  nginx version: nginx/1.22.0 (Ubuntu)
  docker pull nginx:stable-alpine
  port: 80(HTTP), 443(HTTPS)

- Docker

  Docker version 20.10.18, build b40c2f6

- MySQL

  v8.0.30-1.el8
  docker pull mysql:8.0
  port: 3306

- Jenkins CI/CD

  v2.361.1
  docker pull jenkins
  port: 8080

2) 빌드 시 사용되는 환경변수 등 주요 내용 상세 기재

▼ build.gradle 파일

```
buildscript{
  ext {
```

```
    springBootVer = '2.5.2'
    springDependencyMgmtVer = '1.0.11'
    springLoadedVer = '1.2.8'
  }
  repositories {
    mavenCentral()
  }
  dependencies {
    classpath "org.springframework.boot:spring-boot-gradle-plugin:${springBootVer}"
    classpath "io.spring.gradle:dependency-management-plugin:${springDependencyMgmtVer}.RELEASE"
    classpath "org.springframework:springloaded:${springLoadedVer}.RELEASE"
  }
}
plugins {
  id 'java'
  id 'idea'
  id 'org.springframework.boot' version "${springBootVer}"
  id 'io.spring.dependency-management' version "${springDependencyMgmtVer}.RELEASE"
  id "org.asciidoctor.jvm.convert" version "3.3.2"
}

group = 'com.idk'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

configurations {
  asciidoctorExt
  compileOnly {
    extendsFrom annotationProcessor
  }
  providedRuntime
}

repositories {
  mavenCentral()
  maven { url 'https://repo.spring.io/snapshot' }
  maven { url 'https://repo.spring.io/milestone' }
  maven { url "https://repo.spring.io/libs-release" }
  maven { url "https://repo.maven.apache.org/maven2" }
  maven { url "https://build.shibboleth.net/nexus/content/repositories/releases" }
}

//set build time and inject value to application.properties
def buildTime() {
  def date = new Date()
  def formattedDate = date.format('yyyyMMdd_HHmm')
  return formattedDate
}

project.ext.set("build.date", buildTime())

dependencies {
  /*************************
   * Default
   *************************/
  implementation("org.springframework.plugin:spring-plugin-core:2.0.0.RELEASE")
  implementation('commons-io:commons-io:2.6')
  implementation("org.apache.commons:commons-collections4:4.4")
  implementation("org.apache.commons:commons-lang3:3.9")

  implementation 'org.springframework.boot:spring-boot-starter-actuator'
  implementation 'org.springframework.boot:spring-boot-starter-validation'
  implementation 'org.springframework.boot:spring-boot-starter-web'
  implementation 'org.springframework.boot:spring-boot-starter-data-jdbc'
  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'

  developmentOnly 'org.springframework.boot:spring-boot-devtools'
  annotationProcessor 'org.springframework.boot:spring-boot-configuration-processor'

  implementation("io.springfox:springfox-data-rest:3.0.0")
  implementation("io.springfox:springfox-bean-validators:3.0.0")
  implementation("io.springfox:springfox-boot-starter:3.0.0")
  implementation 'javax.annotation:javax.annotation-api:1.3.2'
  annotationProcessor ("javax.annotation:javax.annotation-api:1.3.2")

  /*************************
   * QueryDsl
   *************************/
  implementation("com.querydsl:querydsl-core")
  implementation("com.querydsl:querydsl-jpa")
  annotationProcessor "com.querydsl:querydsl-apt:${dependencyManagement.importedProperties['querydsl.version']}:jpa" // querydsl JPAAnno
```

```
  annotationProcessor("jakarta.persistence:jakarta.persistence-api")
  annotationProcessor("jakarta.annotation:jakarta.annotation-api")

  /*************************
   * Security
   *************************/
  implementation 'org.springframework.boot:spring-boot-starter-security'

  /*************************
   * Lombok
   *************************/
  compileOnly 'org.projectlombok:lombok'
  annotationProcessor 'org.projectlombok:lombok'

  /*************************
   * Jwt
   *************************/
  implementation("com.auth0:java-jwt:3.10.3")
  implementation group: 'io.jsonwebtoken', name: 'jjwt-api', version: '0.11.2'
  runtimeOnly group: 'io.jsonwebtoken', name: 'jjwt-impl', version: '0.11.2'
  runtimeOnly group: 'io.jsonwebtoken', name: 'jjwt-jackson', version: '0.11.2'

  /*************************
   * Database
   *************************/
  runtimeOnly 'mysql:mysql-connector-java'
  runtimeOnly 'com.h2database:h2'

  /*************************
   * WebClient
   *************************/
  implementation group: 'org.springframework.boot', name: 'spring-boot-starter-webflux', version: '2.7.3'

  /*************************
   * Quarts Scheduling
   *************************/
  implementation 'org.springframework.boot:spring-boot-starter-quartz'

  /*************************
   * Test
   *************************/
  testImplementation 'org.springframework.boot:spring-boot-starter-test'
  testImplementation 'org.springframework.security:spring-security-test'

  testImplementation('org.mockito:mockito-inline:3.4.0')
  testImplementation('org.mockito:mockito-core:3.4.0')
  testImplementation('org.mockito:mockito-junit-jupiter:3.4.0')

  /*************************
   * RestDocs
   *************************/
  asciidoctorExt 'org.springframework.restdocs:spring-restdocs-asciidoctor:2.0.4.RELEASE'
  testImplementation 'org.springframework.restdocs:spring-restdocs-mockmvc'

  /*************************
   * Mail
   *************************/
  implementation 'org.springframework.boot:spring-boot-starter-mail'

  /*************************
   * JSON
   *************************/
  implementation group: 'org.json', name: 'json', version: '20220924'
}

/*************************
 * RestDocs
 *************************/

ext {
  snippetsDir = file('build/generated-snippets')
}

tasks.named('test') {
  outputs.dir snippetsDir
  useJUnitPlatform()
}

asciidoctor {
  configurations 'asciidoctorExt'
  inputs.dir snippetsDir
```

```
  dependsOn test
}

task copyDocument(type: Copy) {
  dependsOn asciidoctor
  from file("build/docs/asciidoc")
  into file("src/main/resources/static/docs")
}

build {
  dependsOn copyDocument
}

bootJar {
  dependsOn asciidoctor
  from("${asciidoctor.outputDir}") {
    into 'static/docs'
  }
}
```

▼ Spring Boot Application 빌드 Dockefile

```
FROM openjdk:11
COPY /build/libs/api-0.0.1-SNAPSHOT.jar app.jar
COPY /src/main/resources/application.yml application.yml
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "-Dspring.profiles.active=prod", "-Dspring.config.location=/application.yml,/h
```

- `-jar` : jar파일 실행
- `-Duser.timezone=Asia/Seoul` : 스프링부트 애플리케이션 타임존 설정
- `-Dspring.profiles.active` : prod 프로파일로 스프링부트 애플리케이션 실행
  - oauth, db, aws 등의 민감한 비밀스러운 configuration 포함시킴
- `Dspring.config.location` : application.yml 파일의 경로를 직접 지정

▼ Nginx sites-available/default 설정 파일

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        server_name j7a208.p.ssafy.io;

        return 301 https://$server_name$request_uri;
}

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

        root /usr/share/nginx/html;
        index index.html index.htm;

        server_name j7a208.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/j7a208.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j7a208.p.ssafy.io/privkey.pem;

        location / {
                try_files $uri $uri/ /index.html;
        }

        location /api {
                proxy_pass http://backend:8081;
                proxy_http_version 1.1;
                proxy_set_header Connection "";

                proxy_set_header Host $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;
                proxy_set_header X-Forwarded-Host $host;
                proxy_set_header X-Forwarded-Port $server_port;
        }

        location /docs {
              proxy_pass http://backend:8081;
        }
}
```

- 443(HTTPS), 80(HTTP) port
  - `/` 경로: 정적인 리소스
  - `/api` 경로: `http://localhost:8081` 스프링 부트 애플리케이션으로 리버스 프록시 됨
  - `/docs` 경로: REST API 문서
- 5443(HTTPS)
  - WebRTC 비디오 및 음성 송출을 위한 OpenVidu 서버 port

3) 배포 시 특이사항 기재

A. Hadoop MapReduce를 사용한 빅데이터 분산처리 프로세스

1. 전처리

- Raw Datase (**업종 목적지별 배달 주문건수 데이터셋)**

Table 1

광역시도명	시군구명	날짜	시간대별 시간	한식_배달건수	분식_배달건수	카페/디저트_배달건수	돈까스/일식_배달건수	회_배달건수	치킨_배달건수	피자_배달건수	아시안/양식_배달건수	중식_배달건수	족발/보쌈_배달건수	야식_배달건수	찜탕_배달건수	도시락_배달건수	패스트푸드_배달건수
서울특별시	강남구	2019/08/27	18	0	0	0	0	0	0	0	0	0	5	0	0	0	0
서울특별시	강남구	2019/08/28	16	0	0	0	0	0	0	0	0	0	1	0	0	0	0
서울특별시	강남구	2019/08/29	15	0	0	0	0	0	0	0	0	0	2	0	0	0	0
서울특별시	강남구	2019/08/29	16	0	0	0	0	0	0	0	0	0	5	0	0	0	0
서울특별시	강남구	2019/08/30	10	0	0	0	0	0	0	0	0	0	5	0	0	0	0
서울특별시	강남구	2019/08/30	11	0	0	0	0	0	0	0	0	0	9	0	0	0	0
서울특별시	강남구	2019/08/30	12	0	0	0	0	0	0	0	0	0	2	0	0	0	0
서울특별시	강남구	2019/08/30	13	0	0	0	0	0	0	0	0	0	1	0	0	0	0
서울특별시	강남구	2019/08/31	14	0	0	0	0	0	0	0	0	0	0	0	0	0	1
서울특별시	강남구	2019/09/02	8	0	0	0	0	0	0	0	0	0	2	0	0	0	0
서울특별시	강남구	2019/09/02	9	0	0	0	0	0	0	0	0	0	10	0	0	0	0
서울특별시	강남구	2019/09/02	10	0	0	0	0	0	0	0	0	0	7	0	0	0	0
서울특별시	강남구	2019/09/02	11	0	0	0	0	0	0	0	0	0	7	0	0	0	0
서울특별시	강남구	2019/09/02	13	0	0	0	0	0	0	0	0	0	4	0	0	0	0
서울특별시	강남구	2019/09/02	15	0	0	0	0	0	0	0	0	0	6	0	0	0	0

- 전처리 코드

  preprocessing.py

```
# 1.xlsx을 csv로 만들기
import pandas

xlsx = pandas.read_excel('./input.xlsx', sheet_name=1)
print(xlsx.head())
xlsx.to_csv('input.tsv', sep='\t', index=False, header=False)
```

2. 전처리 결과물

input

강원도	강릉시	2019-08-02	19	없음	83	0.0	25.4	0.3	약	east	204	SSW	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-03	15	없음	83	0.0	25.7	0.6	약	east	251	WSW	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-03	16	없음	83	0.0	25.4	0.2	약	east	140	SE	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-04	11	없음	86	0.0	27.3	1.2	약	east	139	SE	0	0	2	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-04	14	없음	89	0.0	26.2	1.1	약	east	238	WSW	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-04	15	없음	88	0.0	25.9	1.1	약	east	229	SW	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-04	16	없음	85	0.0	25.8	0.7	약	east	228	SW	0	0	2	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-04	18	없음	88	0.0	25.3	1.0	약	east	218	SW	0	0	3	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-05	11	없음	87	0.0	27.4	0.9	약	east	169	S	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-05	12	없음	86	0.0	26.7	1.9	약	east	247	WSW	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-05	14	없음	85	0.0	25.6	0.9	약	east	235	SW	0	0	3	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-05	15	없음	84	0.0	25.1	1.4	약	east	226	SW	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-05	16	없음	85	0.0	24.7	1.2	약	east	242	WSW	0	0	1	0	0	0	0	0	0	0	0	0
강원도	강릉시	2019-08-06	12	없음	86	0.0	25.9	2.4	약	east	328	NNW	0	0	2	0	0	0	0	0	0	0	0	0

3. 분산처리

- 사용 명령어

```
0. start-dfs.sh
1. ant
2. hdfs dfs -mkdir idontknow_wordcount
3. hdfs dfs -put data/input.txt idontknow_wordcount
4. hdfs dfs -rm -r idontknow_out
5. hadoop jar ssafy.jar wordcount idontknow_wordcount idontknow_out
6. hadoop dfs -cat idowntknow_out/part-r-0000 | more
7. hadoop fs -get idontknow_out ~/result
```

- build.xml

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<project name="Hadoop" default="package">
<!--  Load all the default properties, and any the user wants     -->
<!--  to contribute (without having to type -D or edit this file  -->
<property file="${user.home}/build.properties"/>
<property file="${basedir}/build.properties"/>
<property name="build.encoding" value="UTF-8"/>
<property name="lib.dir" value="/home/hadoop/hadoop/share/hadoop/"/>
<property name="works.dir" value="${basedir}/src"/>
<property name="build.dir" value="${basedir}/build"/>
<property name="build.classes" value="${build.dir}/classes"/>
<property name="build.sysclasspath" value="last"/>
<property name="build.works" value="${build.dir}/works"/>
<property name="javac.debug" value="on"/>
<property name="javac.optimize" value="on"/>
<property name="javac.deprecation" value="off"/>
<property name="javac.version" value="1.8"/>
<property name="javac.args" value=""/>
<property name="javac.args.warnings" value=""/>
<property name="javac.args.warnings" value="-Xlint:checked"/>
<!--  the normal classpath  -->
<path id="classpath">
<fileset dir="${lib.dir}">
<include name="**/*.jar"/>
</fileset>
</path>
<!--  ====================================================  -->
<!--  Stuff needed by all targets                           -->
<!--  ====================================================  -->
<target name="init">
<mkdir dir="${build.dir}"/>
<mkdir dir="${build.classes}"/>
<mkdir dir="${build.works}"/>
</target>
<target name="compile-works" depends="init">
<javac encoding="${build.encoding}" srcdir="${works.dir}" includes="**/*.java" destdir="${build.works}" debug="${javac.debug}" optimize
<compilerarg line="${javac.args} ${javac.args.warnings}"/>
<classpath refid="classpath"/>
```

```
</javac>
</target>
<!--  ================================================================  -->
<!--  Make the Hadoop work jar.                                         -->
<!--  ================================================================  -->
<!--                                                                    -->
<!--  ================================================================  -->
<target name="ssafy-works" depends="compile-works">
<jar jarfile="${build.dir}/ssafy.jar" basedir="${build.works}">
<manifest>
<attribute name="Main-Class" value="ssafy/Driver"/>
</manifest>
</jar>
</target>
<!--  ================================================================  -->
<!--  D I S T R I B U T I O N                                           -->
<!--  ================================================================  -->
<!--                                                                    -->
<!--  ================================================================  -->
<target name="package" depends="ssafy-works">
<copy file="${build.dir}/ssafy.jar" todir="${basedir}"/>
</target>
</project>
```

- Driver.java

```
package ssafy;

import org.apache.hadoop.util.ProgramDriver;

public class Driver {
  public static void main(String[] args) {
    int exitCode = -1;
    ProgramDriver pgd = new ProgramDriver();
    try {

      pgd.addClass("wordcount", Wordcount.class, "A map/reduce program that counts pairs in the input files.");
      pgd.driver(args);
      exitCode = 0;
    }
    catch(Throwable e) {
      e.printStackTrace();
    }

    System.exit(exitCode);
  }
}
```

- Wordcount.java

```
package ssafy;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Wordcount {
  /*
  Object, Text : input key-value pair type (always same (to get a line of input file))
  Text, IntWritable : output key-value pair type
  */
  public static class TokenizerMapper
      extends Mapper<Object,Text,Text,IntWritable> {
```

```
  // 자치구 => ID
  private static Map<String, Integer> districtMap = new HashMap<>();
  static {
    districtMap.put("강남구", 1);
    districtMap.put("강동구", 2);
    districtMap.put("강북구", 3);
    districtMap.put("강서구", 4);
    districtMap.put("관악구", 5);
    districtMap.put("광진구", 6);
    districtMap.put("구로구", 7);
    districtMap.put("금천구", 8);
    districtMap.put("노원구", 9);
    districtMap.put("도봉구", 10);
    districtMap.put("동대문구", 11);
    districtMap.put("동작구", 12);
    districtMap.put("마포구", 13);
    districtMap.put("서대문구", 14);
    districtMap.put("서초구", 15);
    districtMap.put("성동구", 16);
    districtMap.put("성북구", 17);
    districtMap.put("송파구", 18);
    districtMap.put("양천구", 19);
    districtMap.put("영등포구", 20);
    districtMap.put("용산구", 21);
    districtMap.put("은평구", 22);
    districtMap.put("종로구", 23);
    districtMap.put("중구", 24);
    districtMap.put("중랑구", 25);
  }
  // variable declairations
  private IntWritable emitVal = new IntWritable();
  private Text emitKey = new Text();

  // map function (Context -> fixed parameter)
  public void map(Object key, Text value, Context context)
      throws IOException, InterruptedException {

    StringBuilder commonKeyText = new StringBuilder();
    // value.toString() : get a line
    String line = value.toString();
    String[] row = line.split("\t");
    String sido = row[0];
    // 서울특별시가 아니면 그대로 리턴
    if("서울특별시".equals(sido)) {
      int sigungu = districtMap.get(row[1]);
      String time = row[3];
      commonKeyText.append(sigungu).append(",").append(time).append(",");
      for (int i = 13; i < row.length ; i++) {
        StringBuilder keyText = new StringBuilder(commonKeyText);
        keyText.append(i - 12);
        emitVal.set(Integer.parseInt(row[i]));
        emitKey.set(keyText.toString());
        // emit a key-value pair
        context.write(emitKey, emitVal);
      }
    }
  }
}

/*
Text, IntWritable : input key type and the value type of input value list
Text, IntWritable : output key-value pair type
*/
public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {

  // variables
  private IntWritable result = new IntWritable();

  // key : a disticnt word
  // values :  Iterable type (data list)
  public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {

    int sum = 0;
    for ( IntWritable val : values ) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key,result);
  }
```

```
  }


  /* Main function */
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf,args).getRemainingArgs();
    if ( otherArgs.length != 2 ) {
      System.err.println("Usage: <in> <out>");
      System.exit(2);
    }
    Job job = new Job(conf,"word count");
    job.setJarByClass(Wordcount.class);

    // let hadoop know my map and reduce classes
    job.setMapperClass(TokenizerMapper.class);
    job.setReducerClass(IntSumReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    // set number of reduces
    job.setNumReduceTasks(2);

    // set input and output directories
    FileInputFormat.addInputPath(job,new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job,new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1 );
  }
}
```

4. 분산처리 결과물

- part-r-00000.txt

```
1,0,11  6
1,0,13  4
1,0,2 1
1,0,4 37
1,0,6 134
1,0,8 41
1,1,1 188
1,1,10  84
1,1,12  34
1,1,14  29
1,1,3 81
1,1,5 59
1,1,7 30
1,1,9 2
1,10,1  244
1,10,10 498
1,10,12 22
1,10,14 17
1,10,3  5
1,10,5  0
1,10,7  3
1,10,9  9
1,11,11 20
1,11,13 71
1,11,2  239
1,11,4  74
1,11,6  35
1,11,8  19
1,12,1  530
1,12,10 365
1,12,12 31
1,12,14 17
1,12,3  28
1,12,5  7
1,12,7  13
1,12,9  79
1,13,11 10
```

- part-r-00001.txt

```
1,0,1 213
1,0,10  106
1,0,12  35
1,0,14  32
1,0,3 94
1,0,5 102
1,0,7 27
1,0,9 6
1,1,11  3
1,1,13  3
1,1,2 1
1,1,4 35
1,1,6 96
1,1,8 48
1,10,11 14
1,10,13 13
1,10,2  72
1,10,4  22
1,10,6  2
1,10,8  14
1,11,1  439
1,11,10 468
1,11,12 43
1,11,14 29
1,11,3  14
1,11,5  22
1,11,7  12
1,11,9  74
1,12,11 13
```

5. 후처리

- postprocessing.py

```
# 2. part-r-00000, part-r-00001 파일을 하나의 csv 파일로 합치기 (정렬 포함)
import csv

res = []
with open('part-r-00000', 'r') as f1, open('part-r-00001', 'r') as f2, open('result.csv', 'w', encoding='utf-8', newline='') as f3:
    rows = f1.readlines()
    for row in rows:
        key, value = row.rstrip().split('\t')
        row = list(map(int, key.split(',')))
        row.append(int(value))
        res.append(row)
    rows = f2.readlines()
    for row in rows:
        key, value = row.rstrip().split('\t')
        row = list(map(int, key.split(',')))
        row.append(int(value))
        res.append(row)
    res.sort()
    csv_writer = csv.writer(f3)
    csv_writer.writerow(["자치구 코드", "시간대", "메뉴 코드", "총 배달 건수"])
    csv_writer.writerows(res)
```

6. 후처리 결과물

result

자치구 코드	시간대	메뉴 코드	총 배달 건수
1	0	1	213
1	0	2	1
1	0	3	94
1	0	4	37
1	0	5	102
1	0	6	134
1	0	7	27
1	0	8	41
1	0	9	6
1	0	10	106
1	0	11	6
1	0	12	35
1	0	13	4
1	0	14	32
1	1	1	188
1	1	2	1
1	1	3	81

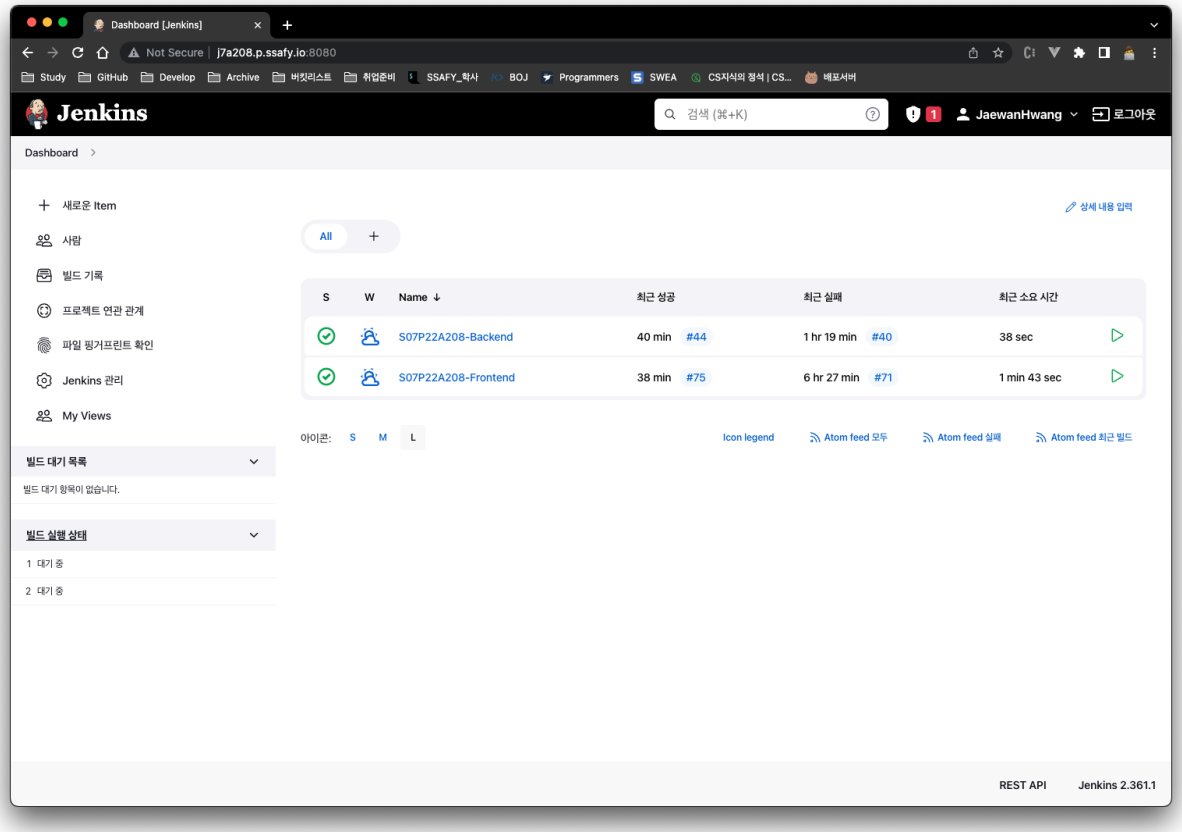
7. 분산처리 최종 결과물 DB에 저장

```
SET FOREIGN_KEY_CHECKS = 0;

USE  idontknow;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\result.csv'
INTO TABLE data
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@district_id, @`time`, @menu_id, @order_quantity)
SET `district_id` = @district_id,
    `time` = SEC_TO_TIME(@`time`*60*60),
    `menu_id` = @menu_id,
    `order_quantity` = @order_quantity;

SET FOREIGN_KEY_CHECKS = 1;
```

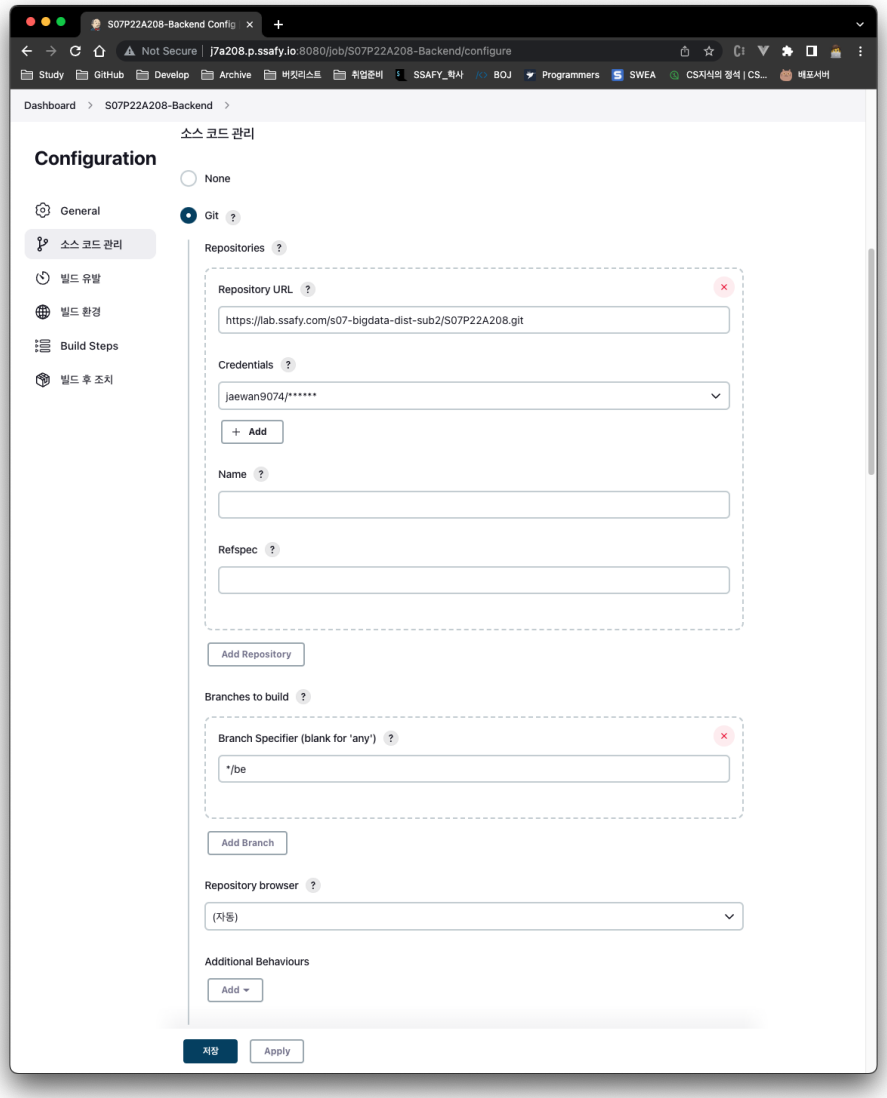
B. Docker + Jenkins CI/CD 설정 과정

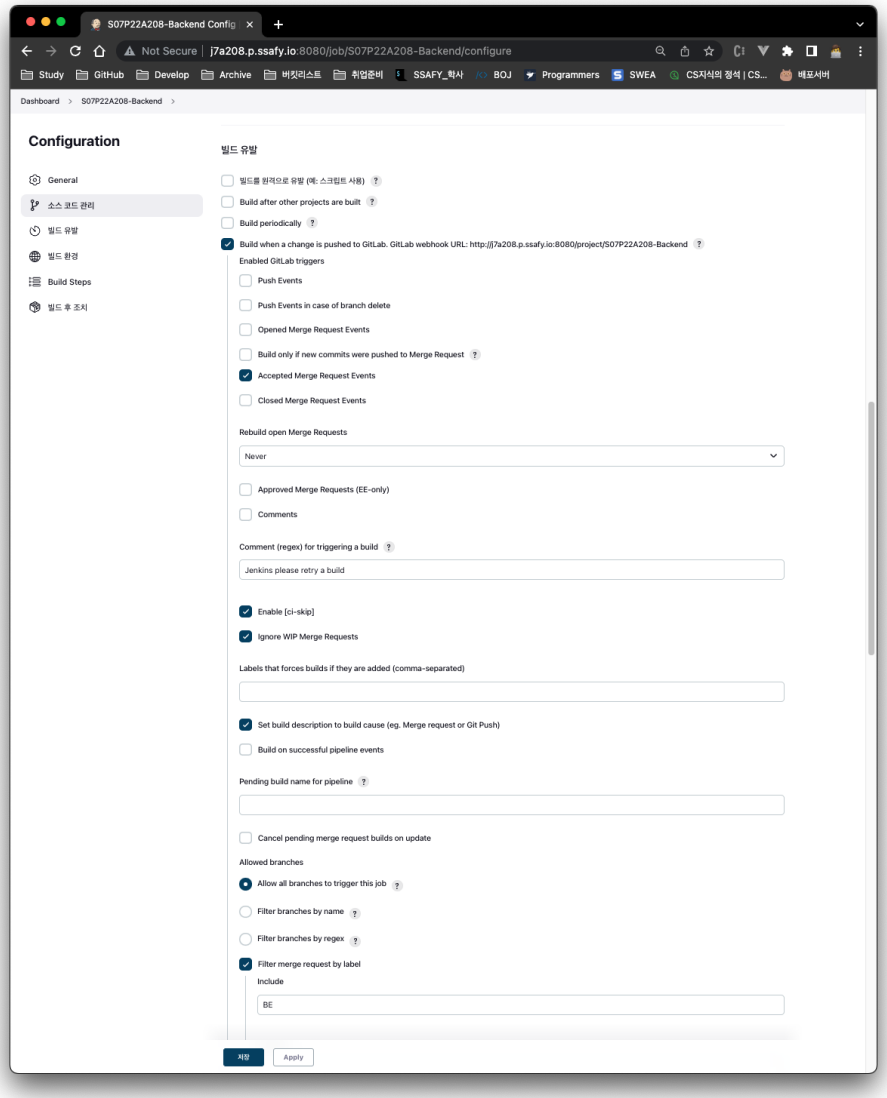
Jenkins 설정

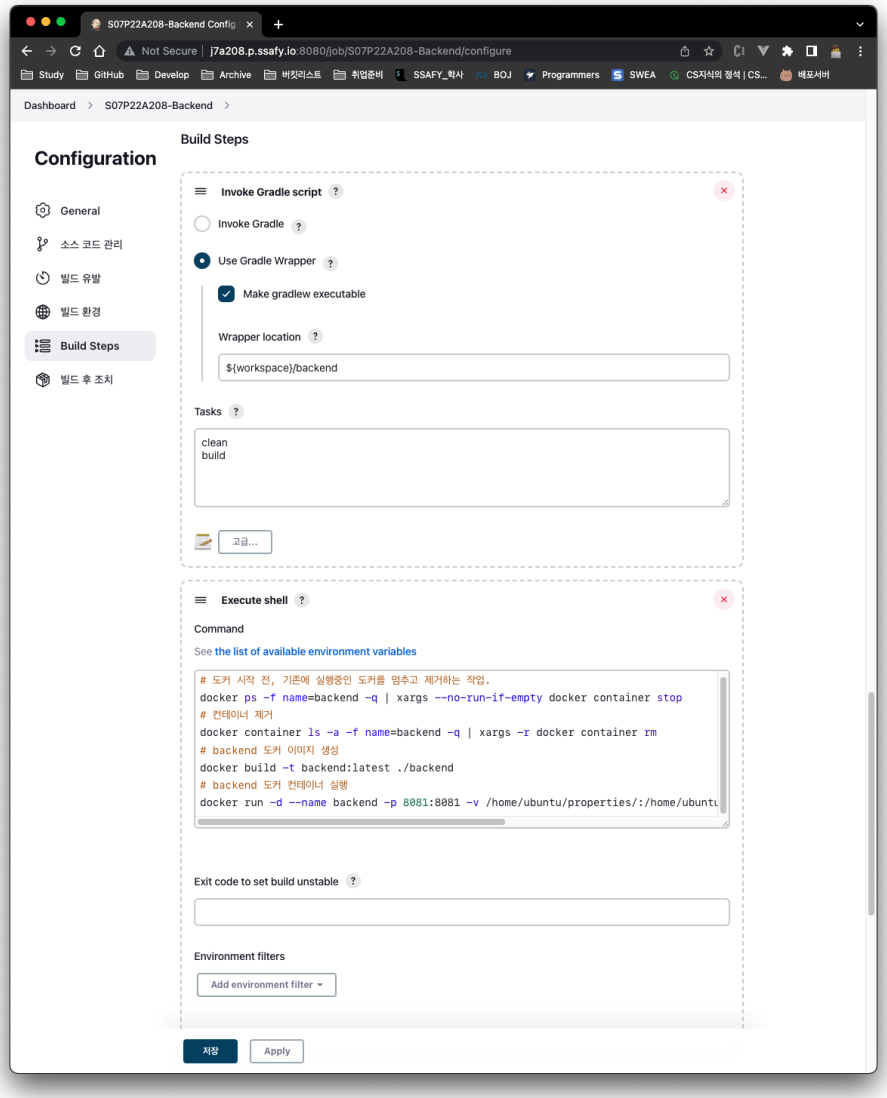
- Jenkins port: 8080
- 아래와 같이 Front-end, Back-end CI/CD 구축을 위한 아이템 2개 생성



- Jenkins Item 구성은 다음과 같이 설정


S07P22A208-Backend Config
Not Secure | j7a208.p.ssafy.io:8080/job/S07P22A208-Backend/configure
Study
GitHub
Develop
Archive
버킷리스트
취업준비
SSAFY_학사
BOJ
Programmers
SWEA
CS지식의 정석 | CS...
배포서버
Dashboard
S07P22A208-Backend
Configuration
General
소스 코드 관리
빌드 유발
빌드 환경
Build Steps
빌드 후 조치
소스 코드 관리
None
Git
Repositories
Repository URL
https://lab.ssafy.com/s07-bigdata-dist-sub2/S07P22A208.git
Credentials
jaewan9074/******
Add
Name
Refspec
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/be
Add Branch
Repository browser
(자동)
Additional Behaviours
Add
저장
Apply


S07P22A208-Backend Config
Not Secure | j7a208.p.ssafy.io:8080/job/S07P22A208-Backend/configure
Study GitHub Develop Archive 버킷리스트 취업준비 SSAFY_학사 BOJ Programmers SWEA CS지식의 정석 | CS... 배포서버
Dashboard > S07P22A208-Backend >
Configuration
General
소스 코드 관리
빌드 유발
빌드 환경
Build Steps
빌드 후 조치
빌드 유발
빌드를 원격으로 유발 (예: 스크립트 사용)
Build after other projects are built
Build periodically
Build when a change is pushed to GitLab. GitLab webhook URL: http://j7a208.p.ssafy.io:8080/project/S07P22A208-Backend
Enabled GitLab triggers
Push Events
Push Events in case of branch delete
Opened Merge Request Events
Build only if new commits were pushed to Merge Request
Accepted Merge Request Events
Closed Merge Request Events
Rebuild open Merge Requests
Never
Approved Merge Requests (EE-only)
Comments
Comment (regex) for triggering a build
Jenkins please retry a build
Enable [ci-skip]
Ignore WIP Merge Requests
Labels that forces builds if they are added (comma-separated)
Set build description to build cause (eg. Merge request or Git Push)
Build on successful pipeline events
Pending build name for pipeline
Cancel pending merge request builds on update
Allowed branches
Allow all branches to trigger this job
Filter branches by name
Filter branches by regex
Filter merge request by label
Include
BE
저장
Apply



- 다음의 명령어 수행: `sudo docker network create idontknownetwork` : docker 컨테이너간의 네트워크 생성
- Jenkins 컨테이너 안에서 Docker를 실행하면 Host의 도커와 연결됨
- Dockerfile을 /backend 와 /frontend에 각각 하나씩 넣고 nginx.conf는 /frontend에 넣음

Spring Boot_application.yml 수정

- `server.address=localhost`로 되어있어 스프링부트로 HTTP 요청을 보낼 수 없었음. 따라서 해당 설정 삭제함.

```
server:
  address: localhost <= 이거 없애야함
  servlet:
    encoding:
      force: 'true'
      charset: UTF-8
      enabled: 'true'
    contextPath: /
  port: '8081'
```

Docker_Frontend 명령어

- `-v /etc/letsencrypt/:/etc/letsencrypt/` 로 경로를 잡은이유는 pem키가 symlink로 저장되었기 때문에 letsencrpyt경로부터 volume mount 해야함!!!

```
# 도커 시작 전, 기존에 실행중인 도커를 멈추고 제거하는 작업.
docker ps -f name=frontend -q | xargs --no-run-if-empty docker container stop

# 컨테이너 제거
docker container ls -a -f name=frontend -q | xargs -r docker container rm

# frontend 도커 이미지 생성
docker build -t frontend:latest ./frontend

# frontend 도커 컨테이너 실행
docker run -d --name frontend -p 80:80 -p 443:443 -v /etc/letsencrypt/:/etc/letsencrypt/ -v /etc/localtime:/etc/localtime:ro --network idon
```

Docker_Frontend Dockerfile

```
FROM node:lts-alpine as build-stage
WORKDIR /frontend

COPY . .
RUN npm install
RUN npm run build

FROM nginx:stable-alpine as production-stage

RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx.conf /etc/nginx/conf.d/nginx.conf
COPY --from=build-stage ./frontend/dist /usr/share/nginx/html
EXPOSE 80 443
CMD ["nginx", "-g", "daemon off;"]
```

Docker_Backend 명령어

```
# 도커 시작 전, 기존에 실행중인 도커를 멈추고 제거하는 작업.
docker ps -f name=backend -q | xargs --no-run-if-empty docker container stop

# 컨테이너 제거
docker container ls -a -f name=backend -q | xargs -r docker container rm

# backend 도커 이미지 생성
docker build -t backend:latest ./backend

# backend 도커 컨테이너 실행
docker run -d --name backend -p 8081:8081 -v /home/ubuntu/properties/:/home/ubuntu/properties/ --network idontknownetwork backend:latest
```

Docker_Backend Dockerfile

- classpath 인식 안되어 application.yml을 스프링이 읽지 못했음 → application.yml 을 컨테이너에 직접 복사 후 사용함

```
FROM openjdk:11
COPY /build/libs/api-0.0.1-SNAPSHOT.jar app.jar
COPY /src/main/resources/application.yml application.yml
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "-Dspring.profiles.active=prod", "-Dspring.config.location=/application.yml,/home
```

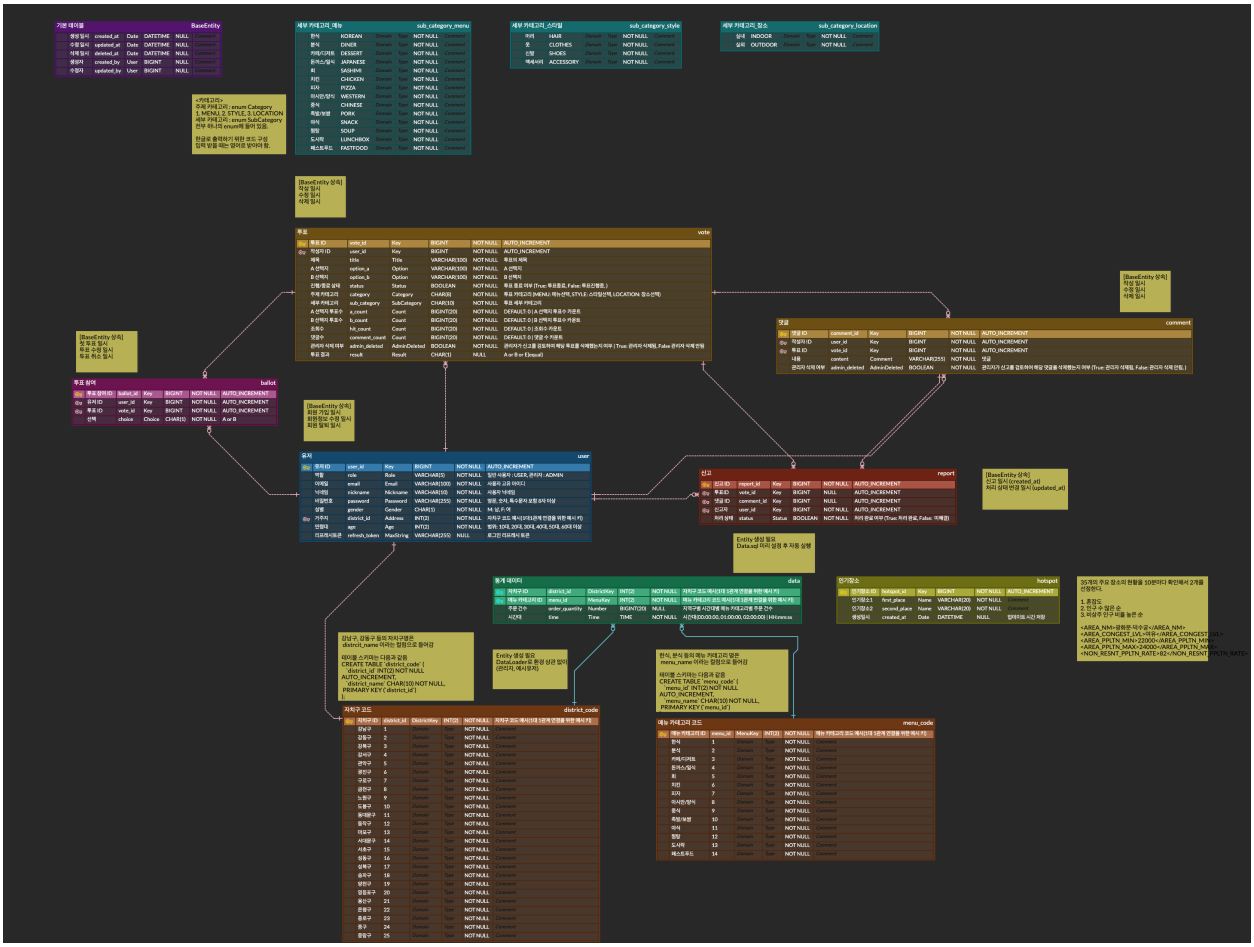
nginx.conf

- `http://backend:8081` 의 `backend` 는 idontknow network에 속한 backend container 이름

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;
```

```
        server_name j7a208.p.ssafy.io;

        return 301 https://$server_name$request_uri;
}

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

        root /usr/share/nginx/html;
        index index.html index.htm;

        server_name j7a208.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/j7a208.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j7a208.p.ssafy.io/privkey.pem;

        location / {
                try_files $uri $uri/ /index.html;
        }

        location /api {
                proxy_pass http://backend:8081;
                proxy_http_version 1.1;
                proxy_set_header Connection "";
                proxy_set_header Host $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;
                proxy_set_header X-Forwarded-Host $host;
                proxy_set_header X-Forwarded-Port $server_port;
        }

        location /docs {
              proxy_pass http://backend:8081;
        }
}
```

4) DB 접속 정보 등 프로젝트에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록



▼ application.yml

- local: 개발 환경
- prod: 배포 환경

```
spring:
  profiles:
    group:
      local:
        - local
        - secret
      prod :
        - secret
        - db
  jpa:
    hibernate:
      naming:
        implicit-strategy: org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
        physical-strategy: org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy
      ddl-auto: update
      dialect: org.hibernate.dialect.MySQL57Dialect
      format_sql: true
      default_batch_fetch_size: 100
    generate-ddl: true
  data:
    web:
      pageable:
        one-indexed-parameters: true
  devtools:
    livereload:
      enabled: 'true'
```

```
  # mail setting
  mail:
    host: smtp.gmail.com
    port: 587
    username: idontknowa208
    password: ${SMTP_PASSWORD}
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true
            required: true

# server
server:
  servlet:
    encoding:
      force: 'true'
      charset: UTF-8
      enabled: 'true'
    contextPath: /
  port: '8081'
build:
  date: '@build.date@'

# log
logging:
  level:
    org:
      springframework:
        security: DEBUG
        web: DEBUG
      apache:
        tiles: INFO
      hibernate:
        SQL: DEBUG
    root: INFO
    com:
      samsung:
        security: DEBUG
  file:
    name: ./ssafy-web.log

# jwt token
jwt:
  access-token-props:
    secret: adsjkQWFRaeiasjodfiwAWeeifjaSDOFJaiewAEWgIREAjORaerjAOESJOgDASIKFJIAJqiojuerfiAE
    expiration-time-milli-sec: '3600000'
  refresh-token-props:
    expiration-time-milli-sec: '864000000'
    secret: ZqefBscadfsGjaiADFSGaoGHdsjjsDFksjgDskdjgDjgHJmdSlzLfjbhikjaqWSogvjdsjzUEvlkvvhnDFGVnkfa==
---
# local only
spring.config.activate.on-profile: local
# database
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      password: idontknow
      username: idontknow
    url: jdbc:mysql://localhost:3306/idontknow?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=
convertToNull&rewriteBatchedStatements=true
```

▼ application-secret.yml

- 오픈 API, SMTP 서비스 사용을 위한 시크릿 키

```
# application-secret.yml
# 서울시 실시간 데이터 Open API Key
SEOUL_CITY_DATA:
  OPEN_API_KEY: 51464b79566a616531313550667a6775
  BASE_URL: http://openapi.seoul.go.kr:8088

# 기상청 단기예보 Open API Key
WEATHER_DATA:
```

```
  OPEN_API_KEY: 4diGvpxxl3AFrDekiPSCCXAlBkwmp2kQiJLE7Q0SLq5iIHb1fm0c5bbHhdkYP0hhM/Q9je0OAKRGvACpczieJQ==
  BASE_URL: http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0

# google mail info
SMTP_PASSWORD: snuwxhqaarfvrumv
FROM_ADDRESS: idontknowa208@gmail.com
```

▼ application-db.yml

- 배포 EC2 인스턴스의 MySQL 접속 정보

```
# application-db.yml
spring:
  datasource:
    url: jdbc:mysql://mysql:3306/idontknow?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=convert
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      password: develop@a208!@
      username: develop
```