🔧 포팅 메뉴얼

1. 개발 환경

🖥️ Frontend

- Node.js
- Next.js
- React.js
- TypeScript

📋 Backend

- Java
- Spring Boot
- Spring Security
- Spring Data JPA
- JWT
- Gradle

📊 Data

- Python

📚 DB

- MySQL
- Redis
- S3

🌍 Deploy

- AWS EC2 Ubuntu
- Jenkins
- Docker
- Docker-compose
- Nginx
- Jenkins

📣 Communication

- 형상 관리 - `Gitlab`, `Sourcetree`
- 이슈 및 스크럼 관리 - `Jira`
- 의사소통, 협업 - `Notion`, `Mattermost`
- 디자인 - `Figma`

2. EC2 서버 설정

Jenkins

1. docker가 설치된 jenkins 이미지 만들기

`docker-install.sh` 파일을 먼저 생성 후 아래의 코드 입력합니다.

```
#!/bin/sh
apt-get update && \
apt-get -y install apt-transport-https \
     ca-certificates \
     curl \
     gnupg2 \
     zip \
     unzip \
     software-properties-common && \
curl -fsSL https://download.docker.com/linux/$(. /etc/os-rele
add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/$(. /e
   $(lsb_release -cs) \
   stable" && \
apt-get update && \
apt-get -y install docker-ce
```

`dockerfile` 생성해줍니다

```
FROM jenkins/jenkins:lts
```

```
#root 계정으로 변경(for docker install)
USER root

COPY docker-install.sh /docker-install.sh
RUN chmod +x /docker-install.sh
RUN /docker-install.sh

RUN usermod -aG docker jenkins
USER jenkins
```

위와 같이 `docker-install.sh`와 `dockerfile`을 만든후 아래의 명령어로 image 생성합니다.

```
docker build -t jenkins-docker .
```

2. Front Pipeline Script

본 프로젝트는 `front`, `back` 2개의 브랜치로 나누어 관리하기 때문에 2개의 파이프라인에 각각의 script를 작성해야 합니다.

우선 Front부분 파이프라인 script 입니다.

`Front pipeline Script`

```
pipeline {
    agent any
    tools {nodejs "nodejs"}

    environment {
        repository = <Docker Repository> // Docker 이미지의 저장
        dockerImage = '' // Docker 이미지 변수 초기화

        registryCredential = <Docker Credential>

        releaseServerAccount = <Server Account>
        releaseServerUri = <Server Uri>
        releasePort = '80'
```

```
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: <Gitlab Clone Branch>, // clone 받
                credentialsId: <GitLab Credential>,
                url: <GitLab Url>
            }
        }
        stage('Node Build') {
            steps {
                // 상대 경로 사용. Jenkins 워크스페이스 내부에서 작업
                // 'sudo' 사용은 Jenkins 스크립트에서 권장되지 않습니
                // 필요한 권한은 Jenkins 사용자가 이미 가지고 있어야
                dir ('../zipzoong-frontend/frontend/zipjung')
                    sh 'npm install -g npm@latest'
                    sh 'npm install next'
                    sh 'GENERATE_SOURCEMAP=false npm run build
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                // 상대 경로 사용을 위해 변경
                dir('../zipzoong-frontend/frontend/zipjung')
                    script {
                        docker.withRegistry('', registryCrede
                            sh "docker buildx create --use --
                            sh "docker buildx build --platfor
                            sh "docker buildx build --platfor
                        }
                    }
                }
            }
        }
        stage('Before Service Stop') {
            steps {
```

```
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh '''
                    CONTAINER_IDS=$(ssh -o StrictHostKeyCheck
                    if [ ! -z "$CONTAINER_IDS" ]; then
                        echo "$CONTAINER_IDS" | xargs -I {} s
                        echo "$CONTAINER_IDS" | xargs -I {} s
                    fi
                    ssh -o StrictHostKeyChecking=no $releaseS
                    '''
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh "ssh -o StrictHostKeyChecking=no $rele
                }
            }
        }
        stage('Volume Initialization') {
            steps {
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $rele
                        ssh -o StrictHostKeyChecking=no $rele
                        ssh -o StrictHostKeyChecking=no $rele
                    '''
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $rele
                        ssh -o StrictHostKeyChecking=no $rele
                    '''
                }
```

```
            }
        }
        stage('Service Check') {
            steps {
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh '''
                        #!/bin/bash

                        for retry_count in $(seq 20)
                        do
                          if curl -s "https://zipzoong.store"
                          then
                              curl -d '{
                                          "text": "[FRONTEND]
                                          "attachments": [
                                            {
                                              "color": "good"
                                              "text": "FRONTE
                                            }
                                          ]
                                       }' -H "Content-Type: a
                              break
                          fi

                          if [ $retry_count -eq 20 ]
                          then
                            curl -d '{
                                        "text": "[FRONTEND] D
                                        "attachments": [
                                            {
                                              "color": "dange
                                              "text": "FRONTE
                                            }
                                        ]
                                    }' -H "Content-Type: appl
                            exit 1
                          fi
```

```
                            echo "The server is not alive yet. 
                            sleep 5
                        done
                    '''
                }
            }
        }
    }
}
```

`Dockerfile` 작성

```
FROM node:20.10
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
EXPOSE 3000

CMD ["npm", "start"]
```

3. Back Pipeline Script

다음은 Back 부분 파이프라인 script 입니다.

`Back pipeline script`

```
pipeline {
    agent any

    environment {
        repository = <Docker Repository> // Docker 이미지의 저장
        dockerImage = '' // Docker 이미지 변수 초기화

        registryCredential = <Docker Credential>
```

```
        releaseServerAccount = <Server Account>
        releaseServerUri = <Server Uri>
        releasePort = '8081'
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: <Gitlab Clone Branch>, // clone 받
                credentialsId: <GitLab Credential>,
                url: <GitLab Url>
            }
        }
        stage('Jar Build') {
            steps {
                dir('backend'){
                    sh 'chmod +x ./gradlew' // gradlew 파일에 실
                    sh './gradlew clean bootJar' // Gradle로 J
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                sh 'mkdir -p ../backend/'
                sh 'cp ./backend/build/libs/ZipJoong-0.0.1-SNA
                dir('./backend') {
                    script {
                        docker.withRegistry('', registryCrede
                            sh "docker buildx create --use --
                            sh "docker buildx build --platfor
                            sh "docker buildx build --platfor
                        }
                    }
                }

            }
        }
        stage('Before Service Stop') {
```

```
            steps {
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh '''
                        if
                            test "`ssh -o StrictHostKeyChecki
                            ssh -o StrictHostKeyChecking=no $
                            ssh -o StrictHostKeyChecking=no $
                            ssh -o StrictHostKeyChecking=no $
                        fi
                    '''
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh "ssh -o StrictHostKeyChecking=no $rele
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $rele
                    '''
                }
            }
        }
        stage('Service Check') {
            steps {
                sshagent(credentials: ['zipzoong-ubuntu']) {
                    sh '''
                        #!/bin/bash

                        for retry_count in \$(seq 20)
                        do
                          if curl -s "http://back.zipzoong.st
```

```
                        then
                            curl -d '{
                                        "text": "[BACKEND]
                                        "attachments": [
                                          {
                                            "color": "good"
                                            "text": "BACKEN
                                          }
                                        ]
                                    }' -H "Content-Type:
                            break
                        fi

                        if [ $retry_count -eq 20 ]
                        then
                          curl -d '{
                                      "text": "[BACKEND] De
                                      "attachments": [
                                          {
                                            "color": "dange
                                            "text": "BACKEN
                                          }
                                      ]
                                  }' -H "Content-Type: appl
                          exit 1
                        fi

                        echo "The server is not alive yet.
                        sleep 5
                    done
                '''
            }
        }
    }
}
```

`dockerfile` 작성

```
FROM blank98/zipzoong-base:latest
WORKDIR /usr/src/app
COPY ./build/libs/ZipJoong-0.0.1-SNAPSHOT.jar .
EXPOSE 8081

ENTRYPOINT ["java","-jar", "ZipJoong-0.0.1-SNAPSHOT.jar"]
```

4. Back 환경변수 설정

- application.properties

```
# Server Configuration
server.port=8081
server.servlet.context-path=/
server.servlet.encoding.charset=UTF-8
server.servlet.encoding.enabled=true
server.servlet.encoding.force=true

# Database Configuration
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://${MYSQL_HOST}:${MYSQL_PORT
spring.datasource.username=${MYSQL_USER}
spring.datasource.password=${MYSQL_PASSWORD}

# JPA/Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.b
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# AWS S3 Configuration
cloud.aws.credentials.accessKey=${AWS_ACCESS_KEY}
cloud.aws.credentials.secretKey=${AWS_SECRET_KEY}
cloud.aws.s3.bucket=zipzoong-bucket
cloud.aws.region.static=ap-northeast-2
```

```
# Redis Configuration
spring.data.redis.host=${REDIS_HOST}
spring.data.redis.port=${REDIS_PORT}

# JWT Configuration
spring.jwt.secret=${JWT_SECRET}

# OAuth2 Configuration
spring.profiles.include=oauth2

# SecurityConfig
app.security.permitAllGetPatterns=/connect/**,/user/nickname/
  /board,/board/detail/*,/board/search/*,/comment/byBoard/*,\
  /swagger-resources/**,/swagger-ui/**,/v3/api-docs,/v3/api-d
app.security.permitAllPostPatterns=/combination/product,/comb
  /board/hit/*,/board/file/*,/survey
```

- application-oauth2.properties (소셜로그인)

```
#Kakao
spring.security.oauth2.client.provider.kakao.authorization-ur
spring.security.oauth2.client.provider.kakao.token-uri=https:
spring.security.oauth2.client.provider.kakao.user-info-uri=ht
spring.security.oauth2.client.provider.kakao.user-name-attrib

spring.security.oauth2.client.registration.kakao.client-name=
spring.security.oauth2.client.registration.kakao.client-id=${
spring.security.oauth2.client.registration.kakao.client-secre
spring.security.oauth2.client.registration.kakao.redirect-uri
spring.security.oauth2.client.registration.kakao.client-authe
spring.security.oauth2.client.registration.kakao.authorizatio
spring.security.oauth2.client.registration.kakao.scope=profil

#Google
spring.security.oauth2.client.registration.google.client-name
spring.security.oauth2.client.registration.google.client-id=$
```

```
spring.security.oauth2.client.registration.google.client-secr
spring.security.oauth2.client.registration.google.redirect-ur
spring.security.oauth2.client.registration.google.authorizati
spring.security.oauth2.client.registration.google.scope=profi

#Naver
spring.security.oauth2.client.provider.naver.authorization-ur
spring.security.oauth2.client.provider.naver.token-uri=https:
spring.security.oauth2.client.provider.naver.user-info-uri=ht
spring.security.oauth2.client.provider.naver.user-name-attrib

spring.security.oauth2.client.registration.naver.client-name=
spring.security.oauth2.client.registration.naver.client-id=${
spring.security.oauth2.client.registration.naver.client-secre
spring.security.oauth2.client.registration.naver.redirect-uri
spring.security.oauth2.client.registration.naver.authorizatio
spring.security.oauth2.client.registration.naver.scope=name,e
```

Nginx

1. SSL 인증서 발급

Certbot 설치

```
sudo apt-get install certbot
```

SSL 인증서 발급

```
sudo certbot certonly --manual --preferred-challenges dns -d
```

2. Nginx 설정

nginx 설정 파일 작성

- `/etc/nginx/conf.d` 경로에 `defaut.conf` 설정 파일을 만들어 아래의 코드를 입력합니다

```
server {
    listen 443 ssl;
    server_name back.zipzoong.store;

    ssl_certificate /etc/letsencrypt/live/zipzoong.store/full
    ssl_certificate_key /etc/letsencrypt/live/zipzoong.store/

    location / {
        proxy_pass http://zipjoong-back:8081;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_http_version 1.1;
        proxy_request_buffering off;
        proxy_buffering off;
    }
}



server {
    listen 443 ssl;
    server_name zipzoong.store www.zipzoong.store;

    ssl_certificate /etc/letsencrypt/live/zipzoong.store/full
    ssl_certificate_key /etc/letsencrypt/live/zipzoong.store/

    location /_next/static/chunks/ {
        alias /nginx/html/_next/static/chunks/;
        expires 365d;
        access_log off;
    }

    location /static/ {
        alias /nginx/html/public/;
        expires 365d;
```

```
        access_log off;
    }

      location / {
          proxy_pass http://zipjoong-front:3000;
          proxy_set_header Host $host:$server_port;
          proxy_set_header X-Real-IP $remote_addr;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwa
          proxy_set_header X-Forwarded-Proto $scheme;

          proxy_http_version 1.1;
          proxy_request_buffering off;
          proxy_buffering off;
      }

}



server {
    listen 80;
    server_name www.zipzoong.store;
    return 301 https://$host$request_uri;
}



server {
    listen 443 ssl;
    server_name jenkins.zipzoong.store;

    ssl_certificate /etc/letsencrypt/live/zipzoong.store/full
    ssl_certificate_key /etc/letsencrypt/live/zipzoong.store/

    location / {
        proxy_pass http://jenkins:8080;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_http_version 1.1;
        proxy_request_buffering off;
        proxy_buffering off;
        add_header 'X-SSH-Endpoint' 'jenkins.zipzoong.store' 
    }
}
```

Docker-compose

마지막으로 `Docker-compose.yml` 파일을 생성 후 아래의 코드를 입력합니다.

```
version: '1'
services:
  jenkins:
    image: jenkins/jenkins

    ports:
      - 8080:8080

    volumes:
      - /home/ubuntu/jenkins:/var/jenkins_home # jenkins가 돌아
      - /home/ubuntu/.ssh:/var/jenkins_home/.ssh # 호스트 ssh 키
      - /var/run/docker.sock:/var/run/docker.sock # host의 doc

    networks:
      - nat

  db:
    image: mysql:latest
    ports:
      - 3300:3306

    volumes:
```

```
        - mysql:/home/ubuntu/mysql

    env_file:
      - ./a204.env
    networks:
      - nat

  zipjoong-back:
    image: awetumnn/a204-backend:latest

    ports:
      - 8081:8081

    volumes:
      - /home/ubuntu/python:/var/recommend_python # 조합 추천을

    environment:
      - RECOM_PATH=/var/recommend_python

    env_file:
      - ./a204.env

    depends_on:
      - db
    networks:
      - nat

  zipjoong-front:
    image: awetumnn/a204-frontend:latest

    ports:
      - 3000:3000

    build:
      context: .

    volumes:
      - nextjs-static:/app/.next/static
```

```
      - public-files:/app/public

    networks:
      - nat

  redis:
    image: redis:latest
    ports:
      - 6379:6379
    volumes:
      - redis-data:/data

  nginx:
   image: nginx:latest
    ports:
      - 80:80
      - 443:443
    volumes:
      - /home/ubuntu/nginx/conf.d:/etc/nginx/conf.d # conf.d
      - /etc/letsencrypt:/etc/letsencrypt
      - nextjs-static:/nginx/html/_next/static # Next.js 정적
      - public-files:/nginx/html/public # Next.js public 폴더
    restart: always # 꺼져도 다시 실행
    depends_on:
      - zipjoong-back
      - zipjoong-front
    networks:
      - nat

networks:
  nat:
    external: true

volumes:
  mysql:
  redis-data:
  nextjs-static:
  public-files:
```

3. 빌드

위의 모든 과정을 완료하였다면 서비스를 실행할 수 있습니다.

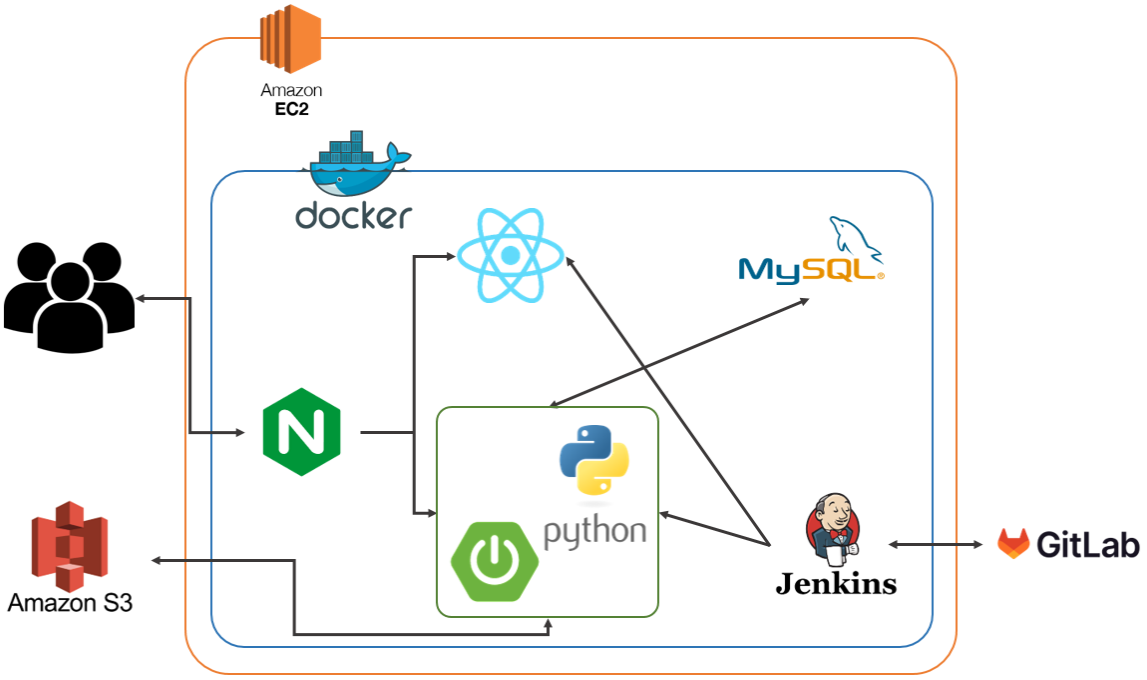
```
docker-compose up
```

- jenkins 주소

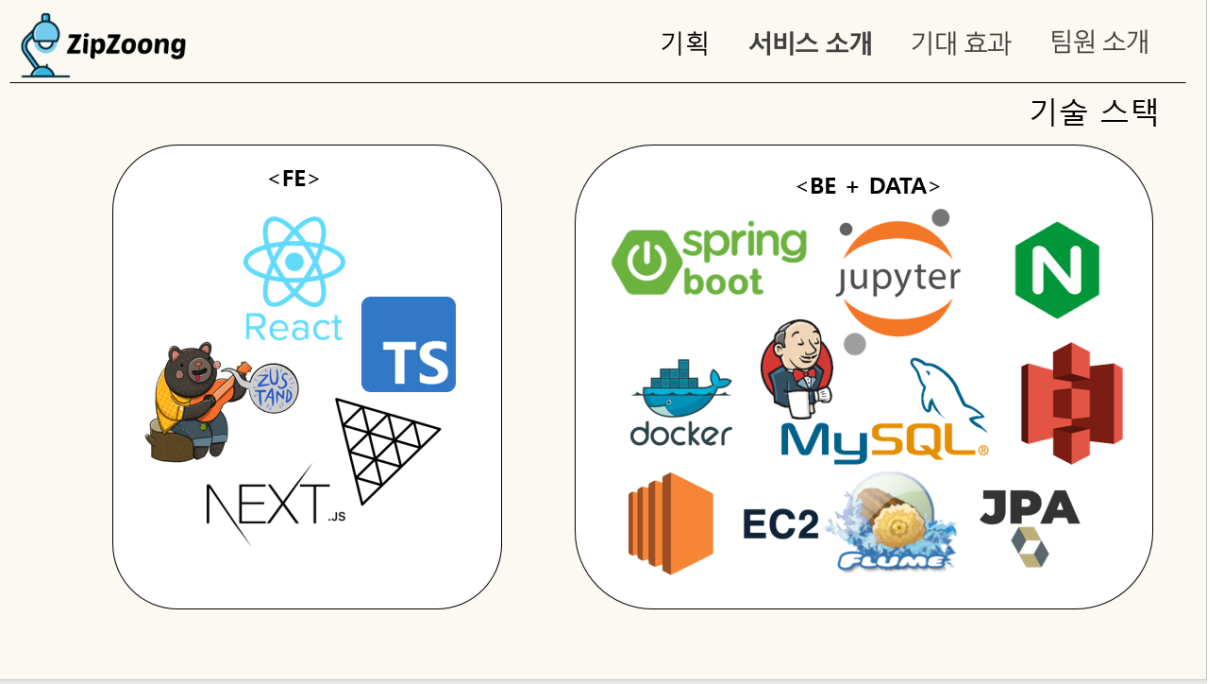
https://jenkins.zipzoong.store/

---

4. 아키텍처 구성



5. 기술사용 스택



6. DB 프로퍼티

```
CREATE TABLE `users` (
    `user_id`   varchar(50) NOT NULL,
    `user_nickname` varchar(30) NULL,
    `user_img`  varchar(510)    NULL,
    `user_created_at`   DATETIME    NULL    DEFAULT CURRENT_T
    `user_updated_at`   DATETIME    NULL    DEFAULT CURRENT_T
    `user_is_deleted`   BIT NULL    DEFAULT FALSE,
    `user_deleted_at`   DATETIME    NULL    DEFAULT CURRENT_T
);

CREATE TABLE `keyboard` (
    `keyboard_id`   INT NOT NULL,
    `keyboard_connect`  ENUM    NULL    DEFAULT ENUM('WIRE',
    `keyboard_interface`    ENUM    NULL    DEFAULT ENUM('USB
```

```
    `keyboard_switch`   varchar(20) NULL    COMMENT '적축, 갈축
    `keyboard_led`  varchar(20) NULL    COMMENT '조명 색상',
    `keyboard_layout`   varchar(20) NULL    COMMENT 'QWERTY, A
    `keyboard_color`    varchar(20) NULL,
    `keyboard_form` varchar(20) NULL    COMMENT '텐키리스 등',
    `keyboard_contact`  varchar(20) NULL    COMMENT '기계식, 맴
);

CREATE TABLE `mouse` (
    `mouse_id`  INT NOT NULL,
    `mouse_connect` ENUM    NULL    DEFAULT ENUM('WIRE', 'WIR
    `mouse_interface`   ENUM    NULL    DEFAULT ENUM('USB', '
    `mouse_type`    varchar(20) NULL    COMMENT 'ERGONOMIC , A
    `mouse_dpi` varchar(20) NULL,
    `mouse_color`   varchar(20) NULL    COMMENT '화이트톤, 블랙톤
    `mouse_weight`  INT NULL    COMMENT 'g',
    `mouse_width`   INT NULL    COMMENT 'mm',
    `mouse_length`  INT NULL    COMMENT 'mm',
    `mouse_height`  INT NULL    COMMENT 'mm'
);

CREATE TABLE `monitor` (
    `monitor_id`    INT NOT NULL,
    `monitor_size`  INT NULL    COMMENT 'inch',
    `monitor_resolution`    varchar(20) NULL,
    `monitor_aspect_ratio`  varchar(20) NULL,
    `monitor_refresh_rate`  varchar(20) NULL,
    `monitor_panel_type`    ENUM    NULL    DEFAULT ENUM('TN'
    `monitor_panel_form`    varchar(20) NULL
);

CREATE TABLE `board` (
    `board_id`  BIGINT  NOT NULL    DEFAULT AUTO_INCREMENT,
    `board_title`   varchar(200)    NULL,
    `board_content` text    NULL,
    `board_hit` INT NULL    DEFAULT 0,
    `board_thumbnail`   varchar(255)    NULL,
    `board_created_at`  DATETIME    NULL    DEFAULT CURRENT_T
```

```
    `board_updated_at`  DATETIME    NULL    DEFAULT CURRENT_T
    `board_deleted_at`  DATETIME    NULL    DEFAULT CURRENT_T
    `board_is_deleted`  BIT NULL    DEFAULT FALSE,
    `user_id`   varchar(50) NOT NULL
);

CREATE TABLE `comment` (
    `comment_id`    BIGINT  NOT NULL    DEFAULT AUTO_INCREMEN
    `comment_content`   varchar(500)    NULL,
    `comment_created_at`    DATETIME    NULL    DEFAULT CURRE
    `comment_updated_at`    DATETIME    NULL    DEFAULT CURRE
    `comment_deleted_at`    DATETIME    NULL    DEFAULT CURRE
    `comment_is_deleted`    BIT NULL    DEFAULT FALSE,
    `user_id`   INT NOT NULL,
    `board_id`  BIGINT  NOT NULL
);

CREATE TABLE `product_like` (
    `user_id`   varchar(50) NOT NULL,
    `product_id`    INT NOT NULL,
    `product_like_created_at`   DATETIME    NULL    DEFAULT C
    `product_like_deleted_at`   DATETIME    NULL    DEFAULT C
    `product_like_is_deleted`   BIT NULL    DEFAULT FALSE
);

CREATE TABLE `combination_like` (
    `user_id`   varchar(50) NOT NULL,
    `combination_id`    BIGINT  NOT NULL,
    `combination_like_created_at`   DATETIME    NULL    DEFAU
    `combination_like_deleted_at`   DATETIME    NULL    DEFAU
    `combination_like_is_deleted`   BIT NULL    DEFAULT FALSE
);

CREATE TABLE `combination` (
    `combination_id`    BIGINT  NOT NULL    DEFAULT AUTO_INCR
    `user_id`   varchar(50) NOT NULL,
    `combination_price` INT NULL,
    `combination_created_at`    DATETIME    NULL    DEFAULT C
```

```
    `combination_updated_at`    DATETIME    NULL    DEFAULT C
);

CREATE TABLE `survey` (
    `survey_id` INT NOT NULL,
    `user_id`   varchar(50) NOT NULL,
    `total_price`   INT NOT NULL    DEFAULT 10  COMMENT '10~5
    `survey_detail` ENUM    NOT NULL    DEFAULT SIMPLE  COMME
    `monitor_price` INT NOT NULL    DEFAULT 0   COMMENT '단위=
    `keyboard_price`    INT NOT NULL    DEFAULT 0   COMMENT '
    `mouse_price`   INT NOT NULL    DEFAULT 0   COMMENT '단위=
    `monitor_usage` INT NOT NULL    DEFAULT 16  COMMENT '사무=
    `keyboard_usage`    INT NOT NULL    DEFAULT 16  COMMENT '
    `mouse_usage`   INT NOT NULL    DEFAULT 16  COMMENT '사무=
    `keyboard_color`    ENUM    NOT NULL    DEFAULT NONE    C
    `mouse_color`   ENUM    NOT NULL    DEFAULT NONE    COMME
    `keyboard_layout`   INT NOT NULL    DEFAULT 4   COMMENT '
    `keyboard_connection`   ENUM    NOT NULL    DEFAULT BOTH
    `mouse_connection`  ENUM    NOT NULL    DEFAULT BOTH    C
    `keyboard_health`   Bool    NOT NULL    DEFAULT False   C
    `mouse_health`  Bool    NOT NULL    DEFAULT False   COMME
    `monitor_size`  INT NULL    DEFAULT 4   COMMENT '24인치 미
    `monitor_ratio` INT NULL    DEFAULT 1   COMMENT '16:9=1,
    `monitor_panel` ENUM    NOT NULL    DEFAULT FLAT    COMME
    `keyboard_type` ENUM    NOT NULL    DEFAULT MECHANICAL  C
    `keyboard_sound`    ENUM    NOT NULL    DEFAULT RED COMME
    `mouse_sound`   Bool    NULL    DEFAULT True    COMMENT '
);

CREATE TABLE `product` (
    `product_id`    INT NOT NULL    DEFAULT AUTO_INCREMENT,
    `product_name`  varchar(100)    NULL,
    `product_price` INT NULL,
    `product_type`  ENUM    NULL    DEFAULT ENUM('MONITOR', '
    `product_img`   varchar(510)    NULL,
    `product_brand` varchar(100)    NULL
);
```

```
CREATE TABLE `board_combination` (
    `board_id`  BIGINT  NOT NULL    DEFAULT AUTO_INCREMENT,
    `combination_id`    BIGINT  NOT NULL    DEFAULT AUTO_INCR
);

CREATE TABLE `combination_product` (
    `combination_id`    BIGINT  NOT NULL    DEFAULT AUTO_INCR
    `product_id`    INT NOT NULL    DEFAULT AUTO_INCREMENT,
    `combination_product_num`   INT NULL
);

CREATE TABLE `file` (
    `file_id`   BIGINT  NOT NULL    DEFAULT AUTO_INCREMENT,
    `file_path` varchar(128)    NULL    COMMENT 's3에 저장된 경
    `file_type` varchar(10) NULL    COMMENT 'jpg, png 등',
    `file_created_at`   DATETIME    NULL    DEFAULT CURRENT_T
    `board_id`  BIGINT  NOT NULL
);

ALTER TABLE `users` ADD CONSTRAINT `PK_USERS` PRIMARY KEY (
    `user_id`
);

ALTER TABLE `keyboard` ADD CONSTRAINT `PK_KEYBOARD` PRIMARY K
    `keyboard_id`
);

ALTER TABLE `mouse` ADD CONSTRAINT `PK_MOUSE` PRIMARY KEY (
    `mouse_id`
);

ALTER TABLE `monitor` ADD CONSTRAINT `PK_MONITOR` PRIMARY KEY
    `monitor_id`
);

ALTER TABLE `board` ADD CONSTRAINT `PK_BOARD` PRIMARY KEY (
    `board_id`
);
```

```
ALTER TABLE `comment` ADD CONSTRAINT `PK_COMMENT` PRIMARY KEY
    `comment_id`
);

ALTER TABLE `product_like` ADD CONSTRAINT `PK_PRODUCT_LIKE` PI
    `user_id`,
    `product_id`
);

ALTER TABLE `combination_like` ADD CONSTRAINT `PK_COMBINATION_
    `user_id`,
    `combination_id`
);

ALTER TABLE `combination` ADD CONSTRAINT `PK_COMBINATION` PRI
    `combination_id`
);

ALTER TABLE `survey` ADD CONSTRAINT `PK_SURVEY` PRIMARY KEY (
    `survey_id`
);

ALTER TABLE `product` ADD CONSTRAINT `PK_PRODUCT` PRIMARY KEY
    `product_id`
);

ALTER TABLE `board_combination` ADD CONSTRAINT `PK_BOARD_COMB
    `board_id`,
    `combination_id`
);

ALTER TABLE `combination_product` ADD CONSTRAINT `PK_COMBINAT
    `combination_id`,
    `product_id`
);

ALTER TABLE `file` ADD CONSTRAINT `PK_FILE` PRIMARY KEY (
```

```
    `file_id`
);

ALTER TABLE `keyboard` ADD CONSTRAINT `FK_product_TO_keyboard_
    `keyboard_id`
)
REFERENCES `product` (
    `product_id`
);

ALTER TABLE `mouse` ADD CONSTRAINT `FK_product_TO_mouse_1` FO
    `mouse_id`
)
REFERENCES `product` (
    `product_id`
);

ALTER TABLE `monitor` ADD CONSTRAINT `FK_product_TO_monitor_1
    `monitor_id`
)
REFERENCES `product` (
    `product_id`
);

ALTER TABLE `product_like` ADD CONSTRAINT `FK_users_TO_produc
    `user_id`
)
REFERENCES `users` (
    `user_id`
);

ALTER TABLE `product_like` ADD CONSTRAINT `FK_product_TO_prod
    `product_id`
)
REFERENCES `product` (
    `product_id`
);
```

```
ALTER TABLE `combination_like` ADD CONSTRAINT `FK_users_TO_co
    `user_id`
)
REFERENCES `users` (
    `user_id`
);

ALTER TABLE `combination_like` ADD CONSTRAINT `FK_combination_
    `combination_id`
)
REFERENCES `combination` (
    `combination_id`
);

ALTER TABLE `board_combination` ADD CONSTRAINT `FK_board_TO_b
    `board_id`
)
REFERENCES `board` (
    `board_id`
);

ALTER TABLE `board_combination` ADD CONSTRAINT `FK_combinatio
    `combination_id`
)
REFERENCES `combination` (
    `combination_id`
);

ALTER TABLE `combination_product` ADD CONSTRAINT `FK_combinat
    `combination_id`
)
REFERENCES `combination` (
    `combination_id`
);

ALTER TABLE `combination_product` ADD CONSTRAINT `FK_product_
    `product_id`
)
```

```
REFERENCES `product` (
    `product_id`
);
```