포팅 매뉴얼

📎 gitlab 소스 클론 이후 빌드 및 배포

💡 웹서버

Docker

- version 3.9

```
version: "3.9"
services:
  front:
    container_name: front
    #build시 적용되는 구성 옵션
    build:
      #Dockerfile이 포함된 디렉토리의 경로 또는 git 리포지토리의 URL
      context: ./front
    ports:
      - 3000:80
    networks:
      - aimer
    restart: unless-stopped
    depends_on:
      - spring
  spring:
    container_name: spring
    build:
      context: ./back_spring/aipjt
    ports:
      - 8080:8080
    networks:
      - aimer
    restart: unless-stopped
  django:
    container_name: django
    build:
      context: ./back_django
    ports:
      - 8081:8081
    networks:
      - aimer
    restart: unless-stopped
    depends_on:
      - spring
  redis:
    container_name: redis
    image: redis:latest
    networks:
      - aimer
    ports:
      - 6379:6379
    volumes:
      - ./data:/data
    command: redis-server --appendonly yes
  mysql:
    container_name: mysql
    image: wns312/mysql-utf8
    networks:
      - aimer
    ports:
      - 3306:3306
    volumes:
      - /var/lib/mysql:/var/lib/mysql
    environment:
      - MYSQL_DATABASE=ssafydb
      - MYSQL_ROOT_PASSWORD=ssafy
    command: --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci
```

```
networks:
  aimer:
```

NGINX

- version 1.18. 0

```
server {

        server_name j5b106.p.ssafy.io;

        location / {
                proxy_pass http://127.0.0.1:3000/;
        }

        location /api {
                proxy_pass http://127.0.0.1:8080/api;
        }

        location /stomp/chat {
                proxy_pass http://127.0.0.1:8080/stomp/chat;
                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_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header Accept-Encoding "";
        }
        location /stomp/room {
                proxy_pass http://127.0.0.1:8080/stomp/room;
                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_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header Accept-Encoding "";
        }


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j5b106.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j5b106.p.ssafy.io/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = j5b106.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80 default_server;
        listen [::]:80 default_server;

        server_name j5b106.p.ssafy.io;
    return 404; # managed by Certbot
}
```

Jenkins

```
pipeline {
    agent any

    stages {
        stage('Clone') {
            steps {
                mattermostSend color: "#525252", message: "Build STARTED : ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link
                git(
```

```
                  url: 'https://lab.ssafy.com/s05-ai-speech/S05P21B106/',
                  credentialsId: 'GitLabJYKUserPass',
                  branch: 'develop'
                )
            }
        }
        stage('Front Env Setting') {
            steps {
                  sh """
cat > front/.env.production <<EOF
VUE_APP_API_URL=https://j5b106.p.ssafy.io/api
VUE_APP_STOMP_URL=https://j5b106.p.ssafy.io/stomp/room
"""
            }
        }
        stage('Django Env Setting') {
            steps {
                  sh """
cat > back_django/.env <<EOF
ACCESS_KEY_ID = 'AKIATD5L7CMI2SALOZHB'
ACCESS_SECRET_KEY = '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'
BUCKET_NAME = 'ssafy-bucket'
"""
            }
        }
        stage('Spring Env Setting') {
            steps {
                sh """
cat > back_spring/aipjt/src/main/resources/production.yml <<EOF
spring:
  redis:
    host: redis
    port: 6379
  datasource:
    url: jdbc:mysql://mysql:3306/ssafydb
    username: root
    password: ssafy
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true

logging.level:
  org.hibernate.SQL: debug

"""
                sh """
cat > back_spring/aipjt/src/main/resources/aws.yml <<EOF

cloud:
  aws:
    credentials:
      accessKey: ''
      secretKey: ''
    s3:
      bucket: ssafy-bucket
    region:
      static: us-east-2
    stack:
      auto: false
"""
            }
        }



        stage('Build') {
            parallel {
                stage('Front Build'){
                    steps {
                        dir('front') {
                            sh "docker build -t aimage_front ."
                        }
                    }
                }
                stage('Spring Build'){
                    steps {
                        dir('back_spring/aipjt') {
                            sh "docker build -t aimage_spring ."
```

```
                    }
                }
            }
            stage('Django Build'){
                steps {
                    dir('back_django') {
                        sh "docker build -t aimage_django ."
                    }
                }
            }
        }
    }
    stage('Deploy') {
        steps {
            sh "docker-compose up -d"
        }
    }
}
post{
    success{
        mattermostSend color: "good", message: "Build SUCCESS : ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to Jenk
    }
    failure{
        mattermostSend color: "danger", message: "Build FAILED : ${env.JOB_NAME} #${env.BUILD_NUMBER} (<${env.BUILD_URL}|Link to Jen
    }
}
}
```

💡 Frontend

Vue

- package.json

```
  "dependencies": {
    "axios": "^0.21.4",
    "core-js": "^3.6.5",
    "primeflex": "^2.0.0",
    "primeicons": "^4.1.0",
    "primevue": "^3.7.2",
    "sockjs-client": "^1.5.2",
    "vue": "^3.0.0",
    "vue-router": "^4.0.0-0",
    "vuex": "^4.0.0-0",
    "vuex-persistedstate": "^4.1.0",
    "webstomp-client": "^1.2.6"
  }
```

- .env.development

```
VUE_APP_API_URL=http://localhost:8080/api
VUE_APP_STOMP_URL=http://localhost:8080/stomp/room
```

- .eslintrc.js

```
module.exports = {
  env: {
    browser: true,
    es6: true,
    node: true
  },
  extends: ["eslint:recommended", "plugin:vue/vue3-strongly-recommended"],
  globals: {
    Atomics: "readonly",
    SharedArrayBuffer: "readonly"
  },
```

```
  parserOptions: {
    ecmaVersion: 2018,
    sourceType: "module"
  },
  plugins: ["vue"],
  rules: {
    "vue/max-attributes-per-line": [
      "warn",
      {
        singleline: {
          max: 4,
          allowFirstLine: true
        },
        multiline: {
          max: 1,
          allowFirstLine: false
        }
      }
    ],
    indent: ["warn", 2],
    "vue/html-indent": ["warn", 2],
    "vue/singleline-html-element-content-newline": ["off"],
    "vue/html-quotes": ["off"],
    "vue/html-self-closing": ["off"],
    "vue/attribute-hyphenation": ["off"]
  }
};
```

💡 Backend

MySql

- MySQL Workbench 8.0

Spring

- plugins

```
plugins {
   id 'org.springframework.boot' version '2.5.4'
   id 'io.spring.dependency-management' version '1.0.11.RELEASE'
   id 'java'
}
```

- dependencies

```
dependencies {
    // Validation
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    // Redis
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    // MySQL
    implementation 'mysql:mysql-connector-java'
    // Jpa Hibernate
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    // Spring Security
    implementation 'org.springframework.boot:spring-boot-starter-security'
    // Dev Tools
    developmentOnly('org.springframework.boot:spring-boot-devtools')
    // Web
    implementation 'org.springframework.boot:spring-boot-starter-web'
    // Websocket
    implementation 'org.springframework.boot:spring-boot-starter-websocket'
    implementation 'org.springframework.integration:spring-integration-stomp'
    // Cloud starter aws
    implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'
    // Lombok
```

```
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testCompileOnly 'org.projectlombok:lombok'
    testAnnotationProcessor 'org.projectlombok:lombok'
    // Test
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    // Kubernetes
    // https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-webflux
    implementation 'org.springframework.boot:spring-boot-starter-webflux:2.5.4'
    implementation 'org.springframework.boot:spring-boot-starter-actuator:2.5.4'
    // Docker image
    implementation 'org.springframework.boot:spring-boot-gradle-plugin:2.5.0'

}
```

- prodution.yml

```
spring:
  redis:
    host: redis
    port: 6379
  datasource:
    url: jdbc:mysql://mysql:3306/ssafydb
    username: root
    password: ssafy
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true

logging.level:
  org.hibernate.SQL: debug
```

- aws.yml

```
cloud:
  aws:
    credentials:
      accessKey: ''
      secretKey: ''
    s3:
      bucket: ssafy-bucket
    region:
      static: us-east-2
    stack:
      auto: false
```

💡 AI

Django

`pip install -r requirements.txt`

`python manage.py runserver`

- .env

  aws S3 key

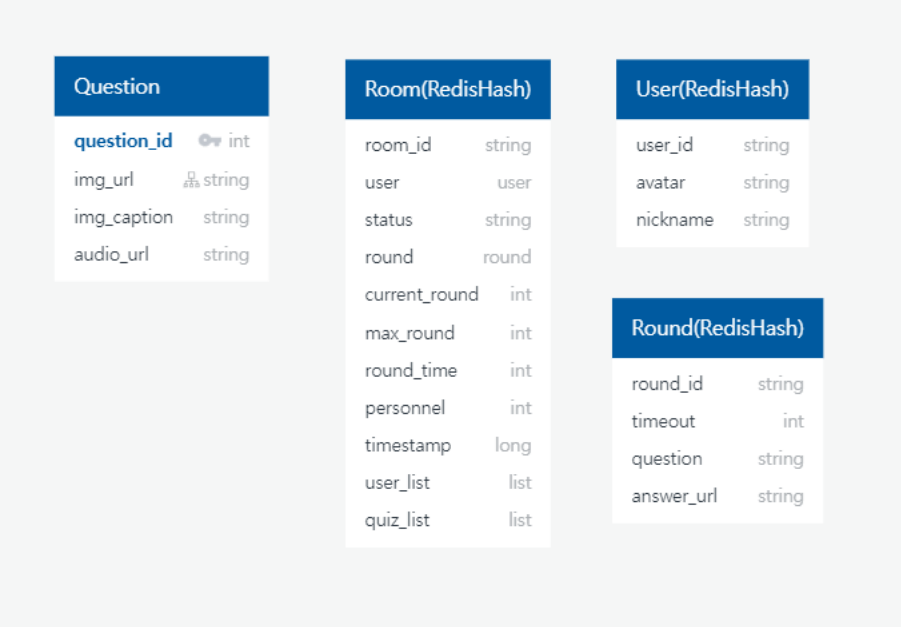
```
ACCESS_KEY_ID = ''
ACCESS_SECRET_KEY = ''
BUCKET_NAME = ''
```

📖 데이터베이스

DataSource

- spring.datasource.username=root
- spring.datasource.password=ssafy

ERD



덤프 데이터

- git clone 후 `S05P21B106/documents/question_db.sql` 사용

⏯️ 시연 시나리오

1. 닉네임 및 아바타 설정
2. 게임 시작을 위한 `방만들기` 클릭
3. `초대 링크` 클릭 후 공유
4. 타 유저는 닉네임 및 아바타 설정 후 `입장하기` 클릭
5. 방장의 게임 설정 및 채팅 시연
6. `게임 시작` 버튼 클릭
7. 문제 출제 및 음성 지원 확인
8. 캔버스 화면 출력 및 타이머 동작 확인
9. 각 라운드별 결과 산정 확인
10. 모든 라운드 후 게임 종료
11. 방으로 돌아가기