

Reniors

포팅 매뉴얼

B307

김우빈, 석재호, 전병찬, 권예슬, 정아현

목차

1. 개요

2. 프로젝트 빌드

3. 프로젝트 배포

4. DB계정

5. 외부 서비스

포팅 매뉴얼

1. 개요

프로젝트 사용 도구

- 이슈관리 : Jira
- 형상관리 : GitLab
- 커뮤니케이션 : Notion, Mattermost
- UI/UX : Pigma

프로젝트 개발 환경

Frontend

- Visual Studio Code : 1.70.0
- Vue.js : 3.0
- Node.js : 16.16.0

Backend

- IntelliJ : 11.0.15+10-b2043.56 amd64
- Java : 1.8
- SpringBoot : 2.7.2

DB

- MySQL : 8.0.29

Server

- Ubuntu : 20.04

2. 프로젝트 빌드

프로젝트 빌드방법

Frontend

```
npm i
npm run build
```

Backend

```
Gradle -> build
```

프로젝트 환경변수

Frontend

```
// 카카오 API KEY
VUE_APP_KAKAO_REST_API_KEY =
// 카카오 URL
VUE_APP_KAKAO_REDIRECT_URI =
```

```
// YOUTUBE API KEY
VUE_APP_YOUTUBE_API_KEY =
// VITO API KEY
VUE_APP_VITO_CLIENT_ID =
// VITO CLIENT SECRET KEY
VUE_APP_VITO_CLIENT_SECRET
```

Backend

```
server:
  port: 포트번호
  error:
    whitelabel:
      enabled:

  servlet:
    context-path: 백엔드 기본주소

spring:
  //mysql
  datasource:
    mysql 드라이버
    url: mysql 주소

    username: 유저이름
    password: 비밀번호

  //메일smtp
  mail:
    host: 메일smtp 호스트
    port: 메일smtp 포트
    username: 유저이름
    password: 비밀번호
    properties:
      mail.smtp.auth:
      mail.smtp.ssl.enable:

  //파일 업로드
  servlet:
    multipart:
      file-size-threshold:
      max-file-size: 최대크기
      max-request-size: 요청크기

  //JWT
  token:
    expiration_time: 토큰 만료 시간
    secret:

  //S3
  cloud:
    aws:
      s3:
        bucket: 서비스명
      credentials:
        access-key: 접근키
        secret-key: 비밀키
      region:
        static: 지역
      auto:
      stack:
      auto:
```

GIT Ignore

Frontend

```
.env.local
```

Backend

```
application.yml
```

3. 프로젝트 배포

EC2 [원격 설정]

저장소 세팅 [ubuntu 20.04(LTS)]

```
sudo apt-get update
sudo apt-get install software-properties-common
sudo add-apt-repository universe
sudo apt-get update

[java version 8 설치]
sudo apt-get install openjdk-8-jdk

[nodejs 16.16.0 설치]
sudo curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash -
sudo apt-get install -y nodejs
node -v
```

certbot 설치

```
sudo apt-get update
sudo apt-get install certbot python3-certbot-nginx
```

SSL 설정 - certbot 이용 자동화 (유효기간 90일) - 멀티 도메인

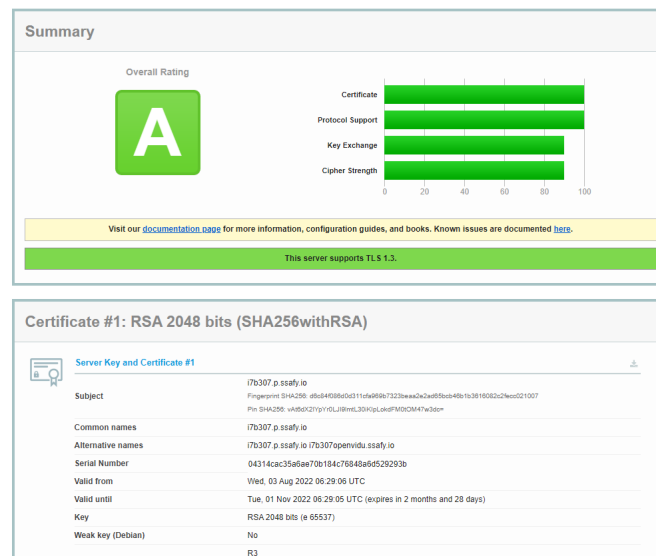
```
[SSL 설정]
sudo certbot --nginx -d i7b307.p.ssafy.io -d i7b307openvidu.ssafy.io

[갱신 테스트]
sudo certbot renew --dry-run

[인증서 만료일 확인]
certbot certificates
```

인증서 파일 위치

/etc/letsencrypt/live/i7b307.p.ssafy.io



<https://www.ssllabs.com/ssltest/> - SSL 적용 확인 및 평가

Crontab을 이용한 SSL 자동 갱신

```
[Crontab 보기]
sudo crontab -l

[Crontab 편집]
sudo crontab -e

[Crontab 실행 로그]
view /var/log/syslog
```

```
# min (0 - 59)
# hour (0 - 23)
# day of month (1 - 31)
# month (1 - 12)
# day of week (0 - 6) (0 to 6 are Sunday to
# Saturday, or use names; 7 is also Sunday)
#
# * * * * * command to execute
```

```
매월 1일 새벽 3시에 갱신되도록 설정
0 18 1 * * /usr/bin/certbot renew --renew-hook="sudo systemctl restart nginx"

18이 새벽 3시인 이유 : 서버시간에 맞춰서 진행 (date로 서버시간 확인)
```

도메인에 맞는 Nginx 설정 생성

```
vi /etc/nginx/sites-enabled/i7b307.conf
vi /etc/nginx/sites-enabled/openvidu.conf
```

i7b307.conf

```
#
#
#
server {
    if ($host = i7b307.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;

    server_name i7b307.p.ssafy.io;

    return 301 https://i7b307.p.ssafy.io$request_uri;
}

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

    server_name i7b307.p.ssafy.io;
    ssl_certificate /etc/letsencrypt/live/i7b307.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i7b307.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

    location / {
        root /home/ubuntu/dist;
        root /var/www/html;
        index index.nginx-debian.html;

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

    # 도커이미지와 했을 경우
    # proxy_pass http://localhost:8081;
    # proxy_redirect off;
    # charset utf-8;
    #
```

```
#         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;
#         proxy_set_header X-Forwarded-Proto $scheme;
#         proxy_set_header X-NginX-Proxy true;
    }

    location /api {
        proxy_pass http://localhost:8080;
        proxy_redirect off;
        charset utf-8;

        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;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy true;
    }

}
```

openvidu.conf

```
#
#
#
server {
    if ($host = i7b307openvidu.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;

    server_name i7b307openvidu.ssafy.io;

    return 301 https://i7b307openvidu.ssafy.io$request_uri;

}

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

    server_name     i7b307openvidu.ssafy.io;
    ssl_certificate /etc/letsencrypt/live/i7b307.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i7b307.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

    location / {
        #         proxy_pass http://localhost:4443;
        #         root /var/www/html;
        #         index index.nginx-debian.html;

        #         try_files $uri $uri/ /index.html;
    }

}
```

Nginx 설정 파일 심볼링 링크

```
ln -s /etc/nginx/sites-enabled/i7b307.conf /etc/nginx/sites-available/
ln -s /etc/nginx/sites-enabled/openvidu.conf /etc/nginx/sites-available/
```

Nginx 명령어

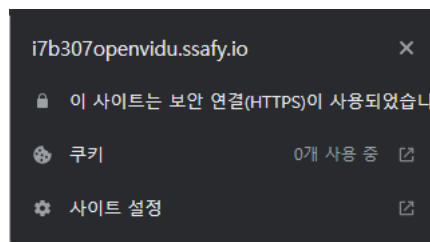
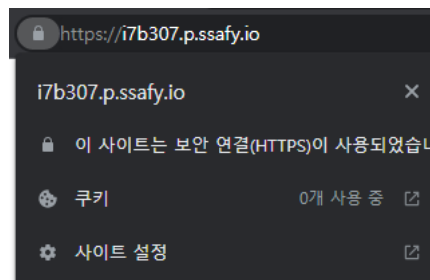
```
//nginx 서버 상태
sudo systemctl status nginx

//nginx 서버 켜기
sudo systemctl start nginx
```

```
//nginx 서버 중지
sudo systemctl stop nginx

//nginx 서버 재시작
sudo systemctl restart nginx
```

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-07-27 05:35:24 UTC; 2s ago
     Docs: man:nginx(8)
  Process: 90070 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 90079 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 90080 (nginx)
    Tasks: 5 (limit: 19204)
   Memory: 5.2M
   CGroup: /system.slice/nginx.service
           └─90080 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─90081 nginx: worker process
               └─90082 nginx: worker process
                 └─90083 nginx: worker process
                   └─90084 nginx: worker process
```



CI/CD 자동배포

Docker & Jenkins

Docker 설치

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

Docker 권한 설정

```
sudo usermod -aG docker $USER
sudo service docker restart

sudo su
sudo su ubuntu

docker ps
```

설치된 도커는 모두 root 권한으로 실행해야 하므로, 현재 유저에게 docker를 사용할 수 있는 권한을 부여하기 위해 해당 커맨드 실행

Docker 명령어


```
[컨테이너 조회]
sudo docker ps -> 실행중인 이미지 조회
sudo docker ps -a -> 중지된 컨테이너까지 모두 조회
```

Jenkins 설치 - Docker로 jenkins image 확보

```
docker pull jenkins/jenkins:lts

[이미지 다운로드 확인]
docker images
```

Jenkins 실행

```
[-d : 백그라운드 실행 ]
[-p : 컨테이너와 호스트 PC간 연결을 위한 포트 지정 ]
[-v : 이미지의 /var/jenkins_home 디렉토리를 호스트 PC내에 마운트 - Jenkins 설치 시 ssh 키값 생성, 저장소 참조 등을 용이하게 하기 위함 ]]
docker run --name jenkins -d -p 9090:8080 -p 50000:50000 -v /home/ubuntu/jenkinsDir:/var/jenkins_home -v /var/run/docker.sock:/var/run
```

port 사용 확인을 위한 tool 설치

```
sudo apt-get install net-tools
netstat -ano [전체 포트 사용 조회]
```

Jenkins 접속(<http://i7b307.p.ssafy.io:9090>)

```
[시크릿키 조회]
docker logs jenkins-docker

[ Admin User ]
ID : B307
PASSWORD : Common7333!
```

Docker Build

```
[jmark 프로젝트안에서 (Dockerfile 위치)]
docker build -t reniors .

[빌드된 도커 이미지 실행]
docker run -d -p 8080:8080 --name reniors_web reniors

[실행중인 도커 중지]
docker stop reniors_web

[도커 Container 삭제]
docker rm reniors_web

[도커 이미지 삭제]
docker rmi reniors
```

Dockerfile - Springboot

```
FROM openjdk:8 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
```

```

COPY settings.gradle .
COPY src src
RUN chmod =x ./gradlew
RUN ./gradlew bootJar

FROM openjdk:8
COPY --from=builder build/libs/reniors-0.0.1-SNAPSHOT.jar reniors.jar

EXPOSE 8080
CMD ["java","-jar","/reniors.jar"]

```

Dockerfile - Vue

```

# build
FROM node:16.16.0 as builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:stable-alpine
RUN rm -rf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/nginx/default.conf /etc/nginx/conf.d/default.conf

RUN rm -rf /usr/share/nginx/html/*
COPY --from=builder /app/dist /usr/share/nginx/html

EXPOSE 80
ENTRYPOINT ["nginx", "-g", "daemon off;"]

```

Jenkinsfile

```

pipeline{
    agent any
    environment {
        BACK_CONTAINER_NAME="reniors_back_container"
        BACK_NAME = "reniors_back"

        FRONT_CONTAINER_NAME="reniors_front_container"
        FRONT_NAME = "reniors_front"
    }
    stages {
        stage('Clean'){// 실행 중이 도커 컨테이너 중지 및 삭제
            steps{
                script {
                    try{
                        sh "docker stop ${BACK_CONTAINER_NAME}"
                        sh "docker stop ${FRONT_CONTAINER_NAME}"
                        sleep 1
                        sh "docker rm ${BACK_CONTAINER_NAME}"
                        sh "docker rm ${FRONT_CONTAINER_NAME}"
                    }catch(e){
                        sh 'exit 0'
                    }
                }
            }
        }
        stage('Build') { // 도커파일로 도커 이미지 생성
            steps {
                script{
                    sh "docker build -t ${BACK_NAME} ./Back_end/reniors/"
                    sh "docker build -t ${FRONT_NAME} ./Front_end/reniors/"
                }
            }
        }
        stage('Deploy'){// 빌드한 도커 이미지를 실행 및 기존 이미지 삭제
            steps {
                sh "docker run -d --name=${BACK_CONTAINER_NAME} -p 8080:8080 ${BACK_NAME}"
                sh "docker run -d --name=${FRONT_CONTAINER_NAME} -p 8081:80 ${FRONT_NAME}"

                sh "docker image prune"
            }
        }
    }
}

```

```
}  
}  
}
```

4. DB계정

MySQL APT Repository 추가

```
sudo wget https://dev.mysql.com/get/mysql-apt-config_0.8.13-1_all.deb
```

MySQL APT Repository 패키지 다운로드

```
sudo dpkg -i mysql-apt-config_0.8.13-1_all.deb  
//pubkey 등록 시  
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys <pubkey>
```

Repository 업데이트 및 mysql-server 설치

```
sudo apt-get update  
sudo apt-get install mysql-server
```

기본세팅

```
sudo ufw allow mysql -- 외부접속 기능 활성화  
sudo systemctl start mysql -- 서비스 시작  
sudo systemctl enable mysql -- 서버 재시작 시 mysql 재시작
```

접속 및 버전확인

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

```
show variables like "%version%";
```

```
mysql> show variables like "%version%";  
+-----+-----+  
| Variable_name | Value                                |  
+-----+-----+  
| admin_tls_version | TLSv1.2,TLSv1.3                    |  
| immediate_server_version | 999999                              |  
| innodb_version | 8.0.29                              |  
| original_server_version | 999999                              |  
| protocol_version | 10                                  |  
| replica_type_conversions |                                     |  
| slave_type_conversions |                                     |  
| tls_version | TLSv1.2,TLSv1.3                    |  
| version | 8.0.29-0ubuntu0.20.04.3            |  
| version_comment | (Ubuntu)                            |  
| version_compile_machine | x86_64                              |  
| version_compile_os | Linux                               |  
| version_compile_zlib | 1.2.11                              |  
+-----+-----+  
13 rows in set (0.01 sec)
```

root 계정 비밀번호 변경

```
alter user 'root'@'localhost' identified with mysql_native_password by 'new password';  
flush privileges;
```

Mysql 외부 원격 접속 설정

/etc/mysql/mysql.conf.d/mysqld.cnf 파일 수정

bind-address 127.0.0.1 적힌 줄 앞에 #를 붙여 주석처리

User 생성 및 권한 부여

```
create user '[username]'@'%' identified by '[password]';
grant all privileges on *.* to '[username]'@'%' with grant option;
flush privileges;
```

5. 외부 서비스

카카오

- <https://developers.kakao.com/>
- 서비스의 편리한 이용을 위해 자체 서비스 로그인/회원가입과 병행하여 카카오 로그인/회원가입 기능을 추가하였습니다. 또한 부가적인 서비스 제공을 위해 카카오 서버를 통해 사용자의 정보를 가져온 뒤 필요한 정보를 추가로 입력받아 회원가입 하는 방식을 사용하였습니다.

어플리케이션 추가

앱 아이콘

파일 선택

Reniors

JPG, GIF, PNG

권장 사이즈 128px, 최대 250KB

앱 이름

Reniors

사업자명

전병찬

- 입력된 정보는 사용자가 카카오 로그인을 할 때 표시됩니다.
- 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

도메인 등록

Web

삭제

수정

사이트 도메인

<https://i7b307.p.ssafy.io>

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

Redirect URI 설정

Redirect URI

[삭제](#)[수정](#)

Redirect URI https://i7b307.p.ssafy.io/kakaologin

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

로그인 활성화

카카오 로그인 **ON**

[동의 화면 미리보기](#)

활성화 설정

상태 **ON**

카카오 로그인 API를 활용하면 사용자들이 번거로운 회원 가입 절차 대신, 카카오톡으로 서비스를 시작할 수 있습니다.
상태가 OFF일 때도 카카오 로그인 설정 항목을 변경하고 서버에 저장할 수 있습니다.
상태가 ON일 때만 실제 서비스에서 카카오 로그인 화면이 연결됩니다.

카카오 인가코드 수신

앱 키

플랫폼	앱 키	재발급
네이티브 앱 키	복사	재발급
REST API 키	복사	재발급
JavaScript 키	복사	재발급
Admin 키	복사	재발급


- 네이티브 앱 키: Android, iOS SDK에서 API를 호출할 때 사용합니다.
- JavaScript 키: JavaScript SDK에서 API를 호출할 때 사용합니다.
- REST API 키: REST API를 호출할 때 사용합니다.
- Admin 키: 모든 권한을 갖고 있는 키입니다. 노출이 되지 않도록 주의가 필요합니다.

• Request URL

```
GET /oauth/authorize?client_id=${REST_API_KEY}&redirect_uri=${REDIRECT_URI}&response_type=code
```

Host: kauth.kakao.com


Kakao Access Token 수신

Request 

URL

```
POST /oauth/token HTTP/1.1
Host: kauth.kakao.com
Content-type: application/x-www-form-urlencoded;charset=utf-8
```

Access Token으로 사용자 정보 가져오기

Request: 액세스 토큰 사용 

URL

```
GET/POST /v2/user/me HTTP/1.1
Host: kapi.kakao.com
Authorization: Bearer ${ACCESS_TOKEN}
Content-type: application/x-www-form-urlencoded;charset=utf-8
```

VITO

- <https://developers.vito.ai/>
- STT서비스 사용을 위한 오픈API

어플리케이션 추가

애플리케이션 추가하기

등록 애플리케이션 수 : 0/5

애플리케이션 이름

Renior

유형

basic

추가하기

Amazon S3

이미지저장

Amazon S3 > 버킷 > reniors

reniors

Info

퍼블릭 액세스 가능

객체

속성

권한

지표

관리

액세스 지점

객체 (3)

객체는 Amazon S3에 저장되어 있는 기본 엔터티입니다. Amazon S3 [인벤토리](#)를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 영사

🔄

S3 URI 복사

URL 복사

다운로드

열기

삭제

작업 ▼


















폴더 만들기

업로드

🔍 접두사로 객체 찾기

<input type="checkbox"/>	이름	▲	유형	▼	마지막 수정	▼	크기
<input type="checkbox"/>	company/		폴더		-		
<input type="checkbox"/>	jobOpening/		폴더		-		
<input type="checkbox"/>	user/		폴더		-		

폴더별 구별

<input type="checkbox"/>	 12d2d478-e847-49ae-a607-870a60ea0a2e.jpg	jpg	2022. 8. 12. am 12:39:26 AM KST
<input type="checkbox"/>	 1b84e09c-b008-4e76-810f-217b986321b3.jpg	jpg	2022. 8. 11. am 11:24:25 AM KST
<input type="checkbox"/>	 215a9198-09ab-4a61-9f9c-ab5baabd3ad9.jpg	jpg	2022. 8. 11. am 10:41:46 AM KST
<input type="checkbox"/>	 267f44c3-049b-4e88-a20f-8e6e177f31db.png	png	2022. 8. 18. pm 4:46:12 PM KST
<input type="checkbox"/>	 36e47622-d91a-4b4e-a323-082d43c24907.jpg	jpg	2022. 8. 9. pm 1:04:08 PM KST
<input type="checkbox"/>	 42143e88-af2a-4171-b9f9-19d6ce695333.jpg	jpg	2022. 8. 10. am 1:13:26 AM KST
<input type="checkbox"/>	 44628b41-8cf0-4d23-b7b0-f3c39adc084d.jpg	jpg	2022. 8. 9. pm 3:38:41 PM KST
<input type="checkbox"/>	 4cc8db4a-46bc-4036-8a3c-953a737214c3.jpg	jpg	2022. 8. 15. pm 3:06:58 PM KST
<input type="checkbox"/>	 67df4b0d-6a17-4295-9653-e487f2c06992.jpg	jpg	2022. 8. 18. pm 1:28:08 PM KST
<input type="checkbox"/>	 75e5188f-c6b1-46a0-b087-9519ebb315f1.png	png	2022. 8. 14. pm 5:06:14 PM KST
<input type="checkbox"/>	 7ef1d808-5513-42c3-9333-779fbe903e0c.png	png	2022. 8. 19. am 1:55:49 AM KST
<input type="checkbox"/>	 7f490277-9e9e-40a0-929c-7f279279fe92.jpg	jpg	2022. 8. 12. am 1:12:21 AM KST
<input type="checkbox"/>	 825968c4-3fa8-4b86-b062-892ae771027b.png	png	2022. 8. 18. pm 6:32:32 PM KST
<input type="checkbox"/>	 83c6f988-e25d-4b97-b124-f39078069dce.png	png	2022. 8. 14. pm 5:11:22 PM KST
<input type="checkbox"/>	 861a69f1-cbad-4e08-86a3-5bdc2495b68d.png	png	2022. 8. 14. am 1:22:02 AM KST
<input type="checkbox"/>	 8a86776e-efd3-40bb-ac30-1112c38ebfb4.png	png	2022. 8. 18. pm 1:27:28 PM KST
<input type="checkbox"/>	 996b42fe-2806-4e1d-bdaf-3bb9b41f64f9.jpg	jpg	2022. 8. 15. pm 3:03:54 PM KST
<input type="checkbox"/>	 99be19e4-9937-4193-98ee-4935af28a104.jpg	jpg	2022. 8. 16. pm 9:18:52 PM KST
<input type="checkbox"/>	 a1c27073-aa28-411c-a14d-5b0bb05ff9b1.jpg	jpg	2022. 8. 10. am 10:25:36 AM KST

build.gradle

```
//S3
implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'
```

AwsS3Config.class

포팅 매뉴얼

14


```

@Configuration
public class AwsS3Config {

    @Value("${cloud.aws.credentials.access-key}")
    private String accessKey;

    @Value("${cloud.aws.credentials.secret-key}")
    private String secretKey;

    @Value("${cloud.aws.region.static}")
    private String region;

    @Bean
    public AmazonS3Client amazonS3Client() {
        BasicAWSCredentials awsCreds = new BasicAWSCredentials(accessKey, secretKey);
        return (AmazonS3Client) AmazonS3ClientBuilder.standard()
            .withRegion(region)
            .withCredentials(new AWSSStaticCredentialsProvider(awsCreds))
            .build();
    }
}

```

AwsS3Service.class

```

public String uploadFile(MultipartFile file, String folderName) {
    // forEach 구문을 통해 multipartFile로 넘어온 파일들 하나씩 fileNameList에 추가
    String fileName = createFileName(file.getOriginalFilename());
    ObjectMetadata objectMetadata = new ObjectMetadata();
    objectMetadata.setContentLength(file.getSize());
    objectMetadata.setContentType(file.getContentType());

    try(InputStream inputStream = file.getInputStream()) {
        amazonS3.putObject(new PutObjectRequest(bucket, folderName+fileName, inputStream, objectMetadata)
            .withCannedAcl(CannedAccessControlList.PublicRead));
    } catch(IOException e) {
        throw new RuntimeException(HttpStatus.INTERNAL_SERVER_ERROR, "파일 업로드에 실패했습니다.");
    }
    return fileName;
}

```