



**MiriStock 미리미리 시작하세요!**

**포팅 메뉴얼**

**목차**

1. **개요**
   1. 프로젝트 개요…………………………………………………………………………………………...(3)
   2. 개발환경 및 기술 스택…………………………………………………………..………………...(3)
   3. 프로젝트 사용도구……………………………………………………………..…..………………...(4)
   4. 외부서비스……………………………………………………………..…..……………………………..(5)
2. **프로젝트 빌드**
   1. Front-end 빌드…………….…………………………………………………………………………...(5)
   2. Back-end 빌드 및 환경설정………………………………………………………………….....(7)
   3. Database 설치 및 데이터 파싱과 입력……………………………………………......(14)
   4. OAuth 2.0 설정…………………………………………….......................................................(15)
3. **Infra 구축 및 환경설정**
   1. 환경 구축 및 설치..........................................................................................................(18)
   2. Nginx설정...........................................................................................................................(20)
   3. Front-end Infra설정…………………………………………………………………………………(21)
   4. Back-end Infra설정………………………………………………………………………………….(23)
4. **주요 사용 명령어**…………………………………………………………………………………………….(24)

**1. 개요**

**1-1. 프로젝트 개요**

**- 과거 데이터로 미리 해보는 주식투자 시뮬레이션**

- 과거 주식 데이터를 기반으로 한 모의투자 시뮬레이션 서비스입니다. 해당 시점의 뉴스 기사를 제공하여 등락의 객관적인 자료를 제시하여 앞으로 예비 투자자의 판단력을 기르는 능력을 길러줍니다. 또 기업의 정보를 제공하여 재무제표 또한 중요하다는 것을 알려줍니다.

- 실제 비용부담 없이 주식투자의 전 과정을 경험할 수 있다는 장점이 있습니다. 이를 통해 적절한 매수·매도 타이밍을 잡고, 자신만의 원칙을 세울 수 있습니다.

- 과거 데이터를 통해 실시간 주식 거래를 경험함으로써 주가 흐름에 대한 이해를 할 수 있고, 거래 시스템을 체험해 볼 수 있다는 것이 장점입니다.

**1-2. 개발환경 및 기술 스택**

**Front-End**

- node 16.18.0 (LTS)

- React 18.2.0

- apexchart 3.37.0

- react-apexcharts 1.4.0

- axios 1.3.3

- framer-motion 8.5.5

- prettier 2.8.3

- react-chartjs-2 5.2.0

- react-router-dom 6.8.1

- recharts 2.4.2

- sweetalert2 11.7.1

- zustand 4.3.3

- mui/material 5.11.9

- react-icon 11.10.5

**Back-end**

- Java zulu 11.62.17 (LTS)

- Spring Boot 2.7.8

- Gradle 7.6

- Spring Security 5.7.6

- JSONwebToken 2.13

- actuator 2.7.8

- Spring DATA JPA

- Hibernate 5.6.14

- QueryDSL 5.0.0

- SpringDoc OpenAPI 1.6.14

- Lombok 1.18.22

- Jsoup 1.15.3

**Database**

- MySQL 8.0.32

- Redis 3.0.504

**Infra**

- AWS EC2

- Ubuntu 22.04.1 LTS

- Docker 20.10.23

- Docker-compose 1.25.0

- nginx 1.18.0

- MySQL 8.0.32

- Redis 5.0.7

**1-3. 프로젝트 사용도구**

- Front-end 개발: VSCode 1.75

- Back-end 개발: IntelliJ IDEA 2022

- 이슈관리 : JIRA

- 형상관리 : Gitlab

- UI/UX : Figma

- 커뮤니케이션 : Mattermost, Notion

- CI/CD : Jenkins

**1-4. 외부서비스**

**1. KIS Developer - 한국 투자증권 Open API**

2018년 1월 2일 - 2022년 12월 29일 국내 상장 일별 주식 데이터

<https://apiportal.koreainvestment.com/about>

**2. Open DART**

상장기업 재무정보

<https://opendart.fss.or.kr/guide/detail.do?apiGrpCd=DS003&apiId=2019016>

**3. 한국정보거래소**

국내 상장 주식 목록

<http://data.krx.co.kr/contents/MDC/MAIN/main/index.cmd>

**4. Kakao 소셜 로그인**

OAuth기반 소셜 로그인 API제공

<https://developers.kakao.com/docs/latest/ko/kakaologin/common>

**5. 네이버 뉴스 검색**

해당 종목 과거 특정 시점 뉴스 검색 내용 출력

<https://news.naver.com/>

**2. 프로젝트 빌드**

**2-1. Front-End 빌드**

1. node.js 16.18.0(LTS) 설치

2. 패키지 설치

npm i

3. 시작

npm start

**2-2. package.json**

{  
 "name": "front",  
 "version": "0.1.0",  
 "private": *true*,  
 "dependencies": {  
 "@emotion/react": "^11.10.5",  
 "@emotion/styled": "^11.10.5",  
 "@mui/material": "^5.11.6",  
 "@testing-library/jest-dom": "^5.16.5",  
 "@testing-library/react": "^13.4.0",  
 "@testing-library/user-event": "^13.5.0",  
 "apexcharts": "^3.36.3",  
 "axios": "^1.2.6",  
 "framer-motion": "^8.5.5",  
 "react": "^18.2.0",  
 "react-apexcharts": "^1.4.0",  
 "react-chartjs-2": "^5.2.0",  
 "react-dom": "^18.2.0",  
 "react-icons": "^4.7.1",  
 "react-router-dom": "^6.8.0",  
 "react-scripts": "5.0.1",  
 "recharts": "^2.3.2",  
 "sweetalert2": "^11.7.1",  
 "web-vitals": "^2.1.4",  
 "zustand": "^4.3.2"  
},  
 "scripts": {  
 "start": "react-scripts start",  
 "build": "react-scripts build",  
 "test": "react-scripts test",  
 "eject": "react-scripts eject"  
},  
 "eslintConfig": {  
 "extends": [  
 "react-app",  
 "react-app/jest"  
 ]  
},  
 "browserslist": {  
 "production": [  
 ">0.2%",  
 "not dead",  
 "not op\_mini all"  
 ],  
 "development": [  
 "last 1 chrome version",  
 "last 1 firefox version",  
 "last 1 safari version"  
 ]  
},  
 "devDependencies": {  
 "prettier": "2.8.3"  
}  
}

**2-2. Back-End 빌드 및 환경설정**

1. IntelliJ에서 프로젝트 열기

2. 우측 탭 Gradle항목을 열고 build/clean클릭

3. 이후 other/compileQueryDsl 클릭

텍스트, 스크린샷, 모니터, 전자기기이(가) 표시된 사진

자동 생성된 설명

4. MiriStockApplicaion 우클릭 후 Application Run 버튼 클릭 하면 서버 실행

5. application.yml 환경 설정추가

*#default*spring:  
 profiles:  
 group:  
 local: local,oauthlocal  
 blue: blue,oauthprod  
 green: green,oauthprod  
 active: local  
---  
server:  
 port:   
  
spring:  
 config:  
 activate:  
 on-profile: local  
 datasource:  
 url :   
 username:   
 password:   
 driver-class-name :   
 jpa:  
 database : mysql  
 database-platform :   
 hibernate:  
 properties:  
 hibernate:  
 format\_sql : true  
 show\_sql : true  
 use\_sql\_comments: true  
 redis:  
 host:   
 port:   
 password:   
 mvc:  
 pathmatch:  
 matching-strategy: ant\_path\_matcher  
logging:  
 level:  
 root: info  
 org:  
 hibernate: info  
redirect:  
 url:   
springdoc:  
 version: '@project.version@'  
 swagger-ui:  
 path: /swagger-ui.html  
 groups-order: DESC  
 operationsSorter: method  
 disable-swagger-default-url: true  
 display-request-duration: true  
 api-docs:  
 path: /api-docs  
 show-actuator: true  
 default-consumes-media-type: application/json  
 default-produces-media-type: application/json  
 paths-to-match:  
 - /\*\*  
---  
server:  
 port:   
  
spring:  
 config:  
 activate:  
 on-profile: green  
 datasource:  
 url   
 username:   
 password:   
 driver-class-name :   
 jpa:  
 database : mysql  
 database-platform :   
 hibernate:  
 properties:  
 hibernate:  
 format\_sql : true  
 show\_sql : false  
 use\_sql\_comments: true  
 redis:  
 host: host.docker.internal  
 port:   
 password:   
 mvc:  
 pathmatch:  
 matching-strategy: ant\_path\_matcher  
logging:  
 level:  
 root: info  
 org:  
 hibernate: warn  
redirect:  
 url:   
---  
server:  
 port:   
  
spring:  
 config:  
 activate:  
 on-profile: blue  
 datasource:  
 url :   
 username:   
 password:   
 driver-class-name :   
 jpa:  
 database : mysql  
 database-platform :   
 hibernate:  
 properties:  
 hibernate:  
 format\_sql : true  
 show\_sql : false  
 use\_sql\_comments: true  
 redis:  
 host: host.docker.internal  
 port:   
 password:   
 mvc:  
 pathmatch:  
 matching-strategy: *ant\_path\_matcher*logging:  
 level:  
 root: info  
 org:  
 hibernate: warn  
redirect:  
 url:   
---  
spring:  
 config:  
 activate:  
 on-profile: oauthlocal  
 security:  
 oauth2:  
 client:  
 registration:  
 kakao:  
 client-id:   
 client-secret:   
 redirect-uri:   
 authorization-grant-type: authorization\_code  
 client-authentication-method: POST  
 client-name: Kakao  
 scope:  
 - profile\_nickname  
 - profile\_image  
 - account\_email  
 naver:  
 client-id :   
 client\_secret :   
 redirect-uri:   
 authorization-grant-type : authorization\_code  
 scope :  
 - name  
 - email  
 - nickname  
 provider:  
 kakao:  
 authorization-uri:   
 token-uri:   
 user-info-uri:   
 user-name-attribute: id  
 naver:  
 authorization-uri :   
 token-uri:   
 user-info-uri:   
 user-name-attribute: response  
---  
spring:  
 config:  
 activate:  
 on-profile: oauthprod  
 security:  
 oauth2:  
 client:  
 registration:  
 kakao:  
 client-id:   
 client-secret:   
 redirect-uri   
 authorization-grant-type: authorization\_code  
 client-authentication-method: POST  
 client-name: Kakao  
 scope:  
 - profile\_nickname  
 - profile\_image  
 - account\_email  
 naver:  
 client-id :  
 client\_secret :   
 redirect-uri:   
 authorization-grant-type : authorization\_code  
 scope :  
 - name  
 - email  
 - nickname  
 provider:  
 kakao:  
 authorization-uri:   
 token-uri:   
 user-info-uri:   
 user-name-attribute: id  
 naver:  
 authorization-uri :   
 token-uri:   
 user-info-uri:   
 user-name-attribute: response

**6. build.gradle 추가**

plugins {  
 id 'java'  
 id 'org.springframework.boot' version '2.7.8'  
 id 'io.spring.dependency-management' version '1.0.15.RELEASE'  
 id "com.ewerk.gradle.plugins.querydsl" version "1.0.10"  
}  
  
group = 'com.udteam'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '11'  
  
configurations {  
 compileOnly {  
 extendsFrom annotationProcessor  
 }  
}  
  
repositories {  
 mavenCentral()  
}  
  
dependencies {  
 implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'  
 implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
 implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.6.1'  
 implementation 'com.querydsl:querydsl-core:5.0.0'  
 implementation "com.querydsl:querydsl-jpa:5.0.0"  
 implementation "com.querydsl:querydsl-apt:5.0.0"  
 annotationProcessor "com.querydsl:querydsl-apt:5.0.0"  
  
 implementation 'org.springframework.boot:spring-boot-starter-security'  
 implementation 'org.springframework.boot:spring-boot-starter-web'  
*// implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.6.1'* compileOnly 'org.projectlombok:lombok'  
 developmentOnly 'org.springframework.boot:spring-boot-devtools'  
 *// https://mvnrepository.com/artifact/org.projectlombok/lombok* annotationProcessor group:'org.projectlombok',name:'lombok',version:'1.18.22'  
 compileOnly group: 'org.projectlombok', name: 'lombok', version: '1.18.24'  
 runtimeOnly 'com.mysql:mysql-connector-j'  
 testImplementation 'org.springframework.boot:spring-boot-starter-test'  
 testImplementation 'org.springframework.security:spring-security-test'  
 *// https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt* implementation group: 'io.jsonwebtoken', name: 'jjwt', version: '0.9.1'  
 *// https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui* implementation group: 'org.springdoc', name: 'springdoc-openapi-ui', version: '1.6.14'  
 implementation 'org.springframework.boot:spring-boot-starter-data-redis'  
 implementation 'org.springframework.boot:spring-boot-starter-actuator'  
 implementation group: 'org.jsoup', name: 'jsoup', version: '1.15.3'  
}  
  
tasks.named('test') {  
 useJUnitPlatform()  
}  
  
*def* querydslDir = "$buildDir/generated/querydsl"  
querydsl {  
 jpa = *true* querydslSourcesDir = querydslDir  
}  
sourceSets {  
 main.java.srcDir querydslDir  
}  
configurations {  
 querydsl.extendsFrom compileClasspath  
}  
compileQuerydsl {  
 options.annotationProcessorPath = configurations.querydsl  
}

**3. Database**

1. MySQL 워크벤치 8.0.30 실행

2. Connection 추가

텍스트이(가) 표시된 사진

자동 생성된 설명

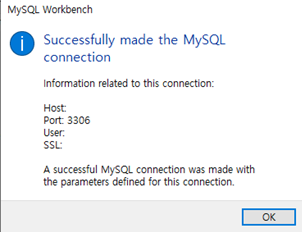
Connection name → 연결 이름

Hostname → AWS EC2 주소

username → 계정명

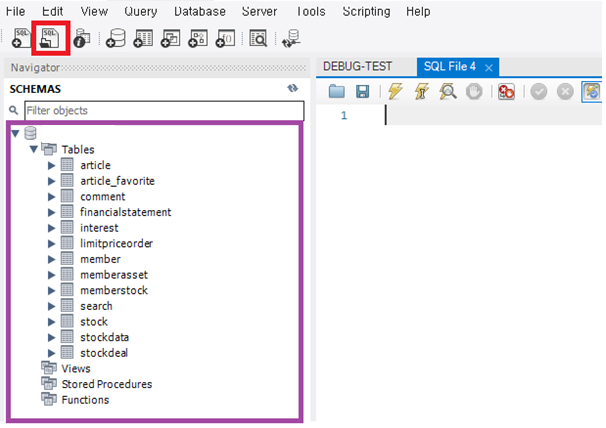
password → 비밀번호 입력

3. 정확한 정보를 입력하고 TestConnection을 시도했을 때 다음 창이 나타나면 성공



4. 연결 후 워크벤치 진입

5. 좌측상단 빨간색 버튼을 클릭한 후에 3.MiriStock\_dump.sql을 열기



6. Ctrl + a -> 번개버튼 클릭하면 MySQL에 데이터 입력

7. KIS 국내주식기간별시세(일/주/월/년)[v1\_국내주식-016] 데이터를 DB에 입력

[**https://apiportal.koreainvestment.com/apiservice/apiservice-sample#L\_apiservice-sample1**](https://apiportal.koreainvestment.com/apiservice/apiservice-sample%23L_apiservice-sample1)

위 링크의 샘플코드를 이용하여 MySQL에 데이터 입력

**2.4 OAuth 2.0 설정**

**카카오 Developer Redirect URL 설정**

- 로컬환경,개발환경,배포환경(ssl 적용,미적용) 으로 나눠서 등록

- 스프링 yml 파일 OAuth 설정에 등록할 Redirect URL과 맞춰서 등록

**네이버 Developer Redirect URL 설정**

- 네이버는 URL 칸이 적어서 로컬,배포 환경만 적용

- 마찬가지로 yml파일 설정의 Redirect URL 환경과 맞춰서 설정

**OAuth Local 설정**

security:  
 oauth2:  
 client:  
 registration:  
 kakao:  
 client-id:   
 client-secret:   
 redirect-uri:]  
 authorization-grant-type: authorization\_code  
 client-authentication-method: POST  
 client-name: Kakao  
 scope:  
 - profile\_nickname  
 - profile\_image  
 - account\_email  
 naver:  
 client-id   
 client\_secret :   
 redirect-uri:   
 authorization-grant-type : authorization\_code  
 scope :  
 - name  
 - email  
 - nickname  
 provider:  
 kakao:  
 authorization-uri:   
 token-uri:   
 user-info-uri:   
 user-name-attribute: id  
 naver:  
 authorization-uri :   
 token-uri:   
 user-info-uri:   
 user-name-attribute: response

- authorization-grant-type : 로그인 성공 시 받을 인증 타입

- client-authentication-method : 인증시 사용할 메서드

- scope : 반환받을 데이터

**3. Infra 구축 및 환경설정**

**3-1. 환경 구축 및 설치**

**AWS EC2 접속**

ssh -i [ec2-key.pem] [ID]@[Host

**Docker 설치**

1. apt 업데이트

sudo apt-get update

2. docker 설치

sudo apt-get install docker-ce docker-ce-cli containerd.io

3. docker 설치 후 확인

docker --version

**Docker-compose 설치**

1. docker-compose 원하는 버전 설치

sudo curl -L "<https://github.com/docker/compose/releases/download/1.25.0/docker-compose-$>(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

2. docker-compose 권한 설정

sudo chmod +x /usr/local/bin/docker-compose

3. 심볼릭 링크 설정

sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

4. docker-compose 설치 확인

docker-compose --version

**Nginx 설치 및 실행,중지,재시작**

1. nginx 설치

sudo apt-get install nginx

2. nginx 버전 확인

nginx -v

3. Nginx 실행/중지/재시작

sudo systemctl start|stop|restart nginx

**Mysql 설치 및 실행**

1. mysql 설치

sudo apt-get install mysql-server

2. mysql 실행

sudo systemctl start mysql

3. mysql 접속

sudo mysql -u root -p

**Redis 설치 및 실행**

1. redis 설치

sudo apt-get install redis-server

2. redis 버전 확인

redis-server --version

3. redis 실행

redis-cli

4. redis 비밀번호 설정 및 테스트

redis-cli

redis 127.0.0.1:PORT> CONFIG SET requirepass "비밀번호"

OK

redis 127.0.0.1:PORT> AUTH "비밀번호"

OK

redis 127.0.0.1:PORT> ping

pong

**3-2. Nginx 설정**

/etc/nginx/conf.d/application.conf

server {  
 listen 80;  
 server\_name [서버이름] [서버주소];  
   
 location / {  
 return 301 [서버리다이렉트 URL];  
}  
}  
   
   
 server {  
 listen 443 ssl http2;  
 *# listen [::]:80;* access\_log off;  
   
 *#server\_name localhost* server\_name [서버이름] [서버주소];  
   
 ssl\_certificate /etc/letsencrypt/live/[서버주소]/fullchain.pem;  
 ssl\_certificate\_key /etc/letsencrypt/live//[서버주소]/privkey.pem;  
   
 include /etc/nginx/conf.d/service-url.inc;  
 include /etc/nginx/conf.d/react\_url.inc;  
   
 location / {  
 proxy\_pass $react\_url;  
 proxy\_set\_header HOST $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
}  
   
 location /api/ {  
 if ($request\_uri ~\* "/api/(.\*)") {  
 proxy\_pass $service\_url/$1;  
}  
 proxy\_set\_header HOST $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
}  
}

**HTTPS 관련 설정 [Let’s Encrypt]**

1. certbot 설치

sudo add-apt-repository ppa:certbot/certbot

sudo apt-get update

sudo apt-get install python-certbot-nginx

2. 도메인주소의 인증 키(pem 파일) 생성

sudo certbot certonly --nginx -d [Domain Name]

3. 인증서확인

/etc/letsencrypt/live/[Domain Name]/fullchain.pem

/etc/letsencrypt/live/[Domain Name]privkey.pem

이 있는지 확인 후 nginx설정파일에 include

**3.3 Front-End Infra설정**

**docker-compose.yml**

version: "3"  
services:  
 miristockreact:  
 container\_name: miristockreact-blue  
 build:  
 dockerfile: Dockerfile *# dockerfile이름* context: /var/lib/jenkins/workspace/miristockpipelinereact/front *# 도커파일 위치 명시* environment:  
 REACT\_APP\_BASE\_UR: api  
 extra\_hosts:  
 - host.docker.internal:host-gateway  
 ports:  
 - 3001:3000 *# port 맵핑* stdin\_open: true  
networks:  
 default:  
 external:  
 name: service-network

Dockerfile

# 가져올 이미지를 정의

FROM node:16.18.0

# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를 뜻함)

COPY package.json .

# 명령어 실행 (의존성 설치)

RUN npm install

# 현재 디렉토리의 모든 파일을 도커 컨테이너의 워킹 디렉토리에 복사한다.

COPY . .

RUN npm run build

# 각각의 명령어들은 한줄 한줄씩 캐싱되어 실행된다.

# package.json의 내용은 자주 바뀌진 않을 거지만

# 소스 코드는 자주 바뀌는데

# npm install과 COPY . . 를 동시에 수행하면

# 소스 코드가 조금 달라질때도 항상 npm install을 수행해서 리소스가 낭비된다.

# 3000번 포트 노출

EXPOSE 3000

# npm start 스크립트 실행

CMD ["npm", "start"]

**3.4 Back-End Infra 설정**

Back-end Infra 설정

version: '3.1'  
  
services:  
 api:  
 build: /var/lib/jenkins/workspace/miristockpipeline/back  
 image: miristock  
 container\_name: miristock-green  
 environment:  
 SPRING\_DATASOURCE\_URL:   
 SPRING\_DATASOURCE\_USERNAME:   
 SPRING\_DATASOURCE\_PASSWORD:   
 SPRING\_PROFILES\_ACTIVE: green  
 SPRING\_REDIS\_HOST: host.docker.internal  
 SPRING\_REDIS\_PORT:   
 SPRING\_REDIS\_PASSWORD:   
 extra\_hosts:  
 - host.docker.internal:host-gateway  
 ports:  
 -   
  
networks:  
 default:  
 external:  
 name: service-network

Dockerfile

FROM azul/zulu-openjdk:11

ENV SPRING\_DATASOURCE\_URL [DATASOURCE\_URL]

ENV SPRING\_DATASOURCE\_USERNAME [USERNAME]

ENV SPRING\_DATASOURCE\_PASSWORD [패스워드]

ARG JAR\_FILE=build/libs/miristock-0.0.1-SNAPSHOT.jar

COPY ${JAR\_FILE} app.jar

ENTRYPOINT ["java","-jar","/app.jar"]

**4. 주요 사용 명령어**

**Docker**

docker ps //현재 실행중인 docker의 목록을 보여준다

docker container ls -al //존재하는 모든 컨테이너 목록을 보여준다

docker images //만들어진 이미지들을 보여준다

docker-compose up //docker-compose.yml에 설정된 내용들을 빌드,실행 시켜준다

docker-compose stop //실행중인 docker를 중지시킨다

docker-compose down //실행중인 docker를 중지시키고 이미지를 삭제한다

docker logs [Container ID | Container Name] //지정한 docker의 로그를 보여준다

docker exec -it [Container ID | Container Name] /bin/bash //실행중인 컨테이너를 쉘로 접속한다

**MySql**

show databases; // 존재하는 데이터베이스들을 출력한다

show tables; // 존재하는 테이블들을 출력한다

create database [DB Name] // DB를 생성한다

use [DB Name] // DB를 사용한다

create user [DB Name]@[Allow host] identified by [Password] // DB에 사용자 추가

grant all privileges on [DB Name].\* to [UserName]@[Allow host] identified by [password] // 유저에 권한을 추가한다

soruce [SQL file] // sql파일을 가져와 실행시킨다

**Redis**

AUTH [Password] // redis에 설정된 패스워드 입력

get [key] // 해당 key의 value값을 가져온다

set [key] [value] // redis에 key와 value 값을 설정한다

keys \* // redis에 존재하는 모든 키값을 가져온다