포팅 메뉴얼

1. 빌드 및 배포

1.1 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전

- JVM: OpenJDK 17
- 웹서버: Nginx 1.24
- WAS: Spring Boot 3.3.6 (Embedded Tomcat 사용)
- IDE: IntelliJ IDEA 2023.3 / VS Code
- Node.js 버전: 18.17.0
- React 버전: 18.2.0

1.2 빌드 시 사용되는 환경 변수 등의 내용 상세 기재

1.2.1

`altteul-fe/.env` (fe 컨테이너 이미지 생성 용도)

```
# API 설정
VITE_API_BASE_PATH=/api/ # NGINX에서 서버로 포워딩

# 소켓 설정
VITE_SOCKET_URL_PROD=/ws # NGINX에서 서버로 포워딩
# 코드 공동 편집 설정
VITE_SIGNALING_URL_PROD=ssafy 제공 ec2 nodejs 서버 URL
```

1.2.2.

`altteul-be/.env` (be 컨테이너 이미지 생성 용도)

```
#서버
SECRET_KEY=
PROD_SERVER_PORT=8080
PROD_SERVER_URL=http://backend:8080
```

```
JWT_ACCESS_VALID_TIME=25000000
JWT_REFRESH_VALID_TIME=50000000

SPRING_PROFILES_ACTIVE=prod

#저지
PROD_JUDGE_SERVER_URL=http://judge:8080
JUDGE_KEY=서버키를 sha256 암호화한 키

#mysql
PROD_MYSQL_HOST=mysql
MYSQL_BINDING_PORT=13306
MYSQL_PORT=3306
MYSQL_USERNAME=
MYSQL_PASSWORD=
MYSQL_ROOT_PASSWORD=
MYSQL_DATABASE=altteul
MYSQL_INIT_VOLUME=./resources/mysql/init
MYSQL_VOLUME=./resources/mysql/data

#redis
PROD_REDIS_HOST=redis
REDIS_BINDING_PORT=6389
REDIS_PORT=6379
REDIS_PASSWORD=
REDIS_DATA_PATH=./resources/redis/data
REDIS_DEFAULT_CONFIG_FILE=./resources/redis/redis.conf

#소셜 로그인
GITHUB_CLIENT_ID=
GITHUB_CLIENT_SECRET=

# openvidu
OPENVIDU_SECRET=altteul
OPENVIDU_PROD_CORS=https://frontend:5173
PROD_IP=https://openvidu-server:4443
OPENVIDU_RECORDING=false
```

```
# open api key
OPEN_AI_SECRET_KEY=

# AWS
AWS_BUCKET_NAME=altteul-792301
AWS_ACCESS_KEY=
AWS_SECRET_KEY=
AWS_URL=
```

1.2.3

`/.env` (docker-compose MYSQL, REDIS, JUDGE 컨테이너 적용 용도)

```
#서버
SECRET_KEY=
PROD_SERVER_PORT=8080
PROD_SERVER_URL=http://backend:8080
JWT_ACCESS_VALID_TIME=25000000
JWT_REFRESH_VALID_TIME=50000000

SPRING_PROFILES_ACTIVE=dev

#저지
PROD_JUDGE_SERVER_URL=http://judge:8080
JUDGE_KEY=서버키를 sha256 암호화한 키

#mysql
PROD_MYSQL_HOST=mysql
MYSQL_BINDING_PORT=13306
MYSQL_PORT=3306
MYSQL_USERNAME=
MYSQL_PASSWORD=
MYSQL_ROOT_PASSWORD=
MYSQL_DATABASE=altteul
MYSQL_INIT_VOLUME=./resources/mysql/init
MYSQL_VOLUME=./resources/mysql/data
```

```
#redis
PROD_REDIS_HOST=redis
REDIS_BINDING_PORT=6389
REDIS_PORT=6379
REDIS_PASSWORD=1q2w3e
REDIS_DATA_PATH=./resources/redis/data
REDIS_DEFAULT_CONFIG_FILE=./resources/redis/redis.conf
```

1.3 배포 시 특이사항 기재

1. docker, docker compose 설치 필요
2. SpringBoot(BE) + React(FE) + 채점 서버 + Redis DB + MySQL : 컨테이너 빌드 후 docker compose와 환경변수 주입을 통해 실행이 가능합니다.

   ```
   docker compose up
   ```

3. altteul-ie-frontend 컨테이너는 NginX + React 서버로 구성되어 있습니다.
   HTTPS 인증을 받기 위해 도메인이 있는 서버를 활용해야 하며 인증서를 발급한 후 연결이 원활하게 진행될 수 있습니다.
4. 동시편집을 위한 웹소켓 서버는 resources/y-socket폴더의 js파일들을 활용합니다. `https://github.com/yjs/y-websocket.git` 레포지토리를 clone한 후 bin/server.cjs 파일을 변경해줍니다. 또한 이 node 서버를 실행시키기 위해 wss 연결을 위해서는 인증서가 필요합니다. letsencrypt로 쉽게 가능합니다. 포트는 1234입니다. 포트를 열어주세요.
5. 웹RTC 통신을 위한 서버는 오픈비두 실행문서를 참고하여 설치합니다. 그 후 .env 설정 파일에서 443 포트만 8443 포트로 변경해줍니다.
6. 저지 서버의 경우 테스트케이스 정보가 필요합니다. 직접 준비한 테스트 케이스를 `/resources/judge/data/backend/test_case/` 의 경로에 {문제명}/1.in (input 정의), {문제명}/1.out (output 정의), {문제명}/1.info (메타데이터) 를 작성해둡니다.

7. info 파일

```
{
    "test_case_number": 1,
    "spj": false,
    "test_cases": {
        "1": {
            "stripped_output_md5": "ed076287532e86365e841e92bfc50d8c",
            "output_size": 2,
            "output_md5": "ed076287532e86365e841e92bfc50d8c",
            "input_name": "1.in",
            "input_size": 0,
            "output_name": "1.out"
        }
    }
}
```

8. redis.conf를 resources/redis 위치에 위치시킵니다.

1.4 DB 접속 정보 등 프로젝트(ERD)에서 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

- `/altteul-be/src/main/resources/application-prod-*.yml`

- `/docker-compose.yml`

```
services:
  frontend:
    container_name: altteul-ie-frontend
    build:
      context: ./altteul-fe
      dockerfile: Dockerfile.prod
    ports:
      - "7080:7080"
      - "443:443"
```

```
    environment:
      - NODE_ENV=production
    volumes:
      - /etc/letsencrypt:/etc/letsencrypt
    depends_on:
      - backend
    networks: # 이 부분 추가
      - altteul-ie-network  # backend와 같은 네트워크 사용

  backend:
    container_name: altteul-ie-backend
    restart: always
    build:
      context: ./altteul-be
      dockerfile: dockerfile.be
    working_dir: /app
    ports:
      - "8081:8080"
    volumes:
      - gradle_cache:/home/gradle/.gradle
    environment:
      - SPRING_PROFILES_ACTIVE=prod
    command: java -jar app.jar
    networks: #사용할 네트워크 지정
      - altteul-ie-network
    depends_on:
      - mysql
      - mongodb
      - redis
    env_file:
      - ./altteul-be/.env

  mysql:
    env_file:
      - ./altteul-be/.env
    container_name: altteul-ie-mysql
    image: mysql/mysql-server:8.0.27
    environment:
```

```
    - MYSQL_DATABASE=${MYSQL_DATABASE}
    - MYSQL_USER=${MYSQL_USERNAME}
    - MYSQL_PASSWORD=${MYSQL_PASSWORD}
    - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
    - TZ=Asia/Seoul
  command: [ "--character-set-server=utf8mb4", "--collation-server=utf8mb
  ports:
    - "13306:3306"
  volumes: #볼륨 지정
    - ./${MYSQL_VOLUME}:/var/lib/mysql
  networks: #사용할 네트워크 지정
    - altteul-ie-network

mongodb:
  image: mongo:latest
  container_name: altteul-ie-mongodb
  ports:
    - "27017:27017"
  volumes:
    - mongodb_dev_data:/data/db
  environment:
    - MONGO_INITDB_DATABASE=altteul

redis:
  env_file:
    - ./altteul-be/.env
  container_name: altteul-ie-redis
  image: redis:6.2.6-alpine
  ports: # 바인딩할 포트:내부 포트
    - ${REDIS_BINDING_PORT}:${REDIS_PORT}
  command: redis-server /usr/local/etc/redis/redis.conf
  volumes: # 마운트할 볼륨 설정
    - ${REDIS_DATA_PATH}:/data
    - ./${REDIS_DEFAULT_CONFIG_FILE}:/usr/local/etc/redis/redis.conf
  restart: always
  networks: #사용할 네트워크 지정
    - altteul-ie-network
```

```
  judge:
    image: registry.cn-hongkong.aliyuncs.com/oj-image/judge:1.6.1
    container_name: altteul-ie-judge
    restart: always
    ports:
      - "9999:8080"
    cap_drop:
      - SETPCAP
      - MKNOD
      - NET_BIND_SERVICE
      - SYS_CHROOT
      - SETFCAP
      - FSETID
    tmpfs:
      - /tmp
    volumes:
      - ${PROD_TESTCASE_ROOT}:/test_case:ro
      - ./resources/judge/data/judge_server/log:/log
      - ./resources/judge/data/judge_server/run:/judger
    environment:
      - SERVICE_URL=${PROD_JUDGE_SERVER_URL}
      - BACKEND_URL=${PROD_SERVER_URL}/api/judge/check
      - TOKEN=${SECRET_KEY}
    networks: #사용할 네트워크 지정
      - altteul-ie-network

volumes:
  mysql_dev_data:
  mongodb_dev_data:
  redis_dev_data:
  gradle_cache:

networks:
  altteul-ie-network:
    driver: bridge
```

-

/altteul-fe/Dockerfile.prod

```
# Production Dockerfile
FROM node:20-alpine AS builder

WORKDIR /app

# Install yarn and enable corepack
RUN corepack enable

# Copy configuration files first
COPY package.json yarn.lock .yarnrc.yml ./

# Clean install dependencies
RUN yarn install

# Copy the rest of the source code
COPY . .

# Build the application
RUN yarn build

# Production stage
FROM nginx:alpine

RUN rm /etc/nginx/conf.d/default.conf

COPY ./nginx/nginx.prod.conf /etc/nginx/conf.d/nginx.conf

# Copy built assets from builder stage
COPY --from=builder /app/dist /usr/share/nginx/html

# Expose port 7080
EXPOSE 7080

# Start nginx
CMD ["nginx", "-g", "daemon off;"]
```

-

/altteul-fe/nginx/nginx.prod.conf

```
server {
    listen 7080;
    server_name i12c203.p.ssafy.io;
    return 301 https://$host$request_uri; # HTTP 요청을 HTTPS로 리다이렉트
}

server {
    listen 443 ssl;
    server_name i12c203.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i12c203.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i12c203.p.ssafy.io/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    client_max_body_size 1024M;

    location / {
        root /usr/share/nginx/html;
        try_files $uri $uri/ /index.html;
        proxy_redirect     off;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header Host $http_host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api {
        proxy_pass http://backend:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
```

```
        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;
    }

    location /ws {
        proxy_pass http://backend: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 Origin "";
    }

    location /sockjs-node {
        proxy_pass https://backend:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_read_timeout 60s;
    }
}


- /altteul-be/Dockerfile.be
```

```
# 그레들 버전, JDK 버전에 따라 다르게 수정
FROM gradle:8.1.0-jdk17-alpine AS builder
WORKDIR /build

# 그레들 파일이 변경되었을 때만 새롭게 의존 패키지 다운로드 받게 함.
COPY build.gradle settings.gradle /build/
RUN gradle dependencies --refresh-dependencies -x test
```

```
# 빌더 이미지에서 애플리케이션 빌드
COPY . /build
RUN chmod +x ./gradlew
RUN gradle clean build -x test --parallel

# APP
FROM openjdk:17-slim
WORKDIR /app

# 타임존 설정
ENV TZ=Asia/Seoul
RUN apt-get update \
    && apt-get install -y tzdata \
    && ln -snf /usr/share/zoneinfo/$TZ /etc/localtime \
    && echo $TZ > /etc/timezone \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# 빌더 이미지에서 jar 파일만 복사
COPY --from=builder /build/build/libs/*-SNAPSHOT.jar ./app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "-Djava.security.egd=file:/dev/./urandom", "-Dsur
```

2. 외부 서비스

```
### AWS S3
- AWS_BUCKET_NAME
- AWS_ACCESS_KEY
- AWS_SECRET_KEY
- AWS_URL

### Open AI
- OPEN_AI_SECRET_KEY
```

Github
- GITHUB_CLIENT_ID
- GITHUB_CLIENT_SECRET

4. 시연 시나리오

1. 메인 페이지 (로그인)
1. 회원가입(로그인)
2. 게임 시작

2. 개인전 진행
1. 개인전 매칭
2. 에디터
3. 코드 실행
4. 제출
5. 결과 확인
6. AI 피드백
7. 상대 코드 보기

3. 친구, 초대, 채팅
1. 친구 요청
2. 대화 걸기
3. 친구 채팅
5. 게임 요청 거절
6. 친구 채팅
7. 팀전 매칭 입장 후 초대
6. 팀전(방입장)
7. 친구 초대
8. 팀방 입장

4. 팀전 진행
1. 게임 시작
2. 동시 편집기로 함께 문제 풀기
3. 보너스 문제 풀기
4. 아이템 획득 및 사용

5. 문제 풀이 완료
6. 결과확인

5. 프로필, 랭킹페이지
1. 랭킹페이지 (필터, 검색 확인)
2. 타인의 프로필 (프로필, 개인전 기록, 팀전 기록 확인)
3. 다른 사람 코드 , AI 코칭 받은 기록 확인