



GAPPA FORTING MANUAL

- 삼성청년SW아카데미 대전캠퍼스 9기 B206
- 특화 프로젝트 (6주, 2023.08.28 ~ 2023.10.06)
- 김동익, 김동현, 김용범, 김정훈, 조해린, 최한윤

목차

목차

1. 프로젝트 기술 스택 및 버전

- 1-1. Front-End
- 1-2. Back-End
- 1-3. Infra
- 1-4. 기타

2. EC2 Setting

- 2.1 EC2 접속하기
- 2.2 우분투 방화벽(UFW) 설정
- 2.3 Docker 설치
- 2.4 Nginx 설정
 - Nginx 설치
 - SSL 설정 (feat. Certbot)
 - Nginx 설정 파일 (feat. 리버스 프록시)
 - Nginx 설정 적용
- 2.5 MySQL 설정
 - MySQL 이미지 pull
 - Docker 컨테이너 볼륨 설정 및 확인
 - MySQL 컨테이너 실행
 - MySQL 새 계정 생성 및 권한 설정
 - 새 계정으로 DB 생성
 - IP 접속 가능 범위 설정
- 2.6 Redis 설정
 - Redis 이미지 pull
 - redis container 실행
 - redis가 설치된 docker 컨테이너 내부로 접속
 - redis 접속
- 2.7 Jenkins 설치
 - docker-compose를 사용하여 jenkins container를 실행하기

3. Jenkins 설정

- 3.1 초기 계정 설정
- 3.2 Plugin 설치
 - 추가로 설치 해야 할 plugin 목록
- 3.3 Credential 설정
- 3.4 jenkins GitLab Connection 등록
- 3.5 Jenkins pipeline 생성
- 3.6 Gitlab webhook 설정

4. Back-End 빌드 및 배포

- 4-1 Back-end 환경변수
- 4-2 자바 및 gradle 설치
- 4-3 Dockerfile 작성
- 4-4 Jenkinsfile 작성
- 4-5 Jenkins 설정

5. Front-End 빌드 및 배포

- 5-1 Dockerfile 작성
- 5-2 Jenkinsfile 작성
- 5-3 Jenkins 설정

6. 외부 서비스

- 6.1 Naver 문자 서비스

[네이버 sens 서비스](#)

[API URL](#)

[NAVER Cloud Platform 인증키 및 Signature 생성](#)

[SENS 서비스 신청하고 서비스 ID 발급받기](#)

[메시지](#)

[메시지 발송](#)

[요청 URL](#)

[Path Variables](#)

[Headers](#)

[API Header](#)

[요청 Body](#)

[응답 Body](#)

[응답 Status](#)

[6-2. FireBase](#)

[FireBase 프로젝트 생성](#)

[웹 앱의 구성 스니펫 가져오기](#)

[비밀 KEY 파일 생성](#)

[Spring백엔드 설정](#)

1. 프로젝트 기술 스택 및 버전

1-1. Front-End

React 18.2.0

Node.js 14.21.3

PWA

1-2. Back-End

SpringBoot 2.7.15

Spring Data JPA

MySQL 8.1.0

JDK 11.0.18

Redis 7.2.1

1-3. Infra

AWS EC2

Jenkins 2.423

Nginx 1.18.0

Docker 24.0.6

docker-compose 2.21.0

1-4. 기타

gitlab

ERDCloud

Figma

Jira

Notion

Mattermost

Discord

2. EC2 Setting

2.1 EC2 접속하기


- MobaXterm활용
- SSH 선택
- Romete host : j9b206.p.ssafy.io
- User private Key : 발급받은 key입력

Session settings

×



Basic SSH settings

Remote host * ☐ Specify username  Port


Advanced SSH settings

Terminal settings Network settings Bookmark settings

☒ X11-Forwarding ☒ Compression Remote environment:

Execute command: ☐ Do not exit after command ends

SSH-browser type: ☐ Follow SSH path (experimental)

☒ Use private key 

Execute macro at session start:

OK

Cancel

2.2 우분투 방화벽(UFW) 설정

```
sudo ufw default deny incoming # 모든 인바운드 연결 차단
sudo ufw default allow outgoing # 모든 아웃바운드 연결 허용
sudo ufw allow ssh # 22번 포트 허용
sudo ufw allow http # 80번 포트 허용
sudo ufw allow https # 443 포트 허용
```

- 방화벽 실행 `sudo ufw enable`
- 허용된 포트 확인 `sudo ufw status`

2.3 Docker 설치

- 공식 문서를 먼저 확인

Install Docker Engine on Ubuntu

Jumpstart your client-side server applications with Docker Engine on Ubuntu. This guide details prerequisites and multiple methods to install.

 <https://docs.docker.com/engine/install/ubuntu/>

- ec2에 docker 설치

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME)" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- docker engine과 그에 따른 plugin설치

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo apt install docker-compose

# 정상 설치 되었는지 확인
sudo docker -v
sudo docker compose version
```

2.4 Nginx 설정

Nginx 설치

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx
```

SSL 설정 (feat. Certbot)

```
# snap을 이용하여 core 설치 -> snap을 최신 버전으로 유지하기 위해 설치
sudo snap install core
# core를 refresh 해준다.
sudo snap refresh core
# 기존에 잘못된 certbot이 설치되어있을 수도 있으니 삭제 해준다.
sudo apt remove certbot
# certbot 설치
sudo snap install --classic certbot
# certbot 명령을 로컬에서 실행할 수 있도록 snap의 certbot 파일을 로컬의 cerbot과 링크(연결) 시켜준다. -s 옵션은 심볼릭링크를 하겠다는 것.
sudo ln -s /snap/bin/certbot /usr/bin/certbot
# certbot 사용해 ssl 설정
sudo certbot --nginx
```

Nginx 설정 파일 (feat. 리버스 프록시)

```
sudo vim /etc/nginx/conf.d/default.conf
```

```
server {
    # 80번 포트에서 연결 수신
    listen 80;
```

```

server_name j9b206.p.ssafy.io;
return 301 https://$host$request_uri;
}

# HTTPS 로 연결 수신
server {
    listen 443 ssl;
    server_name j9b206.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/j9b206.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j9b206.p.ssafy.io/privkey.pem;

    # springboot에 대한 프록시 설정
    location /api {
        # rewrite ^/api(/.*)$ $1 break;
        proxy_pass http://j9b206.p.ssafy.io:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        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;
    }

    # 프론트에 대한 프록시 설정
    location / {
        proxy_pass http://j9b206.p.ssafy.io:3000;
        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;
    }
}

```

Nginx 설정 적용

```

sudo nginx -t
sudo service nginx restart

```

2.5 MySQL 설정

MySQL 이미지 pull

```

sudo docker mysql:latest

```

Docker 컨테이너 볼륨 설정 및 확인

```

# 볼륨 설정
sudo docker volume create mysql-volume
# 볼륨 확인
sudo docker volume ls

```

MySQL 컨테이너 실행

```

sudo docker run -d --name mysql-container -p 2231:3306 -v mysql-volume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=1234 mysql:latest

```

MySQL 새 계정 생성 및 권한 설정

```

mysql -u root -p # MySQL 서버 접속

```

```

# 새 계정 생성
mysql> CREATE USER gappa@'%' identified by '어려운비밀번호';
# 권한 설정

```

```
mysql> GRANT ALL PRIVILEGES ON *.* to gappa@'%';
mysql> FLUSH PRIVILEGES;
mysql> exit;
```

새 계정으로 DB 생성

```
# 생성한 계정으로 MySQL 서버 접속
bash# mysql -u gappa -p
# DB 생성 및 확인
mysql> CREATE DATABASE gtest;
mysql> SHOW DATABASES;
```

IP 접속 가능 범위 설정

```
mysql> SELECT user, host FROM mysql.user;
# root 사용자 접속 권한 삭제
mysql> DELETE FROM mysql.user WHERE User='root' AND Host='%';
mysql> FLUSH PRIVILEGES;
```

2.6 Redis 설정

Redis 이미지 pull

```
sudo docker pull redis
```

redis container 실행

```
sudo docker run -p 9707:6379 --name redis-container -d redis:latest --requirepass "gappa"
```

- `--name redis-container` : 컨테이너에 `redis-container` 라는 이름을 부여
- `-p 9707:6379` : 호스트의 9707 포트와 컨테이너의 6379 포트를 매핑
- `-d` : 컨테이너를 백그라운드에서 실행
- `redis` : 사용할 Docker 이미지의 이름
- `requirepass` : 비밀번호

redis가 설치된 docker 컨테이너 내부로 접속

```
sudo docker exec -it redis-container /bin/bash
```

redis 접속

ubuntu

```
redis-cli -p {port} -a {password}
```

cmd(외부)

```
redis-cli -h {hostname} -p {port} -a {password}
```

2.7 Jenkins 설치

docker-compose를 사용하여 jenkins container를 실행하기

1. jenkins container를 실행시킬 docker-compose 파일을 만들

```
version: '3'

services:
  jenkins:
    image: jenkins/jenkins:2.422
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - "9090:8080"
    user: root
```

- jenkins 기본 포트는 8080인데 9090포트 사용하도록 지정해줌

2. docker-compose로 jenkins container를 실행하기

```
sudo docker-compose up -d
```

3. 정상적으로 container가 실행되고 있는지 확인

```
sudo docker ps -a
```

```
bcaa359676a4 jenkins/jenkins:lts "/usr/bin/tini -- /u..." 3 weeks ago Up 6 days 50000/tcp
cp, 0.0.0.0:9090->8080/tcp, :::9090->8080/tcp jenkins
ubuntu@ip-172-26-12-252:~$
```

4. Jenkins container에 접속하여 Docker를 설치

a. jenkins 컨테이너 내부에 접속

```
ubuntu@ip-172-26-12-252:~$ sudo docker exec -it jenkins /bin/bash
root@bcaa359676a4:/#
```

b. docker 설치

```
sudo docker exec -it jenkins bin/bash

## root~~ 나오면
## jenkins 컨테이너 내부에 접속 완료

##docker 설치
apt-get update
apt-get install ca-certificates curl gnupg lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
    $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-compose
```

5. Jenkins 접속

http://j9b206.p.ssafy.io:9090/

3. Jenkins 설정

3.1 초기 계정 설정

1. 초기 접속화면 Unlock Jenkins
 - `cat /var/lib/jenkins/secrets/initialAdminPassword`
 - 여기서 비밀번호를 확인하고 입력
2. install suggested plugins 선택
3. Create First Admin User
 - a. 계정명 : gappa_admin
 - b. 암호 : e94-nom95-a97-don98-gappa98-kkkkcc
4. Jenkins Url
 - a. default사용해도 됨,

3.2 Plugin 설치

Jenkins 관리 → Plugin in

추가로 설치 해야 할 plugin 목록

- GitLab
- Generic Webhook Trigger
- GitLab API
- GitLab Authentication
- Loading plugin extention
- NodeJs (자동 배포로 프론트엔트 빌드 시 필요)

3.3 Credential 설정

Jenkins 관리 → Security → Credentials → Stores scoped to Jenkins → (global)

→ + Add Credential 선택

Scope ?

Global (Jenkins, nodes, items, all child items, etc) ▼

API token

🔒 Concealed Change Password

ID ?

f563bb03-9b5a-4caa-afc8-3808231ba01e

Description ?

Save

- gitlab에서 먼저 access Token을 발급 받고 토큰을 넣어준다.
- 토큰 발급 받기
gitlab 로그인 → 사람 누름 → Edit profile → access token
이름, 만료날짜, 권한 범위 등을 설정

User Settings

- Profile
- Account
- Applications
- Chat
- Access Tokens**
- Emails
- Password
- Notifications
- SSH Keys
- GPG Keys
- Preferences
- Comment Templates
- Active Sessions
- Authentication Log

User Settings > Access Tokens

Q Search page

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

Token name

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

Expiration date

2023-11-03

Select scopes

Scopes set the permission levels granted to the token. [Learn more.](#)

- ☐ **api**
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ **read_api**
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ **read_user**
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- ☐ **read_repository**
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- ☐ **write_repository**
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

3.4 jenkins GitLab Connection 등록

Jenkins관리 → System → GitLab 이동

- 원하는 connection 이름 설정
- Gitlab 주소 입력

- 앞서 만든 Credential 연결

GitLab

☒ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?

A name for the connection

deploy-gappa

GitLab host URL ?

The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)

https://lab.ssafy.com/

Credentials ?

API Token for accessing GitLab

- none -

+ Add +

API Token for GitLab access required

고급 ^

고급 ^

API-Level ?

API Level for accessing GitLab

autodetect

☐ Ignore SSL Certificate Errors ?

Connection timeout (in seconds) ?

The time to wait for establishing the connection

10

Read timeout (in seconds) ?

The time to wait while receiving the response

10

Test Connection

추가

3.5 Jenkins pipeline 생성

- + 새로운 Item → 이름 입력, Pipeline 선택 → ok
- 프론트와 백엔드 2개를 만들어야 함

S	W	Name	최근 성공	최근 실패	최근 소요 시간	
✓	☀	gappa-back	24 min #249	6 hr 32 min #222	11 min	▶
✓	☀	gappa-front	24 min #239	2 days 3 hr #165	15 min	▶

- 구성 → build Trigger 이동
- build를 유발할 Tirgger 옵션을 선택하여 적용
- 고급을 눌러 webhook 설정을 위한 Secret Toke을 발급

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://9b101.p.ssafy.io:9090/waterbell/project/WaterBell-Pipeline> ?

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

고급 ^

☒ Enable [ci-skip] ?

☒ Ignore WIP Merge Requests ?

Labels that launch a build 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

Secret token ?

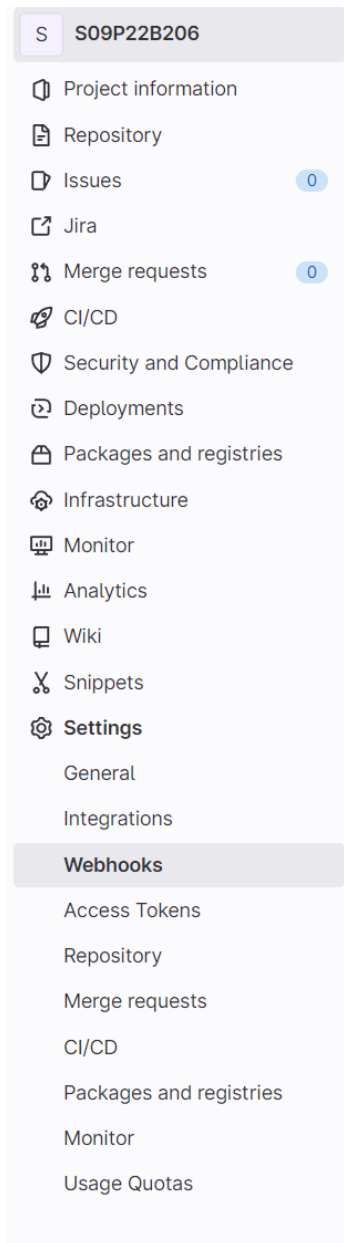
2b9d270a9c20fa96ed67bb524bbbc763

Generate

→ generate를 눌러 Secret token 생성

3.6 Gitlab webhook 설정

- jenkins 작업물의 변화를 감지하여 build, run 하기 위해서는 webHook 필수!
- gitlab project → settings → webhooks



- url : <http://j9b206.p.ssafy.io:9090/project/gappa-back>
- secret token: jenkins System에서 받아온 token 입력
- merge를 할때마다 요청

Webhook

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if it contains one or more special characters.

☒ Show full URL

☐ Mask portions of URL

Do not show sensitive data such as tokens in the UI.

Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

☐ Push events

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

☐ Confidential comments

A comment is added to a confidential issue.

☐ Issues events

An issue is created, updated, closed, or reopened.

☐ Confidential issues events

A confidential issue is created, updated, closed, or reopened.

☒ Merge request events

A merge request is created, updated, or merged.

☐ Job events

A job's status changes.

☐ Pipeline events

A pipeline's status changes.

☐ Wiki page events

A wiki page is created or updated.

☐ Deployment events

A deployment starts, finishes, fails, or is canceled.

☐ Feature flag events

A feature flag is turned on or off.

☐ Releases events

A release is created or updated.

- ssl verification


SSL verification

☒ Enable SSL verification

[Save changes](#)

[Test](#) ▾

- Test : 200이어야 함!

 Hook executed successfully: HTTP 200

[Save changes](#)

[Test](#) ▾

4. Back-End 빌드 및 배포

4-1 Back-end 환경변수

```
spring:
  # 로그파일 설정
  application:
    name: svc1-accounts
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
```

```

url: jdbc:mysql://j9B206.p.ssafy.io:2231/gappa?serverTimezone=UTC&characterEncoding=UTF-8
username: gappa
password: bqefip4o407wgd6-c4sc-zzhpf11-vv1a
redis:
  host: j9b206.p.ssafy.io
  port: 9707
  password: gappa

batch:
  jdbc:
    initialize-schema: never
  job:
    enabled: false

jpa:
  database: mysql
  database-platform: org.hibernate.dialect.MySQL8Dialect
  show-sql: true
  hibernate:
    ddl-auto: validate
    use-new-id-generator-mappings: false
  properties:
    hibernate:
      format_sql: true
      default_batch_fetch_size: 1000

# sms
accessKey: fM7dsWkyM4SRx3dihNM
secretKey: lk1JtbeKB4fR7WcNrbZb7gL4LpQFp27b2msw5RzF
serviceId: ncp:sms:kr:312413481836:gappa

# system number
systemPhoneNumber: 01073877808

# jwt
jwt:
  secret:
    key: 89525749bcff2dc9c717e6014d03d342fe93d46de7fb6b0915c930ed3e675f663adb3932cd145044aba992c3fa3d501e39df232a077319b5f235be80d9647:

# firebase
fcm:
  certification: certification.json

```

4-2 자바 및 gradle 설치

```

# 자바 설치
sudo apt update
sudo apt install openjdk-11-jdk
java -version

# gradle 설치
cd /S09P22B206/BackEnd/gappa
sh gradlew

# gradle 실행
sh gradlew build

```

4-3 Dockerfile 작성

```

FROM openjdk:11-jre-slim
COPY /build/libs/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]

```

4-4 Jenkinsfile 작성

```

pipeline {
  agent any

  environment {
    CONTAINER_NAME = "gappa-back"
    IMAGE_NAME = "gappa"
  }
}

```

```

stages {
    stage('Build') {
        steps {
            dir('Backend/gappa') {
                sh 'chmod +x gradlew'
                sh './gradlew clean build'
            }
        }
        post {
            success {
                echo 'gradle build success'
            }
            failure {
                echo 'gradle build failed'
            }
        }
    }

    stage('Docker Delete') {
        steps {
            script {
                try{
                    sh 'echo "Docker Delete Start"'
                    // 컨테이너 존재 시 삭제
                    sh "docker stop ${CONTAINER_NAME}"
                    sh "docker rm -f ${CONTAINER_NAME}"
                }catch (Exception e){
                    echo "Docker container ${CONTAINER_NAME} does not exist. skip"
                }
                try{
                    // 이미지 존재 시 삭제
                    sh "docker image rm ${IMAGE_NAME}"
                }catch (Exception e){
                    echo "Docker image ${IMAGE_NAME} does not exist. skip"
                }
            }
        }
        post {
            success {
                sh 'echo "Docker delete Success"'
            }
            failure {
                sh 'echo "Docker delete Fail"'
            }
        }
    }

    stage('Dockerizing'){
        steps{
            sh 'echo " Image Bulid Start"'
            // 도커 이미지를 기반으로 컨테이너 빌드
            dir('Backend/gappa') {
                sh 'docker build -t ${IMAGE_NAME} .'
            }
        }
        post {
            success {
                sh 'echo "Bulid Docker Image Success"'
            }
            failure {
                sh 'echo "Bulid Docker Image Fail"'
            }
        }
    }
}

stage('Deploy') {
    steps {
        script{
            sh 'docker run --name ${CONTAINER_NAME} -d -p 8080:8080 ${IMAGE_NAME}'
        }
    }
    post {
        success {
            echo 'Deploy success'
        }
        failure {
            echo 'Deploy failed'
        }
    }
}
}
}

```

4-5 Jenkins 설정

Pipeline

Definition

Pipeline script from SCM

SCM ?
Git

Repositories ?

Repository URL ?
https://lab.ssfy.com/s09-fintech-finance-sub2/S09P228206.git

Credentials ?
stella9808@naver.com/*****

+ Add +

그룹 ▼

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?
*/develop-be

Add Branch

Repository browser ?
(자동)

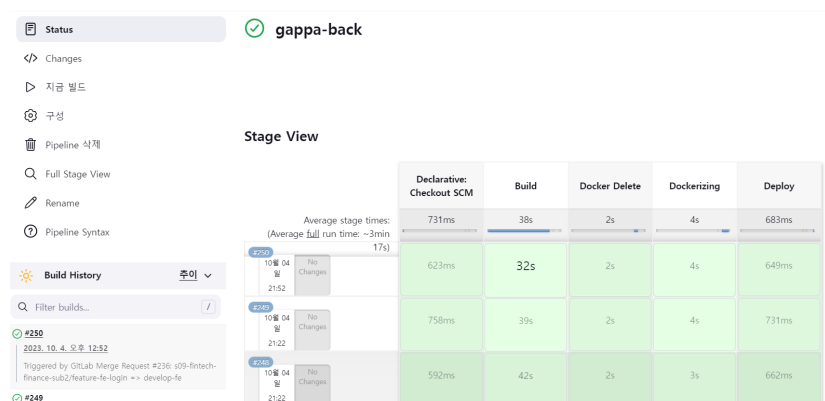
Additional Behaviours
Add ▼

Script Path ?
BackEnd/gappa/jenkinsFolder/jenkinsfile

☒ Lightweight checkout ?

[Pipeline Syntax](#)

- 빌드 실행해 확인



5. Front-End 빌드 및 배포

5-1 Dockerfile 작성

```
FROM node:14.21.3
WORKDIR /app
COPY package*.json ./
RUN npm install
RUN npm install -g serve
COPY . .
```



```
RUN npm run build
ENTRYPOINT ["serve", "-s", "build"]
```

5-2 Jenkinsfile 작성

```
pipeline {
    agent any

    tools {nodejs "node"}

    environment {
        DOCKER = 'sudo docker'
        TIME_ZONE = 'Asia/Seoul'
        TAG = "docker-react:${env.BUILD_ID}"
    }

    stages {
        stage('prepare') {
            steps {
                dir('FrontEnd'){
                    sh 'npm install'
                }
            }
        }
        stage('build') {
            steps {
                dir('FrontEnd'){
                    sh 'npm run build'
                    sh '''
                    echo 'Docker image build'
                    docker build --no-cache -t $TAG .
                    '''
                }
            }
        }
        stage('Deploy') {
            steps {
                dir('FrontEnd'){
                    script {
                        try {
                            sh 'docker stop GappaFront'
                            sh 'docker rm GappaFront'
                        } catch (Exception e) {
                            echo "Failed to stop or remove Docker container, proceeding anyway"
                        }
                    }
                    sh '''
                    echo 'Deploy'
                    docker run -d -p 3000:3000 -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul --name GappaFront $TAG
                    '''
                }
            }
        }
    }
}
```

5-3 Jenkins 설정

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://lab.ssafty.com/s09-fintech-finance-sub2/509P228206.git

Credentials ?

stella9808@naver.com/*****

+Add +

그룹

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/develop-fe

Add Branch

Repository browser ?

(자동)

Additional Behaviours

Add

Script Path ?

FrontEnd/Jenkinsfile

☒ Lightweight checkout ?

[Pipeline Syntax](#)

- jenkins 관리 → Tools → NodeJs 설정
- node 버전과 이름을 입력

NodeJS installations

NodeJS installations ^ Edited

Add NodeJS

NodeJS

Name

node

☒ Install automatically ?

Install from nodejs.org

Version

NodeJS 14.21.3

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'

Global npm packages refresh hours

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72

Add Installer

Save

Apply

- 지금 빌드 실행하여 확인

Status

</> Changes

▷ 지금 빌드

⚙️ 구성

🗑️ Pipeline 삭제

🔍 Full Stage View

✎ Rename

🔗 Pipeline Syntax

Build History

추이

Filter builds...

#240

2023. 10. 4. 오후 12:52

Triggered by GitLab Merge Request #236: s09-fintech-finance-sub2/feature-fe-login => develop-fe

gappa-front

Stage View

Average stage times:

(Average full run time: ~6min 36s)

	Declarative: Checkout SCM	Declarative: Tool Install	prepare	build	Deploy
#240	900ms	105ms	16s	4min 2s	1s
#239	824ms	95ms	15s	4min 0s	1s
#238	1s	90ms	15s	4min 38s	1s
#237	1s	116ms	18s	4min 9s	1s

6. 외부 서비스

6.1 Naver 문자 서비스

네이버 sens 서비스

Simple & Easy Notification Service

Update

SMS, PUSH, 알림톡 등 메시지 알림 기능을 구현하는 서비스

이용 신청하기

요금 계산하기

📞

특징

상세 기능

요금

사용 가이드

알림 및 메시지 기능을 구현하는 쉬운 방법

웹/모바일, iOS / Android 상관없이 다양한 유형의 알림 기능을 적용할 수 있는 서비스입니다.

효율적인 프로젝트 관리

프로젝트를 생성하고 메타 정보를 입력하면 메시지 및 알림 전송 기능을 즉시 사용할 수 있습니다. 각 단계에 필요한 정보를 입력한 후에는 메시지

다양한 발송 옵션

예약 발송 기능을 통해 메시지 발송 스케줄을 관리할 수 있습니다. 또한 알림톡/친구톡에 대한 SMS Fail over 기능을 지원하여, 카카오톡이 설치

다양한 연동 방법

네이버 클라우드 플랫폼에서 제공하는 서비스와의 연동은 물론 APNS(iOS), GCM/FCM(Android), SMS, LMS 외에도 카카오톡 비즈 메시지(알림톡,

API URL

<https://sens.apigw.ntruss.com/sms/v2>

NAVER Cloud Platform 인증키 및 Signature 생성

ncloud 회원가입(<https://www.ncloud.com/>)

인증키 관리 : 네이버 클라우드 플랫폼 메인 페이지 > 마이페이지 > 인증키 관리

> 신규 API 인증키 생성

NAVER CLOUD PLATFORM 소개 서비스 솔루션 마켓플레이스 요금 고객지원 파트너 가이드센터 마이페이지 문의하기 콘솔

이용관리 계정관리 결제관리 나의 문의내역 알림 관리 솔루션 이용 현황

계정 관리

회원정보 변경 비밀번호 변경 파트너 관리 보안 설정 SNS 연동 계정 변경 **인증키 관리** 회원 탈퇴

네이버 클라우드 플랫폼은 제공하는 서비스를 안전하게 이용하도록 회원별 API 인증키를 발급하고 있습니다.
API 인증키는 API를 호출한 사용자가 권한을 가진 사용자인지 식별하는 도구입니다.

API 인증키 이용안내

- API 인증키를 도용되었다고 의심될 때는 기존 API 인증키를 삭제하고 새 API 인증키를 생성하세요.
- 새 API 인증키를 생성한 다음에는 반드시 사용하고 있는 서비스에 변경된 API 인증키를 적용해야 합니다.
- API 인증키를 사용하지 않을 때는 '사용 중지'로 설정할 수 있으며, 삭제는 '사용 중지' 상태에서에서만 가능합니다.

API 인증키 관리

신규 API 인증키 생성

Access Key ID	Secret Key	생성일자	상태	관리
---------------	------------	------	----	----

SENS 서비스 신청하고 서비스 ID 발급받기

서비스 이용 신청하기

Simple & Easy Notification Service Update

SMS, PUSH, 알림톡 등 메시지 알림 기능을 구현하는 서비스

이용 신청하기 요금 계산하기

특징 상세 기능 요금 사용 가이드

콘솔 > SENS 메뉴에서 프로젝트 생성 후, 우측 열쇠 아이콘을 클릭

Project

+ 프로젝트 생성하기
상품 더 알아보기
OPEN API 가이드

② 처음이에요. 무엇을 해야하나요 ?

- 1 메시지를 발송하기 위해서는 먼저 프로젝트를 등록 해야 합니다.
- 2 프로젝트를 등록하기 위해 하단의 생성하기 버튼을 클릭하세요.
- 3 이름과 설명을 입력하고, 사용할길 원하는 발송 서비스를 선택한 뒤, 생성을 완료합니다.
- 4 테이블 서비스 컬럼의 SMS, PUSH, Biz Message를 클릭하시면, 메시지 발송 페이지로 이동 합니다.
- 5 서비스 ID 컬럼의 를 클릭하시면, 서비스 ID를 확인하실 수 있습니다.

수정 삭제

이름	설명	서비스	서비스 ID
 waterbell		SMS	

메시지

메시지 발송

SMS / LMS / MMS 메시지를 발송

요청 URL

POST <https://sens.apigw.ntruss.com/sms/v2/services/{serviceId}/messages>

Content-Type: application/json; charset=utf-8
x-ncp-apigw-timestamp: {Timestamp}
x-ncp-iam-access-key: {Sub Account Access Key}
x-ncp-apigw-signature-v2: {API Gateway Signature}

Path Variables

항목	Mandatory	Type	설명	비고
serviceId	Mandatory	String	서비스 아이디	프로젝트 등록 시 발급받은 서비스 아이디

Headers

API Header

항목	Mandatory	설명
Content-Type	Mandatory	요청 Body Content Type을 application/json으로 지정 (POST)
x-ncp-apigw-timestamp	Mandatory	- 1970년 1월 1일 00:00:00 협정 세계시(UTC)부터의 경과 시간을 밀리초 (Millisecond)로 나타냄- API Gateway 서버와 시간 차가 5분 이상 나는 경우 유효하지 않은 요청으로 간주
x-ncp-iam-access-key	Mandatory	포털 또는 Sub Account에서 발급받은 Access Key ID
x-ncp-apigw-signature-v2	Mandatory	- 위 예제의 Body를 Access Key Id와 맵핑되는 SecretKey로 암호화한 서명- HMAC 암호화 알고리즘은 HmacSHA256 사용

요청 Body

```
{
  "type": "(SMS | LMS | MMS)",
  "contentType": "(COMM | AD)",
  "countryCode": "string",
  "from": "string",
  "subject": "string",
  "content": "string",
  "messages": [
    {
      "to": "string",
      "subject": "string",
      "content": "string"
    }
  ],
  "files": [
    {
      "fileId": "string"
    }
  ],
  "reserveTime": "yyyy-MM-dd HH:mm",
  "reserveTimeZone": "string"
}
```

항목	Mandatory	Type	설명	비고
type	Mandatory	String	SMS Type	SMS, LMS, MMS (소문자 가능)
contentType	Optional	String	메시지 Type	- COMM: 일반메시지- AD: 광고메시지- default: COMM
countryCode	Optional	String	국가 번호	- SENS에서 제공하는 국가로의 발송만 가능- default: 82- 국제 SMS 발송 국가 목록
from	Mandatory	String	발신번호	사전 등록된 발신번호만 사용 가능
subject	Optional	String	기본 메시지 제목	LMS, MMS에서만 사용 가능- LMS, MMS: 최대 40byte
content	Mandatory	String	기본 메시지 내용	- SMS: 최대 80byte- LMS, MMS: 최대 2000byte
messages	Mandatory	Object	메시지 정보	- 아래 항목 참조 (messages.XXX)- 최대 100개
messages.to	Mandatory	String	수신번호	불임표 (-)를 제외한 숫자만 입력 가능
messages.subject	Optional	String	개별 메시지 제목	LMS, MMS에서만 사용 가능- LMS, MMS: 최대 40byte
messages.content	Optional	String	개별 메시지 내용	- SMS: 최대 80byte- LMS, MMS: 최대 2000byte
files.fileId	Optional	String	파일 아이디	MMS에서만 사용 가능 파일 업로드 참조
reserveTime	Optional	String	예약 일시	메시지 발송 예약 일시 (yyyy-MM-dd HH:mm)
reserveTimeZone	Optional	String	예약 일시 타임존	- 예약 일시 타임존 (기본: Asia/Seoul)- 지원 타임존 목록 - TZ database name 값 사용

응답 Body

```
{
  "requestId": "string",
  "requestTime": "string",
  "statusCode": "string",
  "statusName": "string"
}
```

항목	Mandatory	Type	설명	비고
requestId	Mandatory	String	요청 아이디	
requestTime	Mandatory	DateTime	요청 시간	yyyy-MM-dd'T'HH:mm:ss.SSS
statusCode	Mandatory	String	요청 상태 코드	- 202: 성공- 그 외: 실패- HTTP Status 규격을 따름
statusName	Mandatory	String	요청 상태명	- success: 성공- fail: 실패

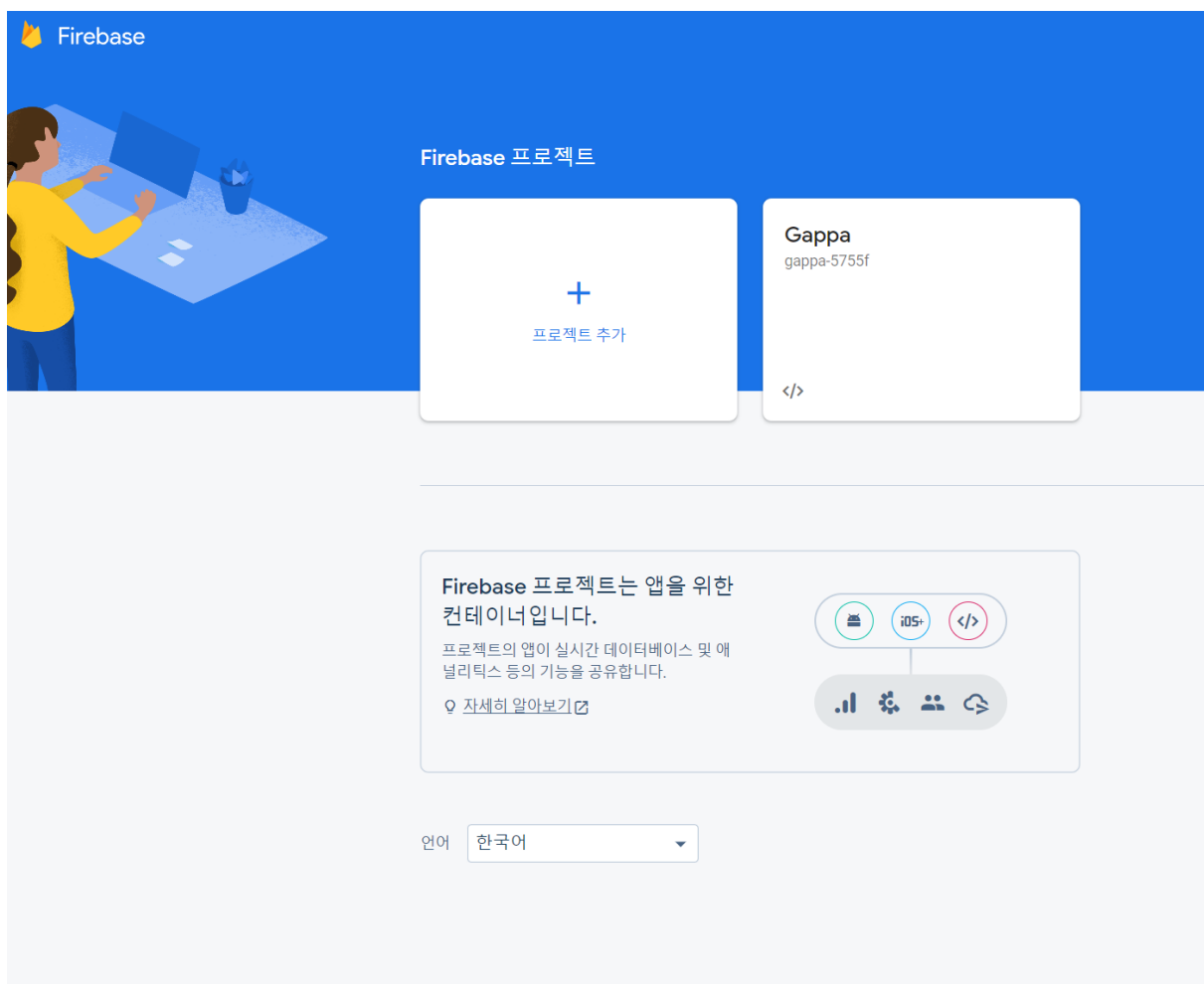
응답 Status

HTTP Status	Desc
202	Accept (요청 완료)
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
429	Too Many Requests
500	Internal Server Error

6-2. FireBase

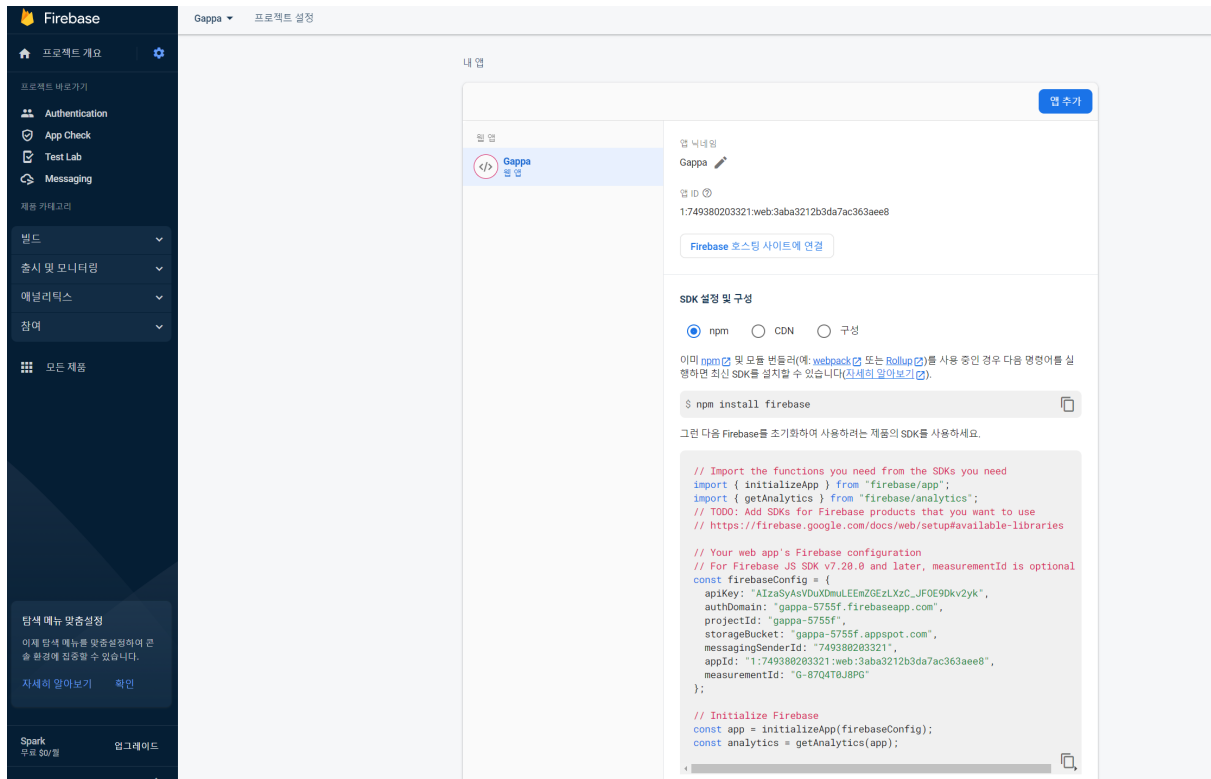
FireBase 프로젝트 생성

<https://console.firebase.google.com/>



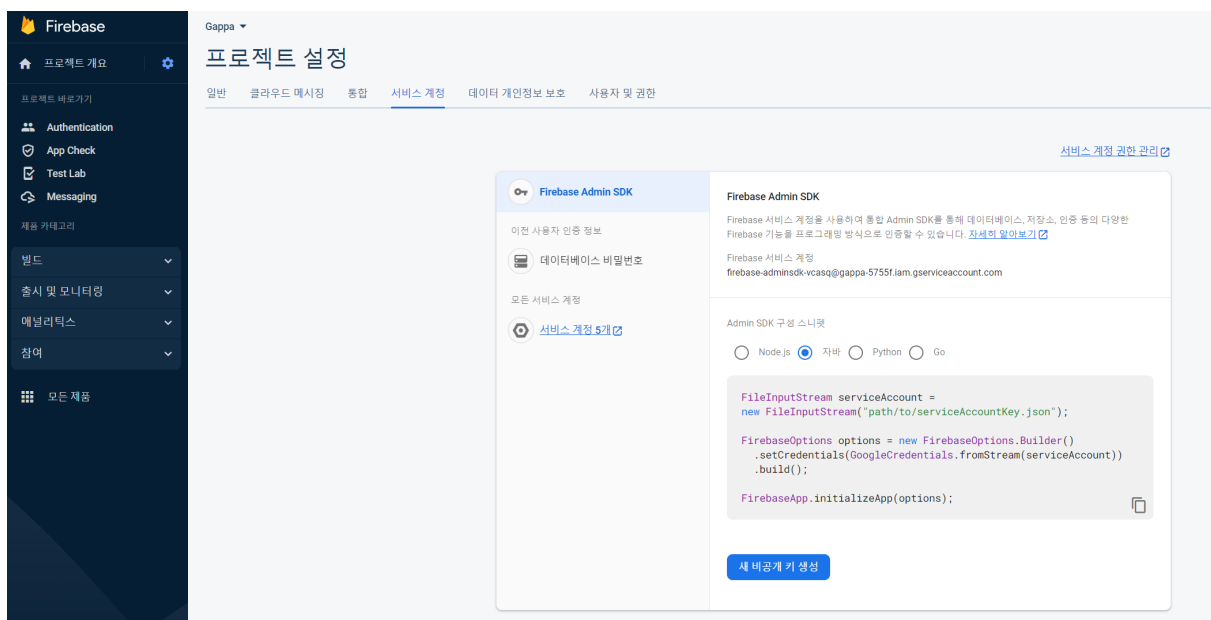
웹 앱의 구성 스니펫 가져오기

1. Firebase에 로그인하고 프로젝트를 열기
2. 개요 페이지에서 앱 추가를 클릭
3. 웹 앱에 Firebase 추가를 선택
4. 스니펫을 복사하여 애플리케이션 HTML에 추가



비밀 KEY 파일 생성

생성한 프로젝트 페이지 > 설정 > 서비스 계정항목




```

1  {
2      "type": "service_account",
3      "project_id": "gappa-5755f",
4      "private_key_id": "71d9e8ba817b6f453878de72c00cb607ba6e17d8",
5      "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCvawC5eiZiQyIr\nZc7wNN6MPEVm0bdW8",
6      "client_email": "firebase-adminsdk-vcasq@gappa-5755f.iam.gserviceaccount.com",
7      "client_id": "111895411385507188327",
8      "auth_uri": "https://accounts.google.com/o/oauth2/auth",
9      "token_uri": "https://oauth2.googleapis.com/token",
10     "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
11     "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-vcasq%40gappa-5755f.iam.gserviceaccount.com",
12     "universe_domain": "googleapis.com"
13 }

```

Spring백엔드 설정

- build.gradle

```

dependencies {
    implementation 'com.google.firebase:firebase-admin:7.1.1'
}

```

- FCM 초기화

```

@Slf4j
@Component
public class FCMInitializer {

    @Value("${fcm.certification}")
    private String googleApplicationCredentials;

    @PostConstruct
    public void initialize() throws IOException {
        ClassPathResource resource = new ClassPathResource(googleApplicationCredentials);

        try (InputStream is = resource.getInputStream()) {
            FirebaseOptions options = FirebaseOptions.builder()
                .setCredentials(GoogleCredentials.fromStream(is))
                .build();

            if (FirebaseApp.getApps().isEmpty()) {
                FirebaseApp.initializeApp(options);
                log.info("FirebaseApp initialization complete");
            }
        }
    }
}

```