포팅매뉴얼

1️⃣ 사용도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- CI/CD : Jenkins
- UCC : Movavi

2️⃣ 개발 환경 및 기술 스택

🎮 CLIENT

Game Client(Unity)

- Unity : 2021.3.33f1
  - MessagePack: 2.5.140

Web Client

- Node js: 20.10.0
- TypeScript: ^4.9
- React: ^18.3.0
  - axios: ^1.6.8
  - styled-components: ^6.1.8
  - Zustand: ^4.5.2 (중앙 상태 관리)
  - React Cookie: ^7.1.4
  - Firebase: ^10.11.1

▼ package.json

```
## 프로젝트 의존성 설정

### 기술 스택 및 라이브러리 목록

#### 프론트엔드 프레임워크 및 라이브러리

- **React**: ^18.3.0
  - JavaScript 라이브러리로, 사용자 인터페이스를 구축하기 위한 것
- **React DOM**: ^18.3.0
  - React와 함께 사용되는 DOM 바인딩 라이브러리
- **React Router**: ^6.23.0
  - React 애플리케이션에서 라우팅을 구현하기 위한 라이브러리
- **React Router DOM**: ^6.23.0
  - 브라우저 환경에서 React Router를 사용할 수 있게 해주는 라이브러리

#### 상태 관리

- **Zustand**: ^4.5.2
  - React 애플리케이션에서 상태 관리를 간편하게 할 수 있게 해주는 라이

#### 스타일링

- **styled-components**: ^6.1.8
```

```
  - CSS-in-JS 라이브러리로, JavaScript 파일 내에서 직접 스타일을 작

#### 애니메이션

- **React Transition Group**: ^4.4.5
  - React 애플리케이션에 애니메이션 효과를 추가하기 위한 라이브러리
- **React Awesome Reveal**: ^4.2.8
  - 스크롤 트리거 애니메이션을 위한 라이브러리

#### HTTP 클라이언트

- **Axios**: ^1.6.8
  - HTTP 요청을 쉽게 할 수 있게 해주는 클라이언트 라이브러리

#### 쿠키 관리

- **React Cookie**: ^7.1.4
  - React 애플리케이션에서 쿠키를 쉽게 관리할 수 있게 해주는 라이브러리

#### Firebase

- **Firebase**: ^10.11.1
  - Google의 Firebase 플랫폼과 통합하기 위한 라이브러리

#### 아이콘

- **React Icons**: ^5.1.0
  - 다양한 아이콘을 React 컴포넌트로 사용할 수 있게 해주는 라이브러리

#### 유틸리티

- **TypeScript**: ^4.9.5
  - JavaScript의 슈퍼셋으로, 정적 타입을 지원하는 언어
- **web-vitals**: ^2.1.4
  - 웹 성능 측정 및 보고를 위한 라이브러리

#### 테스트
```

```
- **@testing-library/jest-dom**: ^5.17.0
  - Jest 테스트에서 DOM 요소의 매처를 제공하는 라이브러리
- **@testing-library/react**: ^13.4.0
  - React 컴포넌트를 테스트하기 위한 라이브러리
- **@testing-library/user-event**: ^13.5.0
  - 사용자 이벤트를 시뮬레이션하는 테스트 라이브러리
- **@types/jest**: ^27.5.2
  - Jest를 TypeScript와 함께 사용하기 위한 타입 정의
- **@types/node**: ^16.18.96
  - Node.js를 TypeScript와 함께 사용하기 위한 타입 정의
- **@types/react**: ^18.3.0
  - React를 TypeScript와 함께 사용하기 위한 타입 정의
- **@types/react-dom**: ^18.3.0
  - React DOM을 TypeScript와 함께 사용하기 위한 타입 정의

#### 개발 스크립트

- **react-scripts**: 5.0.1
  - Create React App에서 사용하는 스크립트와 설정

#### 빌드 및 개발 도구

- **start**: `react-scripts start`
  - 개발 모드에서 애플리케이션을 실행
- **build**: `react-scripts build`
  - 프로덕션 모드에서 애플리케이션을 빌드
- **test**: `react-scripts test`
  - 테스트 러너를 실행
- **eject**: `react-scripts eject`
  - Create React App의 설정을 커스터마이징하기 위해 숨겨진 설정을 노

#### ESLint 구성

- **eslintConfig**: {
  - **extends**: [
    - "react-app"
    - "react-app/jest"
  ]
```

```
}

#### 브라우저 호환성 설정

- **browserslist**: {
  - **production**: [
    - ">0.2%"
    - "not dead"
    - "not op_mini all"
  ]
  - **development**: [
    - "last 1 chrome version"
    - "last 1 firefox version"
    - "last 1 safari version"
  ]
}
```

📲 SERVER

Infra

- AWS EC2: Ubuntu 20.04.6 LTS
- Nginx
- CertBot
- Jenkins 2.444
- MySQL
- MongoDB
- Vault: 1.13.3
- Redis

Cloud

- Firebase
- Google Storage

(Game) Live Server (C++)

- C++: 17
- CMake: 3.11
- Boost Libraries: 1.84.0
- MessagePack: 6.1.1
- Devtools

(Game) Chatting Server (C++)

- C++: 17
- MessagePack: 6.1.1

(Game) Business Server (Java, Spring Boot)

- Java: 17 (JVM : 17.0.10 2024-01-16 LTS)
- Gradle: 8.7
- Spring boot: 3.2.4
  - Lombok
  - Devtools
  - MessagePack: msgpack-core:0.9.4
  - Spring WebFlux
  - MongoDB Reactive

▼ build.gradle

```
dependencies {
    // Spring Boot MongoDB Reactive 지원
    implementation 'org.springframework.boot:spring-boot
-starter-data-mongodb-reactive'

    // Spring WebFlux 지원
    implementation 'org.springframework.boot:spring-boot
-starter-webflux'
```

```
    // Lombok 지원 (컴파일 시)
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    // 개발 환경에서만 사용되는 Spring Boot DevTools
    developmentOnly 'org.springframework.boot:spring-boo
t-devtools'

    // 테스트를 위한 Spring Boot Starter Test
    testImplementation 'org.springframework.boot:spring-
boot-starter-test'

    // Reactor 테스트를 위한 Reactor Test
    testImplementation 'io.projectreactor:reactor-test'

    // MessagePack 지원
    implementation 'org.msgpack:msgpack-core:0.9.4'
}
```

Web(Auth) Server (Java, Spring Boot)

- Java: 17 (17.0.10 2024-01-16 LTS)
- Gradle: 8.7
- Spring boot: 3.2.4
  - springdoc-openapi-starter-webmvc-ui 2.2.0
  - lombok
  - JWT
    - jjwt-api 0.12.2
    - jjwt-impl 0.12.2
    - jjwt-jackson 0.12.2
  - Redis
  - Firebase 9.2.0
  - Jasypt 3.0.5

- JPA
- Spring security 6.3
- Validation
- Vault 3.1.1
- Spring Boot Starter Web
- Spring Boot Devtools
- MySQL
- Spring Boot Starter Test

▼ build.gradle

```
dependencies {
    // OpenAPI (Swagger 3.0) 지원
    implementation 'org.springdoc:springdoc-openapi-s
tarter-webmvc-ui:2.2.0'

    // JWT 지원
    implementation 'io.jsonwebtoken:jjwt-api:0.12.2'
    implementation 'io.jsonwebtoken:jjwt-impl:0.12.2'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.1
2.2'

    // Redis 지원
    implementation 'org.springframework.boot:spring-b
oot-starter-data-redis'

    // Firebase 지원
    implementation 'com.google.firebase:firebase-admi
n:9.2.0'

    // Jasypt (Java Simplified Encryption) 지원
    implementation 'com.github.ulisesbocchio:jasypt-s
pring-boot-starter:3.0.5'

    // JPA 지원
    implementation 'org.springframework.boot:spring-b
```

```
oot-starter-data-jpa'

    // Spring Security 지원
    implementation 'org.springframework.boot:spring-b
oot-starter-security'

    // Validation 지원
    implementation 'org.springframework.boot:spring-b
oot-starter-validation'

    // Vault 지원
    // implementation group: 'org.springframework.vau
lt', name: 'spring-vault-core', version: '3.1.1'

    // Email 문의
    implementation 'org.springframework.boot:spring-b
oot-starter-mail'

    // Spring Web 지원
    implementation 'org.springframework.boot:spring-b
oot-starter-web'

    // Lombok 지원 (컴파일 시)
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    // 개발 환경에서만 사용되는 Spring Boot DevTools
    developmentOnly 'org.springframework.boot:spring-
boot-devtools'

    // MySQL 커넥터
    runtimeOnly 'com.mysql:mysql-connector-j'

    // 테스트를 위한 Spring Boot Starter Test
    testImplementation 'org.springframework.boot:spri
ng-boot-starter-test'

    // Reactor 테스트를 위한 Reactor Test
```

```
    testImplementation 'io.projectreactor:reactor-tes
t'

    // Spring Security 테스트 지원
    testImplementation 'org.springframework.security:
spring-security-test'

}
```

🎨 IDE

- CLion: 2023.3.4
- IntelliJ Ultimate 2023.3.4
- VsCode: 1.86

3️⃣ 설정 파일 및 환경 변수 정보

Business Server

- application.yml

```
spring:
  data:
    mongodb:
      uri: mongodb://localhost:27017/mymongo
```

Web(Auth) Server

- application-prod.yml (배포 버전, jasypt로 암호화 적용)

```
server:
  base-url: k10c209.p.ssafy.io
```

```
spring:
  servlet:
    # file 업로드 관련
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB

  jpa:
    open-in-view: false
    defer-datasource-initialization: true
    generate-ddl: true
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        use_sql_comments: true
        show_sql: true
        jdbc:
          batch_size: 100
        default_batch_fetch_size: 100

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ENC(fkaP7rwbcFvYwi4hnDA0Ym9n3UmIZ8ulrF0VdnEtXth
KqAJ0iPckEfmNAbcBTwGzscSG+C/HbkizZdSp1F1Zim5pvmIu//kHJPA
9Sy8Ev6Pkz4KF0JLgZrT14q5jzvk0+51OGNLXSdV86Ws+uakFB4D/FO/
fbNK0TeolpXh1LmW8ZVf41Bo5SxlGCefC/13ATuAV7X0puARK56Qp2ZB
TZ43nt2OhZk6UAdkdwjDWaiq0UtvskUrnjA==)
    username: ENC(bFqHGm/69wBjk8uekBbBoYgj5V+LS1M2)
    password: ENC(65rsvTX/douB3WmrV4Hh0OblqyK9c+FF)

  # NoSQL 설정
  data:
    # Redis settings
    redis:
      host: ENC(Zp/KVZSQPCR5u2lMfpvwUQKNwkTSUMMISFuav8P6
u8U=)
```

```
      port: ENC(RsQtRjtDoiu6pGZ6OvLC5g==)

# jwt 토큰 설정
jwt:
  accessKey: ENC(s0vsxAuXmlzr6DjkUVLrlIS+MdTML5arOQyKWar
2qbFPkVVJ1zZoR09t83GnhFYhzIvLDaD119HeajudDJS4GXxVTkd+M/z
Q/q1ssX4aw00y+52LOysZk/q/I3T4a3zYmOxfYI4aCslih6D5TTl5rQn
PeZV2rmjdJXAg1X7owfGsquBPnuUptjuKxteIdEjtFFp1SF9QZ/+TvWy
YrlwH1Q==)
  refreshKey: ENC(gE+DTAeZ7HD32Rd6u5HcfJu+bg4kgdULucqZWV
JCbHG5q5SE53YWhNuQcSyE61MVF8Cow5idsgt1RoMi3w0ri7Bq57Fr6+
m6qr5bi+kXL+nQOv/GWoPcj2DxteTB5izXHEp5fzC5uq1m8QrltRiQa1
WfY8s1w6myZFWqPzn4S8FjoR/7/4oG7XU/Q/+HV1I0viF/51ziOKJFVn
0nPAyMnQ==)
  accessExpiration: PT420M
  refreshExpiration: PT10080M

# vault 설정
vault:
  uri: http://localhost:8200
  token: myroot
  keyPath: secret/kickback

# log 관리
logging:
  level:
    org.hibernate:
      type.descriptor.sql: trace
      org.hibernate.SQLQuery: debug

# 기타
app:
  # firebase
  firebase-configuration-file: ENC(WBdcWWmK6e+jZNEI86zoq
JXKVgXcLGUTqEVzaoxgSyc=)
  firebase-bucket: ENC(BZQQDBkTmfMuGavmX4Dt8TMB9Dt6bCb4N
pLprjOr1FvCeMcSuJjqJA==)
```

4 빌드 및 배포

MySQL

```
# 설치 및 설정

apt update
apt install mysql-server
sudo ufw allow 3306 # 방화벽 3306 포트 허용

sudo systemctl start mysql # MySQL 실행

mysql -u root -p # MySQL 접속

# 파일명이 kickback인 database 생성
mysql> CREATE DATABASE kickback CHARACTER SET utf8mb4 COLLA
TE utf8mb4_bin;

mysql> CREATE USER 'username'@'hostname' IDENTIFIED BY 'pas
sword'; # 사용자 생성
# 사용자에게 kickback 데이터베이스에 대한 모든 권한 부여
mysql> GRANT ALL PRIVILEGES ON database_name.* TO 'usernam
e'@'hostname' WITH GRANT OPTION;
```

```
# /etc/mysql/my.cnf 설정 (인코딩 설정)

...

# 인코딩을 utfmb4로 설정합니다.
[mysqld]
character-set-server=utf8mb4
collation-server=utf8mb4_bin

[client]
default-character-set=utf8mb4
```

```
[mysql]
default-character-set=utf8mb4

[mysqldump]
default-character-set=utf8mb4
```

Docker

```
#!/bin/bash

# 기존 도커 패키지 제거 (이전 버전이 설치된 경우)
sudo apt-get remove docker docker-engine docker.io containe
rd runc

# 필수 패키지 설치
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates
curl software-properties-common

# 도커 GPG 키 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | s
udo gpg --dearmor -o /usr/share/keyrings/docker-archive-key
ring.gpg

# 도커 저장소 추가
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-
archive-keyring.gpg] https://download.docker.com/linux/ubun
tu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.l
ist.d/docker.list > /dev/null

# 도커 설치
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.
io

# 도커 컴포즈 최신 버전 다운로드
sudo curl -L "https://github.com/docker/compose/releases/la
```

```
test/download/docker-compose-$(uname -s)-$(uname -m)" -o /u
sr/local/bin/docker-compose

# 실행 권한 부여
sudo chmod +x /usr/local/bin/docker-compose

# 도커 사용자 그룹에 현재 사용자 추가
sudo usermod -aG docker $USER
newgrp docker
sudo service docker restart

# 설치 확인
docker --version
docker-compose --version

# 권한 설정
chmod +x installDocker.sh

# 실행
./installDocker.sh
```

Redis 컨테이너 실행

```
docker pull redis # 이미지 pull
docker run --name redis -d -p 6379:6379 redis # redis 컨테이너
```

Jenkins 컨테이너 실행

```
#!/bin/bash

# Jenkins 폴더 생성
JENKINS_DIR="./jenkins"
if [ ! -d "$JENKINS_DIR" ]; then
    mkdir "$JENKINS_DIR"
fi
```

```
# Docker Compose 실행
docker-compose up -d

# Jenkins 컨테이너가 완전히 실행될 때까지 대기
sudo sleep 60

# Jenkins 폴더로 이동
cd ./jenkins

# Jenkins 폴더가 완전히 생성될 때까지 대기
sudo sleep 60

# update center에 필요한 CA 파일 다운로드
UPDATE_CENTER_DIR="./update-center-rootCAs"
if [ ! -d "$UPDATE_CENTER_DIR" ]; then
    mkdir "$UPDATE_CENTER_DIR"
fi

sudo wget https://cdn.jsdelivr.net/gh/lework/jenkins-update
-center/rootCA/update-center.crt -O "$UPDATE_CENTER_DIR/upd
ate-center.crt"

# Jenkins 설정 파일 수정
sudo sed -i 's#https://updates.jenkins.io/update-center.jso
n#https://raw.githubusercontent.com/lework/jenkins-update-c
enter/master/updates/tencent/update-center.json#' ./hudson.
model.UpdateCenter.xml

# Jenkins 재시작 (필수)
docker restart jenkins

# 현재 폴더 확인
pwd

# /home/ubuntu/develop/CICD 확인
chmod +x ./Install/installJenkins.sh
```

```
# 실행
./Install/installJenkins.sh
```

Nginx 설정

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {}

http {

    include mime.types;

        # 5MB 미만의 데이터 통신 허용
    client_max_body_size 5M;

    server {
        listen 80;
        server_name k10c209.p.ssafy.io;

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

    server {
        # listen [::]:443 ssl ipv6only=on;
        listen 443 ssl;
        server_name k10c209.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/p.ssafy.io/full
        ssl_certificate_key /etc/letsencrypt/live/p.ssafy.io/

        # front-end 설정
        location / {
            root /home/ubuntu/project/front-end/build;
            index index.html;
```

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

        location ~* \.(eot|otf|ttf|woff|woff2)$ {
            add_header Access-Control-Allow-Origin *;
        }

        # back-end api 설정
        location /api/v1 {
            proxy_pass http://k10c209.p.ssafy.io:8080;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_for
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_ssl_server_name on;
        }

        # jenkins 설정
        location /jenkins {
            proxy_pass http://k10c209.p.ssafy.io:8081;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_for
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_ssl_server_name on;
        }


    }
}
```

0. Jenkins Git Clone Pipeline

```
pipeline
{
    agent any
```

```
    stages
    {
            // 깃에서 클론 후 기존 파일 제거 및 이동
        stage('Git Clone')
        {
            steps {
                sh '''
                    rm -rf S10P31C209

                                            git clone -b releas
e --single-branch -n --depth=1 --filter=tree:0 https://bull
ie_hu:hYbzTME7fdFgkE9acjgy@lab.ssafy.com/s10-final/S10P31C2
09.git
                    cd S10P31C209/
                    git sparse-checkout set --no-cone Chatt
ingServer LiveServer WebServer businessServer kickbackweb
                    git checkout

                    chmod -R 777 ./**

                    rm -rf /var/jenkins_home/workspace/fron
t-end/**
                    rm -rf /var/jenkins_home/workspace/back
-end/**
                    rm -rf /var/jenkins_home/workspace/busi
ness-server/**
                    rm -rf /var/jenkins_home/workspace/live
-server/**
                    rm -rf /var/jenkins_home/workspace/chat
ting-server/**

                    mv kickbackweb/** /var/jenkins_home/wor
kspace/front-end
                    mv WebServer/** /var/jenkins_home/works
pace/back-end
                    mv businessServer/** /var/jenkins_home/
workspace/business-server
                    mv LiveServer/** /var/jenkins_home/work
```

```
space/live-server
                    mv ChattingServer/** /var/jenkins_home/
workspace/chatting-server
                '''
            }
        }
        // 프론트엔드 작업 시작
        stage('Front-end') {
            steps {
                build job: 'front-end', parameters: []
            }
        }
        // 백엔드 작업 시작
        stage('back-end') {
            steps {
                build job: 'back-end', parameters: []
            }
        }
        // 비즈니스 서버 작업 시작
        stage('business-server') {
            steps {
                build job: 'business-server', parameters:
[]
            }
        }
        // 라이브 서버 작업 시작
        stage('live-server') {
            steps {
                build job: 'live-server', parameters: []
            }
        }
        // 채팅 서버 작업 시작
        stage('chatting-server') {
            steps {
                build job: 'chatting-server', parameters:
[]
            }
        }
```

```
    }

}
```

1. Front-end

1) Jenkins pipeline

```
pipeline
{
    agent any

    // Node.js 20.10.0 사용
    tools
    {
        nodejs 'nodejs-20.10.0'
    }

    stages
    {
            // 프로젝트 파일 빌드
        stage('Build')
        {
            steps {
                sh '''
                    pwd

                    npm install
                    CI=false npm run build

                    ls
                '''
            }
        }
        // SSH를 통해 EC2로 빌드한 파일 전달
        stage('SSH')
        {
            steps {
```

```
                sshPublisher(
                    publishers: [
                        sshPublisherDesc(
                            configName: 'kickback-ec2',
                            transfers: [
                                sshTransfer(
                                    cleanRemote: false,
                                    excludes: '',
                                    execCommand: '''
                                        pwd
                                    ''',
                                    execTimeout: 120000,
                                    flatten: false,
                                    makeEmptyDirs: false,
                                    noDefaultExcludes: false,
                                    patternSeparator: '[, ]+'
                                    remoteDirectory: 'front-e
                                    remoteDirectorySDF: false
                                    removePrefix: 'build/',
                                    sourceFiles: 'build/**'
                                )
                            ],
                            usePromotionTimestamp: false,
                            useWorkspaceInPromotion: false,
                            verbose: false
                        )
                    ]
                )
            }
        }

    }
}
```

2. Back-end

1) Jenkins pipeline

```
pipeline
{
    agent any

    stages
    {
            // 프로젝트 파일 빌드
        stage('Build')
        {
            steps {
                catchError(buildResult: 'SUCCESS', stageResul
                    sh '''
                        pwd
                        ls -al
                        java --version
                        chmod 711 gradlew
                        ./gradlew clean build
                        ls -al
                        ls -al build/libs
                    '''
                }
            }
        }
        // SSH를 통해 EC2로 빌드한 파일 전달
        stage('SSH')
        {
            steps {
                sshPublisher(
                    publishers: [
                        sshPublisherDesc(
                            configName: 'kickback-ec2',
                            transfers: [
                                sshTransfer(
                                    cleanRemote: false,
                                    excludes: '',
                                    execCommand: '''
```

```
                                        /home/ubuntu/project/
                                    ''',
                                    execTimeout: 120000,
                                    flatten: false,
                                    makeEmptyDirs: false,
                                    noDefaultExcludes: false,
                                    patternSeparator: '[, ]+'
                                    remoteDirectory: 'back-en
                                    remoteDirectorySDF: false
                                    removePrefix: 'build/libs
                                    sourceFiles: 'build/libs/
                                )
                            ],
                            usePromotionTimestamp: false,
                            useWorkspaceInPromotion: false,
                            verbose: false
                        )
                    ]
                )
            }
        }
    }

}
```

2) Dockerfile

```
# 기본 이미지 설정
FROM openjdk:17

# 환경변수 설정
ENV JASYPT_ENCRYPTOR_KEY "ssafy"
# 시간대 설정
ENV TZ Asia/Seoul

# 컨테이너 내부에서 작업할 디렉토리 설정하기
WORKDIR /back-end
```

```
# 호스트 현재 디렉토리 내의 빌드된 파일 옮기기
COPY ./*.jar /back-end

# 컨테이너 시작 후 실행
CMD ["java", "-jar", "authserv-0.0.1-SNAPSHOT.jar"]
```

3) docker_exec.sh

```
#sudo docker stop back-end-container
#sudo docker rm back-end-container

# 기존 실행중인 컨테이너 제거
sudo docker rm --force back-end-container

# 기존 이미지 삭제
sudo docker rmi back-end

# Dockerfile 기반으로 이미지 빌드
sudo docker build -t back-end /home/ubuntu/project/back-end/

# 컨테이너 실행
sudo docker run -d -p 8080:8080 --name back-end-container bac
```

3. Business server

1) Jenkins pipeline

```
pipeline
{
    agent any

    stages
    {
            // 프로젝트 파일 빌드
        stage('Build')
        {
```

```
            steps {
                sh '''
                    pwd
                    ls -al
                    java --version
                    chmod 711 gradlew
                    ./gradlew clean build
                    ls -al
                    ls -al build/libs
                '''
            }
        }
        // SSH를 통해 EC2로 빌드한 파일 전달
        stage('SSH')
        {
            steps {
                sshPublisher(
                    publishers: [
                        sshPublisherDesc(
                            configName: 'kickback-ec2',
                            transfers: [
                                sshTransfer(
                                    cleanRemote: false,
                                    excludes: '',
                                    execCommand: '''
                                        /home/ubuntu/projec
t/business-server/docker_exec.sh
                                    ''',
                                    execTimeout: 120000,
                                    flatten: false,
                                    makeEmptyDirs: false,
                                    noDefaultExcludes: fals
e,
                                    patternSeparator: '[, ]
+',
                                    remoteDirectory: 'busin
ess-server/',
                                    remoteDirectorySDF: fal
```

```
se,
                                        removePrefix: 'build/li
bs/',
                                        sourceFiles: 'build/lib
s/*.jar'
                                    )
                                ],
                                usePromotionTimestamp: false,
                                useWorkspaceInPromotion: false,
                                verbose: false
                            )
                        ]
                    )
                }
            }
        }

}
```

2) Dockerfile

```
# 기본 이미지 설정
FROM openjdk:17

# 시간대 설정
ENV TZ Asia/Seoul

# 컨테이너 내부에서 작업할 디렉토리 설정
WORKDIR /business-server

# 호스트 현재 디렉토리 내의 빌드된 파일 옮기기
COPY ./*.jar /business-server

# 컨테이너 시작 후 실행
CMD ["java", "-jar", "businessServer-0.0.1-SNAPSHOT.jar"]
```

3) docker_exec.sh

```
#sudo docker stop business-server-container
#sudo docker rm business-server-container

# 기존 실행중인 컨테이너 제거
sudo docker rm --force business-server-container

# 기존 이미지 삭제
sudo docker rmi business-server

# Dockerfile 기반으로 이미지 빌드
sudo docker build -t business-server /home/ubuntu/project/bus

# 컨테이너 실행
sudo docker run -d -p 1370:1370 --name business-server-contai
```

4. Live server

1) Jenkins pipeline

```
pipeline
{
    agent any

    stages
    {
            // SSH를 통해 EC2로 빌드한 파일 전달
        stage('SSH')
        {
            steps {
                sshPublisher(
                    publishers: [
                        sshPublisherDesc(
                            configName: 'kickback-ec2',
                            transfers: [
                                sshTransfer(
                                    cleanRemote: false,
```

```
                                    excludes: '',
                                    execCommand: '''
                                        /home/ubuntu/project/
                                        /home/ubuntu/project/
                                    ''',
                                    execTimeout: 120000,
                                    flatten: false,
                                    makeEmptyDirs: false,
                                    noDefaultExcludes: false,
                                    patternSeparator: '[, ]+'
                                    remoteDirectory: 'live-se
                                    remoteDirectorySDF: false
                                    removePrefix: '',
                                    sourceFiles: '**'
                                )
                            ],
                            usePromotionTimestamp: false,
                            useWorkspaceInPromotion: false,
                            verbose: false
                        )
                    ]
                )
            }
        }
    }

}
```

2) Dockerfile

```
# 기본 이미지 설정
FROM ubuntu:20.04

# 시간대 설정
ENV TZ Asia/Seoul

# 컨테이너 내부에서 작업할 디렉토리 설정
WORKDIR /live-server
```

```
# 호스트 현재 디렉토리 내의 빌드된 파일 옮기기
COPY ./files/build/LiveServer /live-server

# 컨테이너 시작 후 실행
CMD ["./LiveServer"]
```

3) build_exec.sh

```
# 기존 빌드 파일 삭제
rm -rf /home/ubuntu/project/live-server/files/build/*

# 빌드 디렉토리 설정
BUILD_DIR="/home/ubuntu/project/live-server/files/build"

# 빌드 디렉토리로 이동
cd ${BUILD_DIR}

# cmake 구성, 빌드
cmake /home/ubuntu/project/live-server/files
make

# 빌드 완료 메시지
echo "Build completed successfully"
```

4) docker_exec.sh

```
#sudo docker stop live-server-container
#sudo docker rm live-server-container

# 기존 실행중인 컨테이너 제거
sudo docker rm --force live-server-container

# 기존 이미지 삭제
sudo docker rmi live-server

# Dockerfile 기반으로 이미지 빌드
```

```
sudo docker build -t live-server /home/ubuntu/project/live-se

# 컨테이너 실행
sudo docker run -d -p 5058:5058/udp --name live-server-contai
```

5. Chatting server

1) Jenkins pipeline

```
pipeline
{
    agent any

    stages
    {
            // SSH를 통해 EC2로 빌드한 파일 전달
        stage('SSH')
        {
            steps {
                sshPublisher(
                    publishers: [
                        sshPublisherDesc(
                            configName: 'kickback-ec2',
                            transfers: [
                                sshTransfer(
                                    cleanRemote: false,
                                    excludes: '',
                                    execCommand: '''
                                        /home/ubuntu/project/
                                        /home/ubuntu/project/
                                    ''',
                                    execTimeout: 120000,
                                    flatten: false,
                                    makeEmptyDirs: false,
                                    noDefaultExcludes: false,
                                    patternSeparator: '[, ]+'
```

```
                                    remoteDirectory: 'chattin
                                    remoteDirectorySDF: false
                                    removePrefix: '',
                                    sourceFiles: '**'
                                )
                            ],
                            usePromotionTimestamp: false,
                            useWorkspaceInPromotion: false,
                            verbose: false
                        )
                    ]
                )
            }
        }
    }

}
```

2) Dockerfile

```
# 기본 이미지 설정
FROM ubuntu:20.04

# 시간대 설정
ENV TZ Asia/Seoul

# 컨테이너 내부에서 작업할 디렉토리 설정
WORKDIR /chatting-server

# 호스트 현재 디렉토리 내의 빌드된 파일 옮기기
COPY ./files/build/ChattingServer /chatting-server

# 컨테이너 시작 후 실행
CMD ["./ChattingServer"]
```

3) build_exec.sh

```
# 기존 빌드 파일 삭제
rm -rf /home/ubuntu/project/chatting-server/files/build/*

# 빌드 디렉토리 설정
BUILD_DIR="/home/ubuntu/project/chatting-server/files/build"

# 빌드 디렉토리로 이동
cd ${BUILD_DIR}

# cmake 구성, 빌드
cmake /home/ubuntu/project/chatting-server/files
make

# 빌드 완료 메시지
echo "Build completed successfully"
```

4) docker_exec.sh

```
#sudo docker stop chatting-server-container
#sudo docker rm chatting-server-container

# 기존 실행중인 컨테이너 제거
sudo docker rm --force chatting-server-container

# 기존 이미지 삭제
sudo docker rmi chatting-server

# Dockerfile 기반으로 이미지 빌드
sudo docker build -t chatting-server /home/ubuntu/project/cha

# 컨테이너 실행
sudo docker run -d -p 1371:1371 --name chatting-server-contai
```

4️⃣ DB Dump

▼ kickback_member.sql (회원 데이터)

```
-- MySQL dump 10.13  Distrib 8.0.34, for Win64 (x86_64)
--
-- Host: k10c209.p.ssafy.io    Database: kickback
-- ------------------------------------------------------
-- Server version   8.0.36-0ubuntu0.20.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLI
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RE
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECT
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CH
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_V
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `member`
--

DROP TABLE IF EXISTS `member`;
/*!40101 SET @saved_cs_client     = @@character_set_client
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `member` (
  `id` binary(16) NOT NULL,
  `role` enum('USER','ADMIN') COLLATE utf8mb4_bin NOT NULL
  `email` varchar(255) COLLATE utf8mb4_bin NOT NULL,
  `nickname` varchar(255) COLLATE utf8mb4_bin NOT NULL,
  `password` varchar(255) COLLATE utf8mb4_bin NOT NULL,
  `profile_image` varchar(255) COLLATE utf8mb4_bin DEFAULT
  `current_token` varchar(512) COLLATE utf8mb4_bin DEFAULT
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`),
  UNIQUE KEY `nickname` (`nickname`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bi
```

```
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIEN
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESU
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTIO
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-05-18 13:04:57
```

▼ kickback_blocked_access_token.sql (중복 로그인 방지를 위한 만료된 access 토큰 데이터)

```
-- MySQL dump 10.13  Distrib 8.0.34, for Win64 (x86_64)
--
-- Host: k10c209.p.ssafy.io    Database: kickback
-- ------------------------------------------------------
-- Server version   8.0.36-0ubuntu0.20.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLI
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RE
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECT
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CH
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_V
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `blocked_access_token`
--

DROP TABLE IF EXISTS `blocked_access_token`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `blocked_access_token` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `email` varchar(255) COLLATE utf8mb4_bin NOT NULL,
  `token` varchar(512) COLLATE utf8mb4_bin NOT NULL,
  `expiration_time` bigint NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=157 DEFAULT CHARSET=utf8mb4
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIEN
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESU
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTIO
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-05-18 13:04:57
```

▼ kickback_soccer_record.sql (축구 모드 기록 데이터)

```
-- MySQL dump 10.13  Distrib 8.0.34, for Win64 (x86_64)
--
-- Host: k10c209.p.ssafy.io    Database: kickback
-- ------------------------------------------------------
-- Server version   8.0.36-0ubuntu0.20.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLI
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RE
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECT
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CH
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
```

```
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_V
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `soccer_record`
--

DROP TABLE IF EXISTS `soccer_record`;
/*!40101 SET @saved_cs_client     = @@character_set_client
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `soccer_record` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `draws` int NOT NULL,
  `gd` int NOT NULL,
  `loses` int NOT NULL,
  `scores` int NOT NULL,
  `wins` int NOT NULL,
  `member_id` binary(16) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UK_eitmynjrqd1lvdwggmppitw9j` (`member_id`),
  CONSTRAINT `FKqpt7q6ey8p2ddi0jvkvhhg71x` FOREIGN KEY (`m
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=utf8mb4
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIEN
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESU
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTIO
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-05-18 13:04:59
```

▼ kickback_speed_record.sql (스피드 모드 기록 데이터)

```
-- MySQL dump 10.13  Distrib 8.0.34, for Win64 (x86_64)
--
-- Host: k10c209.p.ssafy.io    Database: kickback
-- ------------------------------------------------------
-- Server version   8.0.36-0ubuntu0.20.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLI
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RE
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECT
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CH
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_V
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `speed_record`
--

DROP TABLE IF EXISTS `speed_record`;
/*!40101 SET @saved_cs_client     = @@character_set_client
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `speed_record` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `member_id` binary(16) NOT NULL,
  `map` int NOT NULL,
  `millis` int unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_speed_record_member` (`member_id`),
  CONSTRAINT `fk_speed_record_member` FOREIGN KEY (`member_
) ENGINE=InnoDB AUTO_INCREMENT=27 DEFAULT CHARSET=utf8mb4 
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */
```

```
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIEN
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESU
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTIO
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-05-18 13:04:58
```

▼ kickback_board.sql (게시판 데이터)

```
-- MySQL dump 10.13  Distrib 8.0.34, for Win64 (x86_64)
--
-- Host: k10c209.p.ssafy.io    Database: kickback
-- ------------------------------------------------------
-- Server version   8.0.36-0ubuntu0.20.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLI
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RE
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECT
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CH
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_V
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `board`
--

DROP TABLE IF EXISTS `board`;
/*!40101 SET @saved_cs_client     = @@character_set_client
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `board` (
  `id` int NOT NULL AUTO_INCREMENT,
  `title` varchar(255) COLLATE utf8mb4_bin NOT NULL,
  `member_id` binary(16) DEFAULT NULL,
```

```
  `content` varchar(255) COLLATE utf8mb4_bin NOT NULL,
  `category` enum('FREE','SHARE','QNA') COLLATE utf8mb4_bi
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON
  `created_date` datetime(6) DEFAULT NULL,
  `updated_date` datetime(6) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_board_member` (`member_id`),
  CONSTRAINT `fk_board_member` FOREIGN KEY (`member_id`) R
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8mb4 
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIEN
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESU
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTIO
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-05-18 13:04:56
```

▼ kickback_comment.sql (공지사항 데이터)

```
-- MySQL dump 10.13  Distrib 8.0.34, for Win64 (x86_64)
--
-- Host: k10c209.p.ssafy.io    Database: kickback
-- ------------------------------------------------------
-- Server version   8.0.36-0ubuntu0.20.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLI
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RE
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECT
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CH
```

```
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_V
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `comment`
--

DROP TABLE IF EXISTS `comment`;
/*!40101 SET @saved_cs_client     = @@character_set_client
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `comment` (
  `id` int NOT NULL AUTO_INCREMENT,
  `member_id` binary(16) DEFAULT NULL,
  `board_id` int DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON
  `content` text COLLATE utf8mb4_bin NOT NULL,
  `created_time` datetime(6) DEFAULT NULL,
  `updated_time` datetime(6) DEFAULT NULL,
  `comment_content` varchar(255) COLLATE utf8mb4_bin NOT N
  PRIMARY KEY (`id`),
  KEY `fk_comment_member` (`member_id`),
  KEY `fk_comment_board` (`board_id`),
  CONSTRAINT `fk_comment_board` FOREIGN KEY (`board_id`) R
  CONSTRAINT `fk_comment_member` FOREIGN KEY (`member_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 C
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIEN
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESU
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTIO
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

```
-- Dump completed on 2024-05-18 13:04:58
```

▼ kickback_notice.sql (게시판 댓글 데이터)

```
-- MySQL dump 10.13  Distrib 8.0.34, for Win64 (x86_64)
--
-- Host: k10c209.p.ssafy.io    Database: kickback
-- ------------------------------------------------------
-- Server version   8.0.36-0ubuntu0.20.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLI
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RE
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECT
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CH
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_V
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `notice`
--

DROP TABLE IF EXISTS `notice`;
/*!40101 SET @saved_cs_client     = @@character_set_client
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `notice` (
  `id` int NOT NULL AUTO_INCREMENT,
  `content` mediumtext COLLATE utf8mb4_bin NOT NULL,
  `category` enum('NOTICE','UPDATE') COLLATE utf8mb4_bin N
  `member_id` binary(16) NOT NULL,
  `title` varchar(255) COLLATE utf8mb4_bin NOT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON
  PRIMARY KEY (`id`),
```

```
  KEY `fk_notice_member` (`member_id`),
  CONSTRAINT `fk_notice_member` FOREIGN KEY (`member_id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 C
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIEN
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESU
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTIO
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-05-18 13:04:57
```

DB 덤프 및 파일 로드

※ 주의: utf8mb4로 인코딩하여 데이터 베이스에 로드합니다.

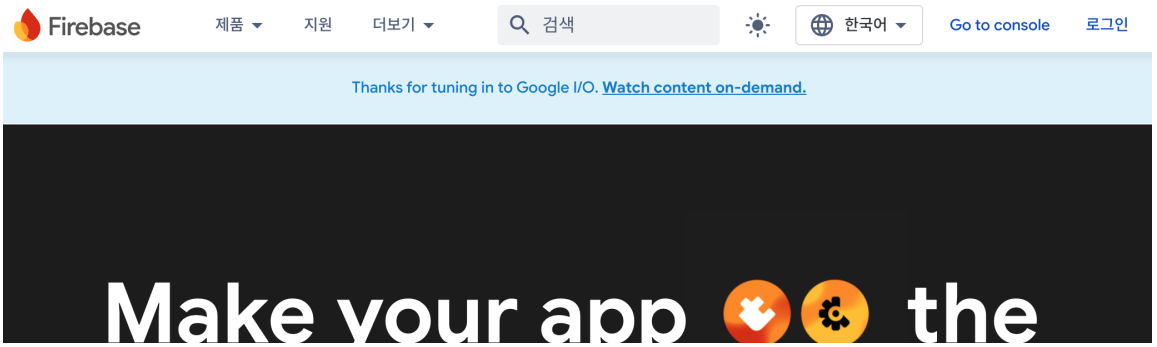
```
# DB dump
mysqldump --default-character-set=utf8mb4 -u kickback -p<비
밀번호> kickback > dump.sql

# DB 복구(store)
mysql --default-character-set=utf8mb4 -u kickback -p<비밀번호
> kickback < dump.sql
```

5️⃣ 외부 서비스 활용 가이드

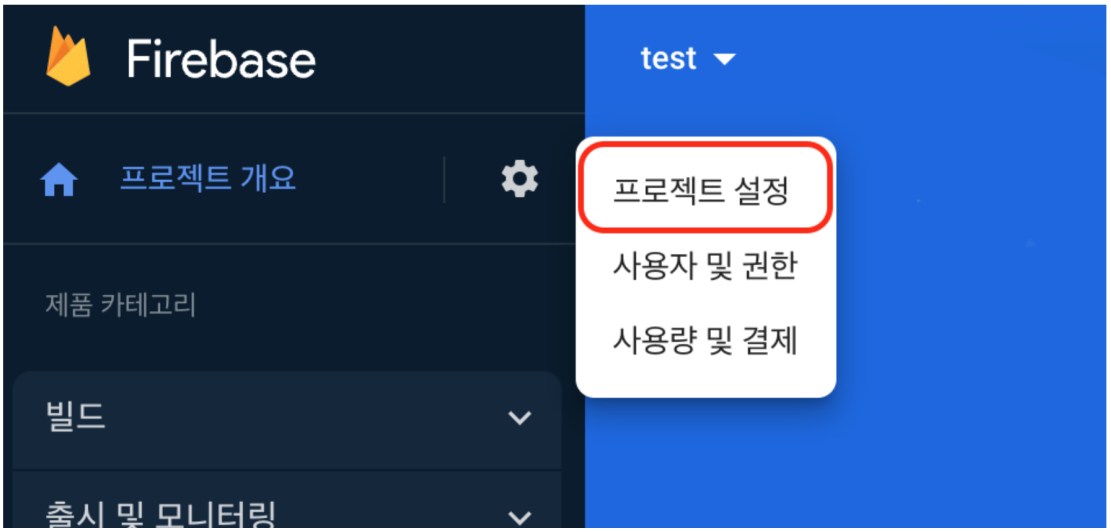
Firebase

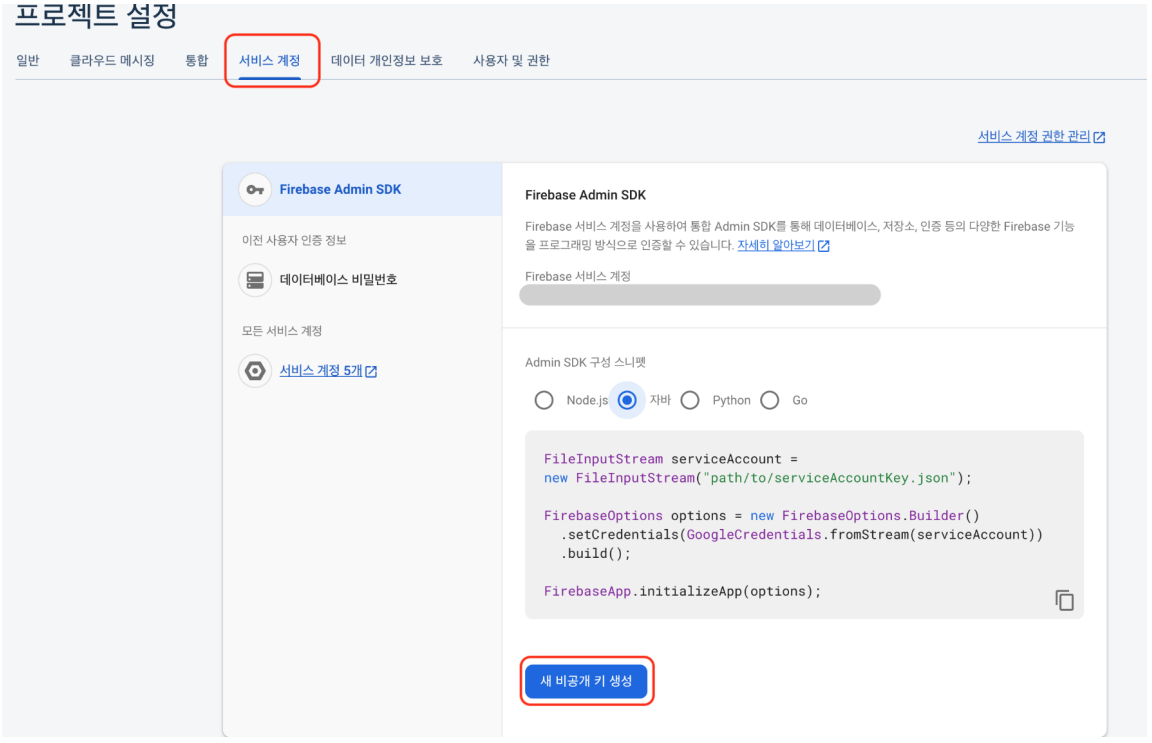
1. https://firebase.google.com/ 접속 및 로그인합니다.



2. 콘솔로 이동 후 프로젝트 추가합니다.

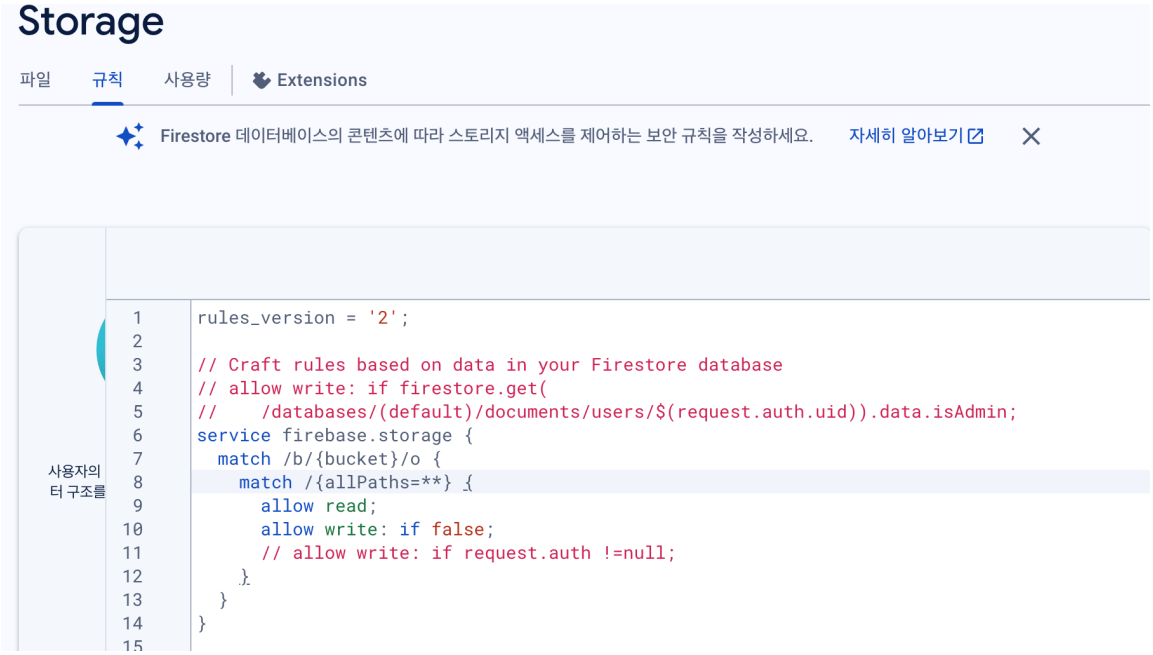
3. 비공개 키를 생성합니다.





이후 생성된 .json 파일의 비공개 키를 Spring-boot 프로젝트의 resources 디렉토리 경로에 넣습니다.

3. 스토리지의 규칙을 적절히 작성합니다.



4. CORS 설정 (gsutils 설치 필요)

   설치 링크. https://cloud.google.com/storage/docs/gsutil_install?hl=ko

   CORS (Cross-Origin Resource Sharing) 오류는 서버(이 경우 Firebase Storage)가 클라이언트(여기서는 `http://localhost:5174`에서 실행 중인 애플리케이션)로부터의 요청을 수락하지 않도록 구성되어 있을 때 발생합니다. 이 문제를 해결하려면 Firebase Storage의 CORS 설정을 조정해야 합니다.

   CORS 설정 방법

   Firebase Storage에서 CORS 설정을 조정하려면 Google Cloud의 `gsutil` 명령어 도구를 사용해야 합니다. 이 도구는 Google Cloud Storage의 리소스를 관리하는 데 사용됩니다. 아래는 CORS 설정 절차를 설명합니다.

   1. **Google Cloud SDK 설치**: 아직 설치하지 않았다면, Google Cloud SDK를 설치해야 합니다. 설치 후, `gsutil` 도구를 사용할 수 있습니다.
   2. **CORS 구성 파일 생성**:
      다음의 내용을 포함하는 JSON 파일(
      `cors.json` 등)을 생성합니다. 이 예제는 모든 출처( )에서 모든 GET 요청을 허용하도록 설정합니다.

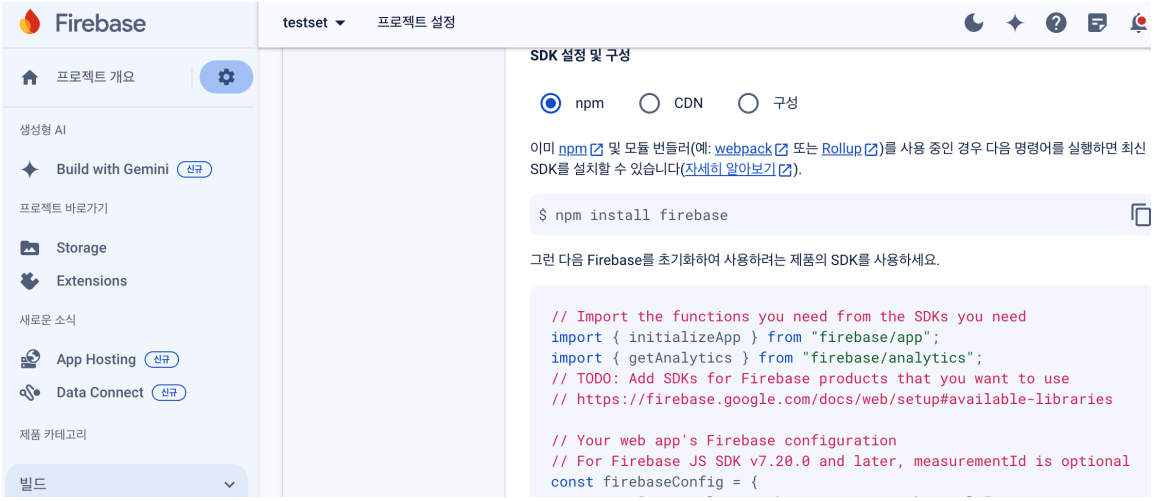
```
    jsonCopy code
[
  {
    "origin": ["*"],
    "method": ["GET"],
    "maxAgeSeconds": 3600
  }
]
```

3. CORS 설정 적용:

   `gsutil` 명령어를 사용하여 CORS 설정을 적용합니다. 아래 명령어에서 `[YOUR_BUCKET_NAME]`을 실제 버킷 이름으로 대체하세요.

```
bashCopy code
gsutil cors set cors.json gs://[YOUR_BUCKET_NAME]
```

4. 앱 등록 후 React 는 npm sdk 설정 및 구성을 firebase.js 파일로 저장하여 활용합니다.



Google Cloud Storage를 사용한 파일 호스팅 설정 방법

1. Google Cloud 프로젝트 생성 및 설정

1. **Google Cloud Console** (https://console.cloud.google.com/)에 접속합니다.
2. 새 프로젝트를 생성하거나 기존 프로젝트를 선택합니다.
3. **네비게이션 메뉴**에서 **"Storage"** > **"Browser"**를 선택하여 Google Cloud Storage 섹션으로 이동합니다.

2. 버킷 생성

1. **"버킷 만들기"** 버튼을 클릭합니다.
2. 버킷 이름을 지정하고, (전 세계에서 고유해야 함) 지리적 위치를 선택합니다. 위치 선택은 데이터의 접근 속도와 비용에 영향을 미칩니다.
3. 스토리지 클래스를 선택합니다 (예: Standard, Nearline, Coldline, Archive). 정적 파일 호스팅의 경우 **"Standard"**가 적합합니다.
4. 접근 권한 설정에서 **"모든 사용자에게 공개"**를 선택하여 파일이 인터넷에서 공개적으로 접근 가능하도록 설정합니다.
5. 설정을 확인하고 버킷을 생성합니다.

3. 파일 업로드

1. 생성된 버킷으로 이동합니다.
2. **"파일 업로드"** 버튼을 클릭하여 호스팅할 APK 파일 또는 다른 정적 파일을 업로드합니다.
3. 파일이 버킷에 성공적으로 업로드되면, 파일 이름 옆에 있는 **"공개 액세스"** 옵션을 통해 파일을 공개적으로 설정합니다.

4. 파일 URL 접근

1. 파일을 클릭하여 상세 페이지로 이동합니다.
2. 오른쪽의 세부 정보 패널에서 **"공개 URL"**을 복사합니다. 이 URL을 통해 누구나 인터넷에서 파일에 접근할 수 있습니다.

5. CORS 설정 (필요한 경우)

웹 애플리케이션에서 Google Cloud Storage의 리소스를 사용하려면, CORS 설정이 필요할 수 있습니다. `gsutil` 명령어 도구를 사용하여 CORS 설정을 구성할 수 있습니다:

1. 로컬 컴퓨터에서 CORS 설정 파일(`cors.json`)을 생성합니다:

```
[
  {
```

```
    "origin": ["*"],
    "responseHeader": ["Content-Type"],
    "method": ["GET", "HEAD", "DELETE"], // 허용할 메서
드 타입 설정
    "maxAgeSeconds": 3600
  }
]
```

2. 다음 `gsutil` 명령어로 CORS 설정을 적용합니다:

```
코드 복사
gsutil cors set cors.json gs://your-bucket-name
```

6. **ACL(Access Control List) 변경**

1. 권한 확인

- **IAM 권한 확인**: Google Cloud Console에서 현재 계정의 IAM 권한을 확인하세요. 현재 계정이 버킷에 대해 `Storage Admin` 또는 `Storage Object Admin` 역할을 가지고 있는지 확인합니다. 이 역할들은 버킷 및 객체에 대한 관리 권한을 제공합니다.
  - Google Cloud Console에 로그인합니다.
  - 좌측 메뉴에서 "IAM & Admin" > "IAM"을 선택합니다.
  - 권한 목록에서 현재 사용자의 권한을 확인하고 필요한 경우 적절한 역할을 추가합니다.

2. 버킷 설정 점검

- **Uniform Bucket-Level Access 점검**: 버킷이 Uniform Bucket-Level Access(UBLA)를 사용 중인지 확인합니다. UBLA가 활성화된 경우, 개별 객체에 대한 ACL을 변경할 수 없습니다.
  - Google Cloud Console에서 해당 버킷의 설정을 점검합니다.
  - "Storage" > "Browser"로 이동하여 버킷을 선택합니다.
  - "Permissions" 탭을 확인하여 UBLA 설정을 확인합니다. UBLA가 활성화되어 있다면, 해당 설정을 비활성화하거나 객체 공개를 위한 다른 방법을 사용해야 합니다.

3. ACL 대신 IAM 권한 사용

- UBLA가 활성화되어 있고 비활성화할 수 없는 경우, 대신 IAM 정책을 사용하여 객체를 공개할 수 있습니다. `gsutil iam` 명령을 사용하여 공개 접근 권한을 설정할 수 있습니다:

  ```
  gsutil iam ch allUsers:objectViewer gs://kickback
  ```

  이 명령은 버킷의 모든 객체에 대해 읽기 권한을 부여합니다(`objectViewer`). 특정 객체에 대한 권한을 설정하려면, IAM 대신 객체의 공개 링크를 생성할 수 있습니다.