Gitab 소스 클론 이후 빌드 및 배포 정리 문서

애플리케이션 환경

> 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전 (IDE 버전 포함)

- JVM : Liberica OpenJDK 17 LTS
- SpringBoot : 3.4.2
- Nginx : 1.27.3
- Docker : 27.5.1
- Jenkins : 2.479.3
- MySql : 8.0.41
- IDE 버전
  - IntelliJ IDEA 2023.3.8
  - Android Studio koala

환경 변수

> 빌드 시 사용되는 환경 변수 내용 상세 기재

```
DOCKER_IMAGE = munbangu-backend
DOCKER_CONTAINER = spring-backend
DOCKER_PORT = 8081
DB_USERNAME = d106
DB_PASSWORD = hellod106
DB_NAME = munbangu
EC2_PUBLIC_IP = i12d106.p.ssafy.io
MYSQL_PORT = 3306
MYSQL_CONTAINER = mysql
JWT_SECRET_KEY = KbPeShVmYq3t6w9z$C&F)H@McQfTjWnZr4u7x!A%D*G-
KaNdRgUkXp2s5v8y/B?E
AWS_S3_ACCESS_KEY = AKIAWAA66I4STYAW2J5G
AWS_S3_SECRET_KEY = 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
```

B.E

Dockerfile

```
# Liberica JDK 17 버전을 기반 이미지로 사용
FROM bellsoft/liberica-openjdk-debian:17

# 컨테이너 내부의 작업 디렉토리 설정
WORKDIR /app

# Gradle 빌드를 통해 생성된 JAR 파일을 컨테이너 내부로 복사
# build/libs/ 디렉토리에 있는 모든 JAR 파일을 app.jar라는 이름으로 복사
COPY build/libs/*.jar app.jar

# 컨테이너 실행 시 자동으로 실행될 명령어 설정
# java -jar app.jar 명령어를 실행하여 스프링부트 애플리케이션 시작
ENTRYPOINT ["java", "-jar", "app.jar"]

# 컨테이너가 사용할 포트 명시 (문서화 목적)
# 실제 포트 바인딩은 docker run 명령어의 -p 옵션으로 설정
EXPOSE 8080
```

Docker Image build

```
docker build -t munbangu-backend:latest .
```

Docker container run

```
docker run -d --name spring-backend -p 8080:8080 munbangu-backend:latest
```

Nginx

`default.conf`

```
# HTTP -> HTTPS 리다이렉트
server {
    listen 80;
    server_name i12d106.p.ssafy.io;
    return 301 https://$server_name$request_uri;
}

# HTTPS 서버 설정
```

```
server {
    listen 443 ssl;
    server_name i12d106.p.ssafy.io;

    # SSL 설정
    ssl_certificate
/etc/letsencrypt/archive/i12d106.p.ssafy.io/fullchain1.pem;
    ssl_certificate_key
/etc/letsencrypt/archive/i12d106.p.ssafy.io/privkey1.pem;

    # 로그 설정
    access_log /var/log/nginx/host.access.log main;
    error_log /var/log/nginx/error.log;

    # 파일 업로드
    client_max_body_size 10M;

    # 기본 location
    location / {
        return 404;
    }

    # Spring Boot API (기본 경로)
    location /api {
        proxy_pass http://spring-backend:8081;
        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
        proxy_connect_timeout 300;
        proxy_send_timeout 300;
        proxy_read_timeout 300;
        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;
    }

    # Swagger UI 리버스 프록시 설정
    location /swagger-ui/ {
        proxy_pass http://spring-backend:8081/swagger-ui/;
        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
        proxy_connect_timeout 300;
        proxy_send_timeout 300;
        proxy_read_timeout 300;
        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;
    }

    # Swagger API 문서 리버스 프록시 설정
    location /v3/api-docs {
        proxy_pass http://spring-backend:8081/v3/api-docs;
        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
        proxy_connect_timeout 300;
        proxy_send_timeout 300;
        proxy_read_timeout 300;
        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;
    }

    # Jenkins
    location /jenkins {
    proxy_pass http://jenkins:8080;

    # 기본 프록시 헤더
    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;

    # Jenkins specific headers
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Port $server_port;
    proxy_set_header X-Forwarded-Context /jenkins;
    proxy_set_header X-Forwarded-Prefix /jenkins;

    # WebSocket 지원
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

    # 타임아웃 설정
    proxy_connect_timeout 300;
    proxy_send_timeout 300;
    proxy_read_timeout 300;
    # 버퍼 크기 설정
    proxy_buffer_size 128k;
    proxy_buffers 4 256k;
    proxy_busy_buffers_size 256k;

    # 정적 파일 캐싱 설정
    proxy_cache_use_stale error timeout http_500 http_502 http_503 http_504;
    proxy_cache_valid 200 302 1h;
    proxy_cache_valid 404 1m;
}

# Jenkins의 정적 파일을 위한 추가 location 블록
location ~* ^/jenkins/static/(.*)$ {
    proxy_pass http://jenkins:8080/jenkins/static/$1;
    proxy_set_header Host $host;
    proxy_cache_valid 200 302 24h;
    proxy_cache_valid 404 1m;
}

    # 에러 페이지
    error_page 404 /404.html;
    error_page 500 502 503 504 /50x.html;
}
```

MySQL

Docker Volume 생성

```
docker volume create mysql-volume
```

Docker Container run

```
docker run --rm -d --name mysql \
--network app-network \
-p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=${DB_PASSWORD} \
-e MYSQL_DATABASE=${DB_NAME} \
-v mysql_data:/var/lib/mysql \
mysql:8.0.41
```

AWS S3

Docker Container run

```
docker run -d --name ${DOCKER_CONTAINER} \
--network app-network \
-e TZ=Asia/Seoul \
-e SPRING_PROFILES_ACTIVE=prod,ENV \
-e MYSQL_CONTAINER=${MYSQL_CONTAINER} \
-e JWT_SECRET_KEY=${JWT_SECRET_KEY} \
-e DOCKER_PORT=${DOCKER_PORT} \
-e EC2_PUBLIC_IP=${EC2_PUBLIC_IP} \
-e MYSQL_PORT=${MYSQL_PORT} \
-e DB_NAME=${DB_NAME} \
-e DB_USERNAME=${DB_USERNAME} \
-e DB_PASSWORD=${DB_PASSWORD} \
-e AWS_ACCESS_KEY=${AWS_ACCESS_KEY} \
-e AWS_SECRET_KEY=${AWS_SECRET_KEY} \
${DOCKER_IMAGE}:latest
```

Jenkinsfile

/backend_project/Jeniknsfile

```
 pipeline {
        agent any

        environment {
            DOCKER_IMAGE = credentials('DOCKER_IMAGE')
            DOCKER_CONTAINER = credentials('DOCKER_CONTAINER')
            DOCKER_PORT = credentials('DOCKER_PORT')
            DB_USERNAME = credentials('DB_USERNAME')
            DB_PASSWORD = credentials('DB_PASSWORD')
            DB_NAME = credentials('DB_NAME')
            EC2_PUBLIC_IP = credentials('EC2_PUBLIC_IP')
            MYSQL_PORT = credentials('MYSQL_PORT')
            MYSQL_CONTAINER = credentials('MYSQL_CONTAINER')
            JWT_SECRET_KEY = credentials('JWT_SECRET_KEY')
            AWS_S3_ACCESS_KEY = credentials('AWS_S3_ACCESS_KEY')
            AWS_S3_SECRET_KEY = credentials('AWS_S3_SECRET_KEY')
        }

        stages {
            stage('Checkout') {
                steps {
                    checkout scm
                }
            }


            stage('Build') {
                steps {
                    dir('backend_project') {
                        sh 'chmod +x ./gradlew'
                        sh './gradlew clean build -x test'  // 테스트 스킵 -> 추
후에 테스트 작성해야 할 땐 -x test 빼기
                        //sh './gradlew clean build'
                    }
                }
            }

            stage('Initial Infrastructure Setup') {
                steps {
                    script {
                        // 네트워크 생성 (없는 경우에만)
                        sh 'docker network create app-network || true'

                        // MySQL이 실행 중이 아닐 때만 실행
                        sh '''
                            if [ -z "$(docker ps -q -f name=mysql)" ]; then
                                echo "Starting MySQL container..."
                                docker run -d --name ${MYSQL_CONTAINER} \
                                    --network app-network \
                                    -e MYSQL_ROOT_PASSWORD=${DB_PASSWORD} \
                                    -e MYSQL_DATABASE=${DB_NAME} \
                                    -e TZ=Asia/Seoul \
                                    -v mysql_data:/var/lib/mysql \
                                    -p ${MYSQL_PORT}:${MYSQL_PORT} \
                                    mysql:8.0.41
                            else
                                echo "MySQL is already running."
                            fi
                        '''

                        // Nginx가 실행 중이 아닐 때만 실행
                        // HTTPS 포트(443) 세팅 및 인증서 파일 위치 볼륨 마운트 설정
```

버전

```
                        sh '''
                            if [ -z "$(docker ps -q -f name=nginx)" ]; then
                                echo "Starting Nginx container..."
                                docker run -d --name nginx \
                                  --network app-network \
                                  -e TZ=Asia/Seoul \
                                  -p 80:80 -p 443:443 \
                                  -v /etc/letsencrypt:/etc/letsencrypt:ro \
                                  -v /etc/nginx/conf.d:/etc/nginx/conf.d \
                                  nginx:latest
                            else
                                echo "Nginx is already running."
                            fi
                        '''
                    }
                }
            }


            stage('Deploy Application') {
                steps {
                    dir('backend_project') {
                        sh '''
                            echo "Stopping existing Spring Boot container..."
                            if [ ! -z "$(docker ps -q -f
name=${DOCKER_CONTAINER})" ]; then
                                docker stop ${DOCKER_CONTAINER} || true
                                docker rm ${DOCKER_CONTAINER} || true
                            fi

                            echo "Building new Docker image..."
                            docker build -t ${DOCKER_IMAGE}:latest .

                            echo "Starting new Spring Boot container..."
                            docker run -d --name ${DOCKER_CONTAINER} \
                                --network app-network \
                                -e TZ=Asia/Seoul \
                                -e SPRING_PROFILES_ACTIVE=prod,ENV \
                                -e MYSQL_CONTAINER=${MYSQL_CONTAINER} \
                                -e JWT_SECRET_KEY=${JWT_SECRET_KEY} \
                                -e DOCKER_PORT=${DOCKER_PORT} \
                                -e EC2_PUBLIC_IP=${EC2_PUBLIC_IP} \
                                -e MYSQL_PORT=${MYSQL_PORT} \
                                -e DB_NAME=${DB_NAME} \
                                -e DB_USERNAME=${DB_USERNAME} \
                                -e DB_PASSWORD=${DB_PASSWORD} \
                                -e AWS_S3_ACCESS_KEY=${AWS_S3_ACCESS_KEY} \
                                -e AWS_S3_SECRET_KEY=${AWS_S3_SECRET_KEY} \
                                ${DOCKER_IMAGE}:latest
                        '''
                    }
                }
            }
        }


        post {
            success {
                echo 'Deployment successful!'
                echo 'MySQL status:'
                sh 'docker ps -f name=mysql'
                echo 'Nginx status:'
                sh 'docker ps -f name=nginx'
                echo 'Spring Boot status:'
                sh 'docker ps -f name=${DOCKER_CONTAINER}'
            }
            failure {
                echo 'Deployment failed!'
            }
            always {
                cleanWs()
            }
        }
    }
```

배포 시 특이사항

> 배포 시 특이사항 기재

보안 설정

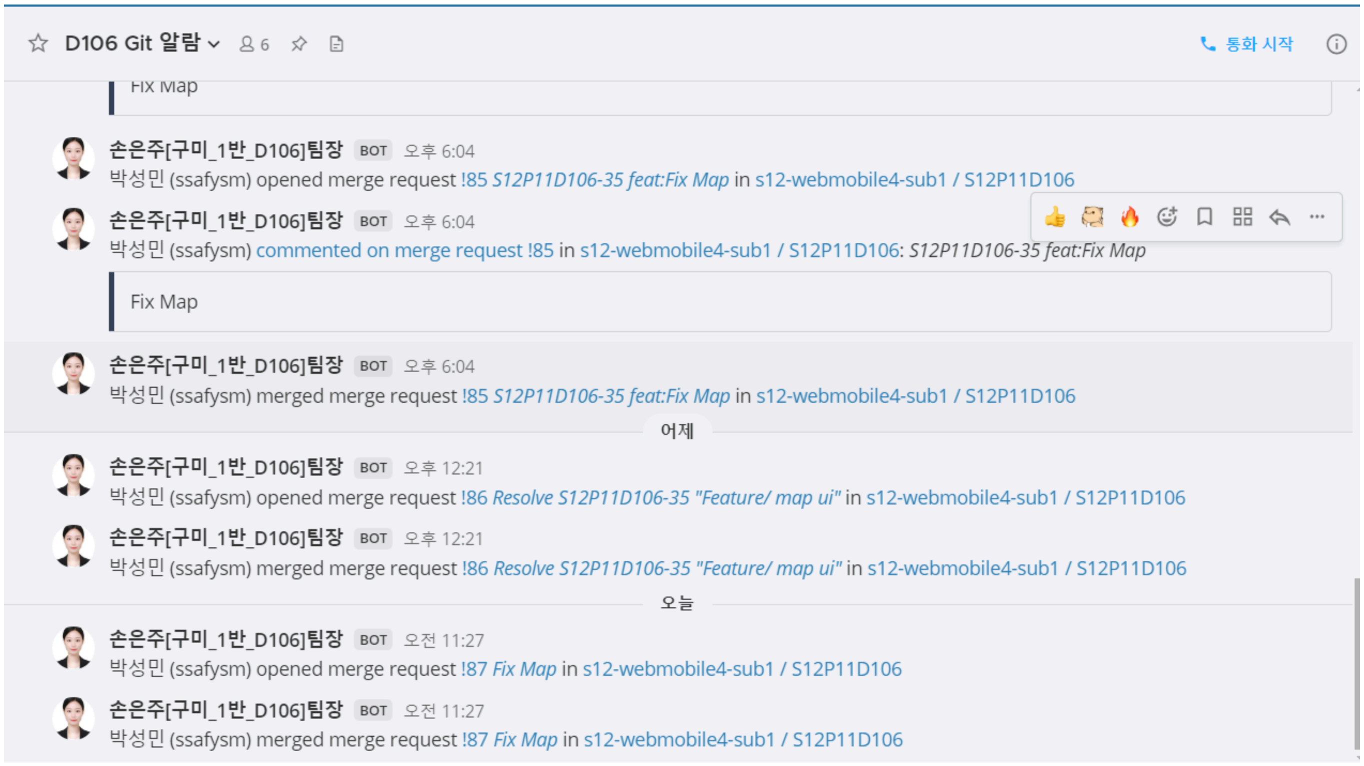
1. 컨테이너 접근 제한
   - 모든 서비스(Jenkins, MySQL, Spring Boot)는 Nginx 리버스 프록시를 통해서만 접근 가능
   - Docker network `app-network` 를 통한 컨테이너 간 통신
2. 방화벽 설정

- UFW를 통해 필요한 포트만 허용
- HTTP(80)에서 HTTPS(443)로 자동 리다이렉트 설정
- 허용된 포트:
  - 22 (SSH)
  - 80 (HTTP -> HTTPS 리다이렉트)
  - 443 (HTTPS)

Mattermost Bot 생성

다음 이벤트에 대한 알림을 받도록 설정:

- Merge requests
  - 머지 요청이 열릴 때 (opened)
  - 머지가 완료될 때 (merged)



Fix Map

손은주[구미_1반_D106]팀장 BOT 오후 6:04
박성민 (ssafysm) opened merge request !85 *S12P11D106-35 feat:Fix Map* in s12-webmobile4-sub1 / S12P11D106

손은주[구미_1반_D106]팀장 BOT 오후 6:04
박성민 (ssafysm) commented on merge request !85 in s12-webmobile4-sub1 / S12P11D106: *S12P11D106-35 feat:Fix Map*

Fix Map

손은주[구미_1반_D106]팀장 BOT 오후 6:04
박성민 (ssafysm) merged merge request !85 *S12P11D106-35 feat:Fix Map* in s12-webmobile4-sub1 / S12P11D106

어제

손은주[구미_1반_D106]팀장 BOT 오후 12:21
박성민 (ssafysm) opened merge request !86 *Resolve S12P11D106-35 "Feature/ map ui"* in s12-webmobile4-sub1 / S12P11D106

손은주[구미_1반_D106]팀장 BOT 오후 12:21
박성민 (ssafysm) merged merge request !86 *Resolve S12P11D106-35 "Feature/ map ui"* in s12-webmobile4-sub1 / S12P11D106

오늘

손은주[구미_1반_D106]팀장 BOT 오전 11:27
박성민 (ssafysm) opened merge request !87 *Fix Map* in s12-webmobile4-sub1 / S12P11D106

손은주[구미_1반_D106]팀장 BOT 오전 11:27
박성민 (ssafysm) merged merge request !87 *Fix Map* in s12-webmobile4-sub1 / S12P11D106

주요 계정 정보 및 프로퍼티

> DB 접속 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

MySQL

- HOST : i12d106.p.ssafy.io
- PORT : 3306
- Database : munbangu

- ID : d106
- Password : hellod106

Jenkins

- HOST : i12d106.p.ssafy.io/jenkins
- PORT : 8080
- ID : ssafyd106
- PASSWORD : hellod106

Springboot

application-prod.properties

```
server.port=${DOCKER_PORT}

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://${MYSQL_CONTAINER}:${MYSQL_PORT}/${DB_NAME}
?useSSL=false&serverTimezone=UTC
spring.datasource.username=${DB_USERNAME}
spring.datasource.password=${DB_PASSWORD}

app.environment=prod

# JWT Configuration
jwt.secret=${JWT_SECRET_KEY}
# Token validity in milliseconds (24 hours)
jwt.token.validity=86400000

# Amazon S3
spring.cloud.aws.credentials.access-key=${AWS_S3_ACCESS_KEY}
spring.cloud.aws.credentials.secret-key=${AWS_S3_SECRET_KEY}
```

application-dev.properties

```
server.port=8080
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/munbangu?
useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=ssafy
```

```
app.environment=dev
app.domain=http://localhost:8080

# JWT Configuration
jwt.secret=KbPeShVmYq3t6w9z$C&F)H@McQfTjWnZr4u7x!A%D*G-KaNdRgUkXp2s5v8y/B?E
# Token validity in milliseconds (24 hours)
jwt.token.validity=86400000

mybatis.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl

# Amazon S3
spring.cloud.aws.credentials.access-key=AKIAWAA66I4SRO4T4OBO
spring.cloud.aws.credentials.secret-
key=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
spring.cloud.aws.region.static=ap-southeast-2
spring.cloud.aws.s3.bucket=munbangu
spring.cloud.aws.stack.auto=false
```

application.properties

```
spring.application.name=munbangu

# DTO, VO
mybatis.type-aliases-package=com.ssafy.model.entity

# mapper.xml
mybatis.mapper-locations=classpath:mapper/*.xml

# Swagger
springdoc.swagger-ui.path=/swagger-ui.html
springdoc.api-docs.path=/v3/api-docs
springdoc.packages-to-scan=com.ssafy
springdoc.paths-to-match=/api/**

# photo mission
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB

spring.web.resources.static-locations=classpath:/static/

server.servlet.encoding.charset=UTF-8
server.servlet.encoding.force=true

# Amazon S3 region
spring.cloud.aws.region.static=ap-northeast-2
spring.cloud.aws.s3.bucket=munbangu-seoul
```

application-ENV.properties

```
firebase.service-key.type=service_account
firebase.service-key.project-id=munbangu-4a922
firebase.service-key.private-key-id=2f5b0a99ecfd071d945d9a9d369e1e60fa56edcb
firebase.service-key.private-key=-----BEGIN PRIVATE KEY-----
\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDkLeLhPiMrjJt2\n987mEwId2f
0Og+9+aBr1Vw3XRHUHVsxGo8ZhWyd58ko8VPOxyFJE3v0kCtRPB1h7\nY1/bWvYQ5FB1zXgVBqpVKa
oaVPP6JIevup+/Pm+SivaDzjA2hdmMgJU5kEXd/C/R\nI+n5jnWFynoXBtCMShSXex3JX7Ezmdn90u
MwE++GBFPiLS+V241s/ixm1By++hRq\n3L4BpOGC+FQ+Buj/Eb/z72XPueJ1djgl2aY3gJGxPv8sYS
nrS+Mw9hJ0r+Tlv1Jn\ngcCz27rysoqY9vdB9lorQ4G0VfKdFOS9R/0CVhkDkmE/hcS7l3toh1D+ir
bcaJi0\n1mRptCaJAgMBAAECggEADIP3zc2lYDN2KNsEHi/XYzZxtY9xs3KXsp1JGmWcMFWH\nyYLH
fs+L0BnaDd/FqMVevdL9T4dzMw0pbUXQxtdWZ9bIva1nc2p01/b7vYtWAS0V\nGYPC/1pirvGwU8pw
xTd87Jf2QhIHwwVHb3voztIMNX/FnT23x/odkbF5u3HkO6WP\neL+bSuLEE4HokX+jAJHNLBP5+Q6u
n/mDIwH8vXiCCJXeyRuy3yTvEh0NEkTTKFg/\nGm8vpORYFNTGNDgeOtDRFjjqK4vOtkkU98Eyi5fy
mM0ZWK9CDcbLuHVG+hKcmLvD\nhEKAU/PHryhDxFcV0/RhwxLedlzZW+Ahblb6NmstgQKBgQD+HqXe
/ik9MbjQ7jc9\nRjKBJww9Gs2I5wZL7FVmHuvDblNtwH+rJbywkGSj/Ws0zx/bJnYss19hbWHh/gzn
\nNGZKlcIXCSzinl1VAmxpJZVswwe8uv+Z21/P33OXwFHxmEgdHKuVXkzm+kxfQlyp\nWvW42EGBKI
hKMETmJKu7VwSeAwKBgQDl3hoeCQjGxdC6i04f4AfI0ODKql1js6tl\nOo4ys6GalZm9itXFbs5Ztj
1U3G1v8LrdkPtyAlOKY5zl2JvfXznUHei/2Z0+5ruW\nBhGR5aVjyGCWAEwPXlyCwLnsqIICfV4jiu
mP0nYiolhpKZYmvtw5rTButCjObMsC\n0IRy8aMZgwKBgQDywjZ0+1pMAkbrGNaea4Pty26upP86TH
DAcOn4H4vca3W5wA52\nYhuKlLX/zyTuEvv8sBZuh4CToxnB+Z8789vQXpZGyVYeu2ivwBvuqp+/ge
TPT9jD\nk0VJYM6dZnUhlfc2EGB91dMjSHNTASNiCen5hmW1TBI+xTPco6WG1w8rHwKBgQCa\nxCB3
ioOjOXrgrnxqT69OEPzY1z3LrCPBI1ysXG95IjKCAKEPrhw3INFIeqUgUkeu\neDgjudc/fPQdeOrx
yS6pV33m9gC+YiF40BsyTRk/BqGpPQf47QKCcnp8EbR/MqOY\ncKXy1LP/e6jlHT/gVG74UlgioSA8
Y9mHypbywSicOwKBgFRzI0/qLlWfWhjFrk30\n6gXiJXVHWSDuUHMqCWbrSwvIPDScXTkrUYLnJQrR
kQ7IFKC0l2yXgnisnYW/LOm4\nwTYMZOqqGNQJbRB9sz65WpbD6kHbKOVLD+fsnK9uBEpZxeYU4nVb
TOSxNhpk9Zah\n+y8XbL81Idxbcc/ElQX2wzeT\n-----END PRIVATE KEY-----\n
firebase.service-key.client-email=firebase-adminsdk-fbsvc@munbangu-
4a922.iam.gserviceaccount.com
firebase.service-key.client-id=106390883342001672264
firebase.service-key.auth-uri=https://accounts.google.com/o/oauth2/auth
firebase.service-key.token-uri=https://oauth2.googleapis.com/token
firebase.service-key.auth-provider-x509-cert-
url=https://www.googleapis.com/oauth2/v1/certs
firebase.service-key.client-x509-cert-
url=https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-
```

```
fbsvc%40munbangu-4a922.iam.gserviceaccount.com
```