

# 포팅 메뉴얼

---

## 1 사용도구

협업 도구

배포 도구

설계 도구

개발 도구

## 2 개발 환경

2.1 Frontend

2.2 Backend

2.3 Server

2.4 Service

## 3 환경 변수

3.1 Backend

## 4 배포 및 서비스 실행

4.1 배포 포트

4.3 배포 과정

## 5 외부 서비스 연동

5.1 데이터베이스 연동(MySQL)

5.2 Redis 연동

5.2 외부 서비스 연동

## 1 사용도구

### 협업 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab, Git
- 커뮤니케이션 : Notion, Mattermost

### 배포 도구

- CI/CD : Jenkins, DockerHub, Docker, EC2

### 설계 도구

- 와이어프레임 : Figma
- ERD : ERD Cloud
- 요구사항 정의서 : Google Sheets

- 기능 명세서 : Google Sheets
- 간트차트 : Google Sheets

## 개발 도구

- **프론트엔드 IDE**
- IntelliJ : 2022.3.2 (Ultimate Edition)
- DataGrip : 2022.3.2 (Ultimate Edition)
- **CICD 터미널(Ubuntu..??)**

## 2 개발 환경

### 2.1 Frontend

```
"@babel/plugin-proposal-private-property-in-object": "^7.21.1",
"@emotion/react": "^11.11.3",
"@emotion/styled": "^11.11.0",
"@fortawesome/fontawesome-svg-core": "^6.5.1",
"@fortawesome/free-brands-svg-icons": "^6.5.1",
"@fortawesome/free-regular-svg-icons": "^6.5.1",
"@fortawesome/free-solid-svg-icons": "^6.5.1",
"@fortawesome/react-fontawesome": "^0.2.0",
"@material-ui/core": "^4.12.4",
"@material-ui/icons": "^4.11.3",
"@mui/icons-material": "^5.15.6",
"@mui/material": "^5.15.6",
"@nivo/bar": "^0.84.0",
"@nivo/core": "^0.84.0",
"@nivo/line": "^0.84.0",
"@nivo/pie": "^0.84.0",
"@react-three/drei": "^9.97.0",
"@react-three/fiber": "^8.15.16",
"@reduxjs/toolkit": "^2.0.1",
"@tanstack/react-query": "^5.18.1",
"@tanstack/react-query-devtools": "^5.18.1",
"@testing-library/jest-dom": "^5.16.1",
"@testing-library/react": "^13.0.0",
```

```
"@testing-library/user-event": "^13.5.0",
"axios": "^1.6.5",
"font-awesome": "^4.7.0",
"html2canvas": "^1.4.1",
"jotai": "^2.6.4",
"js-cookie": "^3.0.5",
"react": "^18.2.0",
"react-beautiful-dnd": "^13.1.1",
"react-big-calendar": "^1.8.6",
"react-cookie": "^7.0.2",
"react-dom": "^18.2.0",
"react-modal": "^3.16.1",
"react-redux": "^9.1.0",
"react-router-dom": "^6.21.3",
"react-scripts": "5.0.1",
"redux-persist": "^6.0.0",
"socket.io-client": "^4.7.4",
"styled-components": "^6.1.8",
"sweetalert2": "^11.10.5",
"three": "^0.161.0",
"web-vitals": "^2.1.4",
"workbox-background-sync": "^6.4.2",
"workbox-broadcast-update": "^6.4.2",
"workbox-cacheable-response": "^6.4.2",
"workbox-core": "^6.4.2",
"workbox-expiration": "^6.4.2",
"workbox-google-analytics": "^6.4.2",
"workbox-navigation-preload": "^6.4.2",
"workbox-precaching": "^6.4.2",
"workbox-range-requests": "^6.4.2",
"workbox-routing": "^6.4.2",
"workbox-strategies": "^6.4.2",
"workbox-streams": "^6.4.2",
"xlsx": "^0.18.5"
```

## 2.2 Backend

프로그램	버전
jvm	openjdk version "17.0.8.1" 2023-08-24
gradle	7.3.0
spring boot	3.2.2
spring security	6.2.1
jpa	6.4.1.FINAL
querydsl	5.0.0
mysql connector-j	8.3.0
lettuce	6.3.1.RELEASE
spring batch	5.1.0
spring cache	3.2.2
jjwt	0.11.5
spring cloud aws	2.2.6.RELEASE
nurigo sdk	4.2.7

## 2.3 Server

프로그램	버전
AWS EC2	CPU : ??, RAM : 16GB, OS : Ubuntu
AWS S3	

## 2.4 Service

프로그램	버전
MySQL	
Redis	
Docker	23.0.4
Ubuntu	Ubuntu 20.04 LTS
Jenkins	2.442

# 3 환경 변수

## 3.1 Backend

- react : .env (S10P12A306/Frontend/BID.env)

```
GENERATE_SOURCEMAP=false
REACT_APP_STU_API=https://i10a306.p.ssafy.io/student
REACT_APP_TCH_API=https://i10a306.p.ssafy.io/admin
```

- spring boot : application.yml (/src/main/resources에 위치)

```
server:
  servlet:
    encoding:
      charset: UTF-8
      force: true
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://i10a306.p.ssafy.io:3306/BID
    username: root
    password: olrlobt
  autoconfigure:
    exclude: org.springframework.boot.autoconfigure.security.
  data:
    redis:
      host: i10a306.p.ssafy.io
      port: 8998
  jpa:
    open-in-view: true
    hibernate:
      ddl-auto: update
      show-sql: false
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
  batch:
    job:
      enabled: false
    jdbc:
      initialize-schema: always
  main:
```

```

        allow-bean-definition-overriding: true
logging:
  level:
    org.hibernate.sql: debug
    org.springframework: info
jwt:
  expirationTime: 2592000000 # 30분
  refreshExpirationTime: 60480000 # 7일
  secret: 4d2gl3g4ADh546f4j67sxz5he4as6wz22r7edg353gv3sb2v223
cool-sms:
  apiKey: NCSZRYXPEY3PEBDS
  apiSecret: 5Q09UGFNEPYJRR2RXYRACCU3BJW7UTFI
  fromNumber: 010-5511-0625
  domain: https://api.coolsms.co.kr
cloud:
  aws:
    credentials:
      access-key: AKIA5EDPLAXHCZOVRAAF
      secret-key: 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
    s3:
      bucket: ssafya306
      region:
        static: ap-northeast-2

```

# 실시간 알림을 위한 한글인코딩

```

server:
  servlet:
    encoding:
      charset: UTF-8
      force: true

```

# 스프링 프레임워크가 동작할 수 있도록 하는 설정

```

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver # MySQL 드라이버
    url: jdbc:mysql://i10a306.p.ssafy.io:3306/[MySQL 콘솔을 통해 생성된 DB 이름]
    username: [MySQL 사용자 ID]
    password: [MySQL 사용자 PW]

```

```

data:
  redis:
    host: [redis 서버 IP]
    port: [redis 서버 접속포트]
jpa:
  hibernate:
    ddl-auto: update # 초기에 DB CREATE하기 위함, 최초 실행 이후0
    show-sql: false # 자동으로 SQL 생성해준 것을 콘솔에 띄우는 설정
    properties:
      hibernate:
        format_sql: false # 콘솔에 뜨는 SQL을 예쁘게 해주는 설정
        dialect: org.hibernate.dialect.MySQL8Dialect
batch:
  job:
    enabled: false
  jdbc:
    initialize-schema: always
  main:
    allow-bean-definition-overriding: true
logging:
  level:
    org.hibernate.sql: debug
    org.springframework: info
jwt:
  expirationTime: [엑세스토큰 만료기간]
  refreshExpirationTime: [리프레시토큰 만료기간]
  secret: [jwt 암호화]
cool-sms:
  apiKey: [cool-sms api 키]
  apiSecret: [cool-sms api 시크릿키]
  fromNumber: [cool-sms api 인증코드 송신 번호]
  domain: https://api.coolsms.co.kr
cloud:
  aws:
    credentials:
      access-key: [aws s3 액세스 키]
      secret-key: [aws s3 시크릿 키]
    s3:

```

```
bucket: [aws s3 버킷명]
region:
static: ap-northeast-2
```

## 4 배포 및 서비스 실행

### 4.1 배포 포트

#### BACK-END

- spring boot application(module-admin) : 8081:8081
- spring boot application(module-student) : 8082:8082

#### FRONT-END

- Nginx(React 빌드파일) : 8998:6379

#### NGINX (서버)

- 백엔드 서버 : 80:443
- 프론트엔드 서버 : 3000:3000
- 젠킨스 서버 : 8999:8080

### 4.3 배포 과정

#### 1. EC2 서버 세팅

```
# 서버시간 변경
ubuntu@ip-172-31-0-240:~$ sudo timedatectl set-timezone Asia/

# 변경한 서버 시간 올바른지 확인o
ubuntu@ip-172-31-0-240:~$ date
Sat Jan 27 18:10:22 KST 2024

# 미리 서버 변경
ubuntu@ip-172-31-0-240:~$ sudo vim /etc/apt/sources.list

# 위에서 열어준 파일의 ~.ubuntu.com(서버경로)를 모두 변경
:s/ap-northeast-2.ec2.archive.ubuntu.com/mirror.kakao.com/
```



```

# 패키지 업그레이드 및 업데이트
ubuntu@ip-172-31-0-240:~$ sudo apt update
ubuntu@ip-172-31-0-240:~$ sudo apt upgrade

# 가상 메모리 할당
# 현재 메모리 용량 확인(할당전)
ubuntu@ip-172-31-0-240:~$ free -h
               total        used        free      shared  buff/
Mem:           3.8Gi        171Mi        2.4Gi         0.0Ki
Swap:           0B           0B           0B

# swap 메모리 할당
ubuntu@ip-172-31-0-240:~$ sudo dd if=/dev/zero of=/swapfile b

# 스왑 파일에 대한 읽기 및 쓰기 권한 업데이트
ubuntu@ip-172-31-0-240:~$ sudo chmod 600 /swapfile

# Linux 스왑 영역 설정
ubuntu@ip-172-31-0-240:~$ sudo mkswap /swapfile
Setting up swapspace version 1, size = 2 GiB (2147479552 byte
no label, UUID=f68154c6-613a-45c2-aee6-711d73b08dd7

# 스왑 공간에 스왑 파일을 추가하여 스왑 파일을 즉시 사용할 수 있도록 만듦
ubuntu@ip-172-31-0-240:~$ sudo swapon /swapfile

# 절차 성공 여부 확인
ubuntu@ip-172-31-0-240:~$ sudo swapon -s
Filename                                Type              Size              Used              P
/swapfile                               file              2097148

# /etc/fstab 파일을 편집하여 부팅시 스왑 파일을 활성화
ubuntu@ip-172-31-0-240:~$ sudo vi /etc/fstab

# 편집기로 연 파일에 아래 한줄 추가
/swapfile swap swap defaults 0 0

# 할당 후 메모리 확인

```

```
ubuntu@ip-172-31-0-240:~$ free -h
```

	total	used	free	shared	buff
Mem:	3.8Gi	191Mi	290Mi	0.0Ki	
Swap:	2.0Gi	0B	2.0Gi		

## 2. Nginx, SSL 설정

```
# 방화벽 확인 및 허용
# UFW 활성화
ubuntu@ip-172-31-0-240:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation at your own risk.
Firewall is active and enabled on system startup

# 방화벽 확인(디폴트 22번 포트; ssh 열려있음)
ubuntu@ip-172-31-0-240:~$ sudo ufw status
Status: active
```

To	Action	From
--	-----	----
22	ALLOW	Anywhere

```

# 방화벽 허용(B:D 서비스 기준)
ubuntu@ip-172-31-0-240:~$ sudo ufw allow 80 # http
Rule added
Rule added (v6)
ubuntu@ip-172-31-0-240:~$ sudo ufw allow 443 # https
Rule added
Rule added (v6)
ubuntu@ip-172-31-0-240:~$ sudo ufw allow 8989 # gerrit
Rule added
Rule added (v6)
ubuntu@ip-172-31-0-240:~$ sudo ufw allow 9000 # jenkins
Rule added
Rule added (v6)
ubuntu@ip-172-31-0-240:~$ sudo ufw allow 8080 # spring-boot(b
Rule added
Rule added (v6)
ubuntu@ip-172-31-0-240:~$ sudo ufw allow 8081 # spring-boot(b

```

```

Rule added
Rule added (v6)
ubuntu@ip-172-31-0-240:~$ sudo ufw allow 8082 # spring-boot(b
Rule added
Rule added (v6)

# Nginx 설치
buntu@ip-172-31-0-240:~$ sudo apt install nginx -y

# NGINX 상태확인
ubuntu@ip-172-31-0-240:~$ sudo systemctl status nginx

# 아래 화면 올라오면 성공(active -> running)
• nginx.service - A high performance web server and a reverse
    Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
    Active: active (running) since Sat 2024-01-27 18:45:51 KST; 1min 11s ago
    Docs: man:nginx(8)
    Process: 14881 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master process 14881
    Process: 14882 ExecStart=/usr/sbin/nginx -g daemon on; master process 14881
    Main PID: 14977 (nginx)
    Tasks: 3 (limit: 4667)
    Memory: 3.9M
    CPU: 29ms
    CGroup: /system.slice/nginx.service
            └─14977 "nginx: master process /usr/sbin/nginx -t -q -g daemon on; master process 14881"
            └─14979 "nginx: worker process" "" "" "" "" "" ""
            └─14980 "nginx: worker process" "" "" "" "" "" ""

Jan 27 18:45:51 ip-172-31-0-240 systemd[1]: Starting A high performance web server and a reverse proxy engine: nginx:
Jan 27 18:45:51 ip-172-31-0-240 systemd[1]: Started A high performance web server and a reverse proxy engine: nginx:

# SSL 설정
# letsencrypt 설치
ubuntu@ip-172-31-0-240:~$ sudo apt-get install letsencrypt

# certbot 설치
ubuntu@ip-172-31-0-240:~$ sudo apt-get install certbot python3-certbot-nginx

```

```

# certbot - NGINX 연결
ubuntu@ip-172-31-0-240:~$ sudo certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log

# 이메일 입력
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): hgene0929@gmail.com
- - - - -

# 약관동의
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.3-September-21-2021.pdf
before you agree to the terms and conditions. Do you agree to the
terms and conditions? (Y)es/(N)o: Y
- - - - -

# 이메일 수신동의
Would you be willing, once your first certificate is successfully
issued, to allow EFF to share your email address with the Electronic Frontier
Foundation (EFF), a non-profit 501(c)(3) public charity? We'd like to
send you email about our work, including EFF news, campaigns, and ways
to support digital freedom.
- - - - -
(Y)es/(N)o: Y
Account registered.

# 도메인 입력
Please enter the domain name(s) you would like on your certificate.
(space separated) (Enter 'c' to cancel): i10{팀코드}.p.ssafy.io

# http 입력시 리다이렉트 여부
2

# SSL 설정 이후 Nginx 환경설정
~~~

```

## 2. Docker 설치 :

```
# Docker Repository 등록 및 docker-ce 패키지 설치
sudo apt-get update
&& apt-get -y install apt-transport-https ca-certificates curl
&& curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo ${ID})
&& add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo ${ID})"
&& apt-get update
&& apt-get -y install docker-ce

# docker 버전 확인
ubuntu@ip-172-31-0-240:~$ docker --version
Docker version 25.0.1, build 29cf629

# docker-compose 설치
ubuntu@ip-172-31-0-240:~$ sudo apt-get install docker-compose

# docker-compose 버전 확인
ubuntu@ip-172-31-0-240:~$ docker-compose --version
docker-compose version 1.29.2, build unknown

# sudo 없이 docker 명령어 사용하기
# 현재 사용자를 docker group 에 포함
ubuntu@ip-172-31-0-240:~$ sudo usermod -aG docker ${USER}

# 터미널 재시작 후 결과 확인(끝에 docker 가 있는지 확인)
ubuntu@ip-172-31-0-240:~$ id -nG
ubuntu adm dialout cdrom floppy sudo audio dip video plugdev
```

## 3. Docker 컨테이너 생성 후, Jenkins Image 다운로드 :

```
# Jenkins 컨테이너 생성 및 구동
# 시간 및 포트 설정: 시간은 현재 서울기준시, 포트는 9000으로 접근가능하도록
# /home/ubuntu/jenkins/ 아래를 jenkins 컨테이너 내부와 연결
# 소켓관련조사 더해보기 -> 이해하기
# docker-compose 관련 조사 더해보기 -> 이해하기
# 컨테이너 이름, 이미지 버전: jenkins라는 컨테이너, java17버전용 이미지
```

```
docker run -d --env JENKINS_OPTS=--httpPort=8080
-v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul -p 9000:
-v /home/ubuntu/jenkins:/var/jenkins_home
-v /var/run/docker.sock:/var/run/docker.sock
-v /usr/local/bin/docker-compose:/usr/local/bin/docker-compos
--name jenkins -u root jenkins/jenkins:jdk17
```

```
docker run -d --env JENKINS_OPTS=--httpPort=8080 -v /etc/loca
```

# 이미지가 없으므로 생성(pull)한 후에 실행시킴

Unable to find image 'jenkins/jenkins:jdk17' locally

lts: Pulling from jenkins/jenkins

1b13d4e1a46e: Pull complete

5303cfd924b5: Pull complete

902fe2af3265: Pull complete

866f59365203: Pull complete

da9419a1cfff4: Pull complete

0f760cf88b2d: Pull complete

e1f034047864: Pull complete

b7fd15023031: Pull complete

be03ab118c25: Pull complete

13230d8adc6e: Pull complete

52b66e48bb82: Pull complete

7a718fc8b5a4: Pull complete

Digest: sha256:a786794ea0a2429e65d67a88fe497cbd97a0d73d73a39f

Status: Downloaded newer image for jenkins/jenkins:jdk17

# 초기비번 기억(나중에 확인도 가능: Jenkins 초기 설정화면에 나타나는 경로  
9f7a26485f63a9bae1c3a4e8da5bf707621dc20dea4ccc83365edcd714b22

# 구동상태를 보기 위해 아래 명령어로 로그 확인

ubuntu@ip-172-31-0-240:~/jenkins-data\$ sudo docker logs jenkins

# docker ps : 현재 실행중인 컨테이너 목록(-a 옵션은 실행중이 아닌 컨테이너  
ubuntu@ip-172-31-0-240:~/jenkins-data\$ docker ps

CONTAINER ID	IMAGE	COMMAND
0aa8b6bd52a6	jenkins/jenkins:jdk17	"/usr/bin/tini -- /u..."

#### 4. Jenkins 접속 및 초기설정 (계정 생성, 필요한 플러그인 설치) :

- 웹브라우저의 `http://{EC2 도메인네임/IP주소}:{설정포트}` 로 접속.
- 로그인 확인 단계에서 초기 패스워드를 입력하고 계정 생성할 것.
  - 초기 패스워드는 제공되는 경로의 파일에 적혀있음(아래 이미지의 빨간색 글자).

#### Getting Started

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

- Jenkins 대시보드 → Jenkins 관리 → Plugins.

# ssh 커맨드 입력에 사용  
SSH Agent

# docker 이미지 생성에 사용  
Docker  
Docker Commons  
Docker Pipeline  
Docker API

# 웹훅을 통해 브랜치 merge request 이벤트 발생시 Jenkins 자동 빌드에  
Generic Webhook Trigger

# 타사 레포지토리 이용시 사용 (GitLab, Github 등)  
GitLab

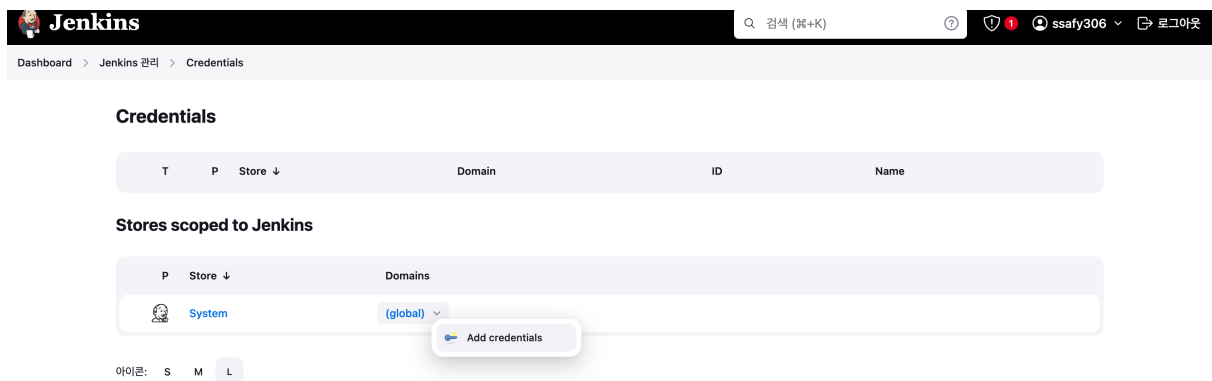
GitLab API  
GitLab Authentication  
GitHub Authentication

# Node.js 빌드시 사용  
NodeJS

## 5. 아이템에 대한 credentials 추가

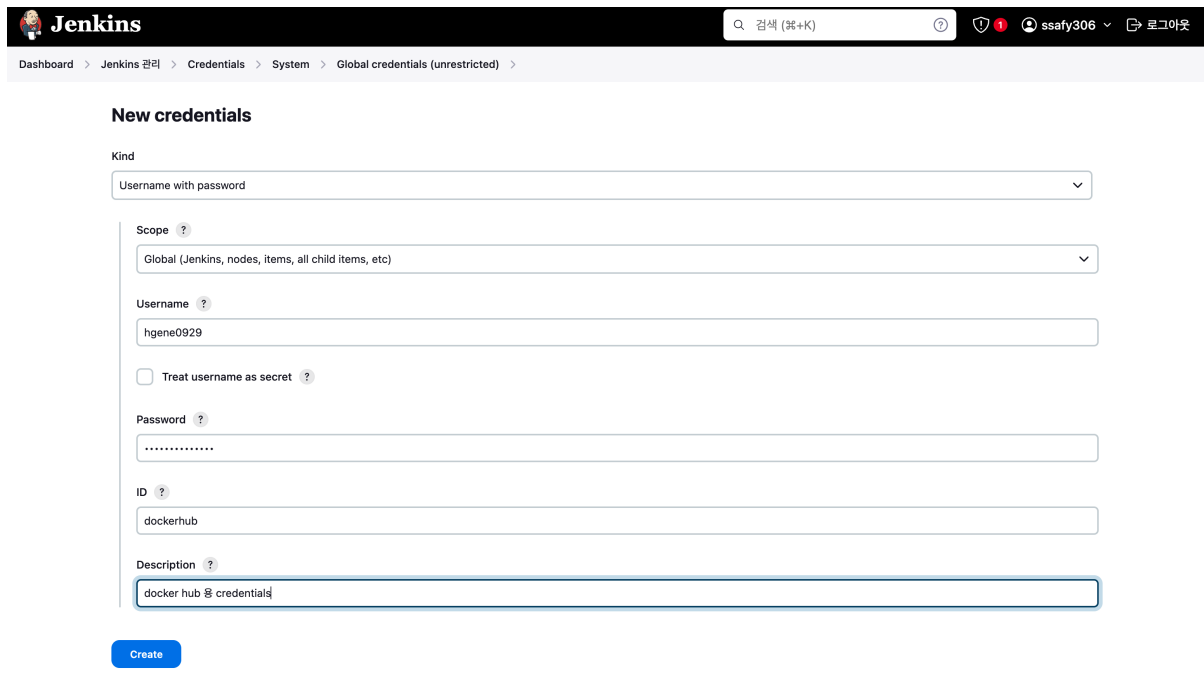
- Jenkins 대시보드 → Jenkins 관리 → credentials.

1. Docker Hub(Id, Password 등록).
2. Gitlab(Id, AccessToken 등록).
3. Ubuntu(EC2 .pem 키 등록, keygen도 가능).



*username with password*





**Jenkins** 검색 (Ctrl+K) [?] [🔒] [👤 ssafy306] [🚪 로그아웃]

Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >

### New credentials

Kind  
Username with password

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)

Username ?  
hgene0929

☐ Treat username as secret ?

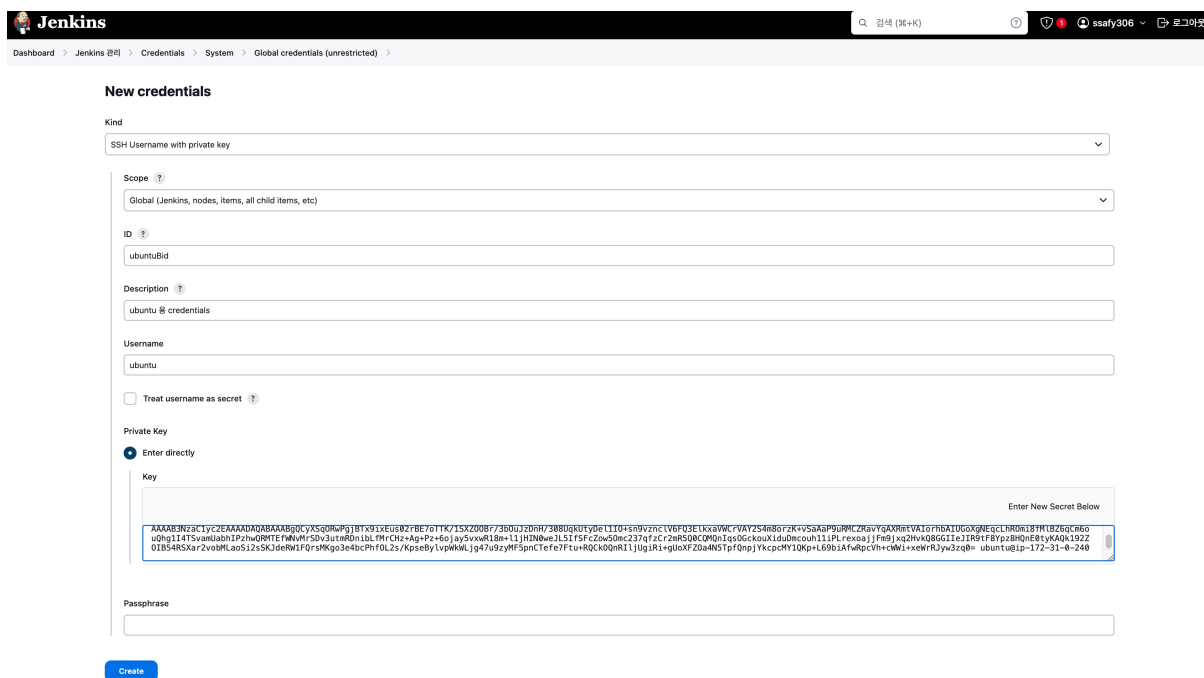
Password ?  
.....

ID ?  
dockerhub

Description ?  
docker hub 용 credentials

Create

## ssh username with private key



**Jenkins** 검색 (Ctrl+K) [?] [🔒] [👤 ssafy306] [🚪 로그아웃]

Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >

### New credentials

Kind  
SSH Username with private key

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)

ID ?  
ubuntuBid

Description ?  
ubuntu 8 credentials

Username  
ubuntu

☐ Treat username as secret ?

Private Key  
☒ Enter directly  
☐ Import from file  
 Key  
 Enter New Secret Below  
 AAAAB3NzaC1yc2EAAAADAQABAAQBAQCX5G8WpG18TX013Eu8Z7BE76TTK/15XZ008f/360UJ2DhU/388UgKUTy0e1101+sn9vzncV8F03ELkxawC7AYZ54886FzK+v5A8aP9uWMC28avTgX08tVAlomBA1UGoXgNEscL3R0818P1L8Z6Cm60  
 uQhg114TSvamb1hP7uWQRTEfW8Wf5D3utw8n1alFMCT8+AgPz+60/ay3vnuR18e+11jH2NwC15315fCzow50e237efZC+2nFS8NCQDn1sp06CcoU1d6dmcouh11PLr8x84jJ7m9/kq2hW4886G11eUJ2RFF87p8B8NE8tyKAO8192Z  
 01B54RSXar2vobMLao512sSKJdeW1F0rsMKgo3e4bcPh0LZs/KpseByLv9WkLjg47u9zyMF5pnCTe7FTu+ROCK0QhR11JugIR1+GuoXF20a4NSpF0npjYkcpRY1OKp+L69b1AfwRcpVh+cWl+xeNR3Jy3zq8= ubuntu@ip-172-31-0-240

Passphrase

Create

## keygen

```
# ubuntu 키젠 생성해서 환경변수에 등록
ubuntu@ip-172-31-0-240:~$ cd /
```

```
# keygen 생성
```

```
ubuntu@ip-172-31-0-240:/$ ssh -keygen -t rsa
```

```
# /home/ubuntu/.ssh/id_rsa.pub(디폴트 경로)에 키젠 값 가지고 있음
ubuntu@ip-172-31-0-240:/$ cat /home/ubuntu/.ssh/id_rsa.pub
```

## 6. Jenkins 컨테이너 접속 후 Docker 설정

```
# jenkins 컨테이너 root 계정으로 접속
ubuntu@ip-172-31-0-240:~/jenkins-data$ docker exec -it -u root

# docker-jenkins.service 등록
sudo vim /etc/systemd/system/docker-jenkins.service

# docker-jenkins.service 파일을 편집기로 연다음 다음 내용 작성
[Unit]
Description=docker-jenkins
Wants=docker.service
After=docker.service

[Service]
RemainAfterExit=yes
ExecStart=/usr/bin/docker start jenkins
ExecStop=/usr/bin/docker stop jenkins

[Install]
WantedBy=multi-user.target

# docker-jenkins 서비스 시작
sudo systemctl start docker-jenkins.service

# docker-jenkins 서비스 활성화
sudo systemctl enable docker-jenkins.service

# docker 서비스 시작
sudo systemctl start docker
```

```
# docker 서비스 활성화
sudo systemctl enable docker
```

## 7. Jenkins의 Item (CI/CD) 생성 및 구성 :

- *pipeline* :

```
pipeline {
    agent any

    environment {
        IMAGE_CORE = "olrlobt/bid-core"
        IMAGE_ADMIN = "olrlobt/bid-admin"
        IMAGE_STUDENT = "olrlobt/bid-student"
        IMAGE_FRONT = "olrlobt/bid-front"
        IMAGE_SOCKET = "olrlobt/bid-socket"

        BUILD_NUMBER = 'latest' // 실제 환경에서는 Jenkins의 빌드
        SERVER_ACCOUNT = 'ubuntu'
        SERVER_IP_ADD = 'i10a306.p.ssafy.io'
        WEBHOOK_URL = credentials('webhook')
        DOCKERHUB_CREDENTIALS = credentials('dockerhub_access
    }

    stages {
        stage('Preparation') {
            steps {
                git branch: 'release', credentialsId: 'gitlab
            }
        }

        stage('Docker Login') {
            steps {
                script {
                    sh "echo $DOCKERHUB_CREDENTIALS_PSW | doc
                }
            }
        }
    }
}
```

```

    }

    stage('Build and Deploy') {
        steps {
            script {
                // Docker 이미지 빌드
                sh'''
                docker build -t ${IMAGE_FRONT}:${BUILD_NUMBER}
                docker push ${IMAGE_FRONT}:${BUILD_NUMBER}
                docker build -t ${IMAGE_SOCKET}:${BUILD_NUMBER}
                docker push ${IMAGE_SOCKET}:${BUILD_NUMBER}

                cd BackEnd/bid/
                ./gradlew bid-core:clean bid-core:build
                docker build -t ${IMAGE_CORE}:${BUILD_NUMBER}
                docker push ${IMAGE_CORE}:${BUILD_NUMBER}

                ./gradlew bid-admin:clean bid-admin:build
                docker build -t ${IMAGE_ADMIN}:${BUILD_NUMBER}
                docker push ${IMAGE_ADMIN}:${BUILD_NUMBER}

                ./gradlew bid-student:clean bid-student:build
                docker build -t ${IMAGE_STUDENT}:${BUILD_NUMBER}
                docker push ${IMAGE_STUDENT}:${BUILD_NUMBER}

                docker-compose down
                docker-compose up -d
                '''
            }
        }
    }

    post {
        success {
            script {
                sh '''

```

```

        curl -i -X POST -H 'Content-Type: application.
        "attachments": [
            {
                "fallback": "BUILD SUCCESS !",
                "color": "#6badff",
                "title": "\n\nBuild Event Success
                "text": " ",
                "fields": [
                    {
                        "short": false,
                        "title": "Branch",
                        "value": "${GIT_BRANCH}"
                    },
                    {
                        "short": true,
                        "title": "Commit",
                        "value": "${GIT_COMMIT}"
                    }
                ]
            }
        ]
    }' $WEBHOOK_URL
'''
}

failure {
    script {
        sh '''
        curl -i -X POST -H 'Content-Type: application.
        "attachments": [
            {
                "fallback": "BUILD FAIL",
                "color": "#e78e77",
                "title": "\n\nBuild Event Fail",
                "text": " ",
                "fields": [
                    {
                        "short": false,

```



```
FROM openjdk:17-jdk
```

```
WORKDIR /app
```

```
COPY build/libs/bid-student-0.0.1-SNAPSHOT.jar app.jar
```

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

```
EXPOSE 8082
```

- *Dockerfile(frontend)* :

```
# 사용할 이미지 선택
```

```
FROM node:16-alpine as build
```

```
# 작업 디렉토리 설정
```

```
WORKDIR /app
```

```
# 컨테이너 내부로 package.json 파일들을 복사
```

```
COPY package*.json ./
```

```
# 명령어 실행 (의존성 설치)
```

```
RUN yarn install --network-timeout 1000000
```

```
COPY . .
```

```
#yarn build
```

```
RUN yarn build
```

```
# prod environment
```

```
FROM nginx:stable-alpine
```

```
# 이전 빌드 단계에서 빌드한 결과물을 /usr/share/nginx/html으로 복사
```

```
COPY --from=build /app/build /usr/share/nginx/html
```

```
# 기본 nginx 설정 파일을 삭제
```

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

```
# custom 설정파일을 컨테이너 내부로 복사
```

```
COPY nginx.conf /etc/nginx/conf.d
```

```
# 연결할 포트번호
EXPOSE 443

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

- *Dockerfile(FE - websocket)* :

```
# Node.js 공식 이미지를 기반으로 합니다.
FROM node:16-alpine as build

# 앱 디렉터리 생성
WORKDIR /app

# 앱 의존성 설치
# package.json과 package-lock.json을 모두 복사합니다.
COPY package*.json ./

RUN yarn install --network-timeout 1000000

COPY ./ ./

# 앱이 3001 포트에서 실행됨을 알립니다.
EXPOSE 3001

# 앱 실행
CMD ["node", "index.js"]
```

## 8. Nginx 구성 :

- *default* :

```
server {
    listen 80;
    server_name i10a306.p.ssafy.io www.i10a306.p.ssafy.io;
    return 301 https://$server_name$request_uri; # 모든 HTTP 요청을 HTTPS로 리다이렉트
}

server {
```



```

listen 443 ssl;
server_name i10a306.p.ssafy.io www.i10a306.p.ssafy.io;

access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log;

ssl_certificate /etc/letsencrypt/live/i10a306.p.ssafy.io/
ssl_certificate_key /etc/letsencrypt/live/i10a306.p.ssafy

ssl_prefer_server_ciphers on;
ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES1
ssl_protocols TLSv1.2 TLSv1.3;
ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;

    location ^~ /admin {
        rewrite ^/admin/(.*)$ /$1 break;
        proxy_pass http://localhost:8081; # Docker 컨테이너
        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 $scheme;
        proxy_buffering off;
    }
    location ^~ /student {
        rewrite ^/student/(.*)$ $1 break;
        proxy_pass http://localhost:8082; # Docker 컨테이너
        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;
    }

    location / {
        proxy_pass https://localhost:3000; # Docker 컨테이
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;

```



```

        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;
    }
}

server {
    listen 3001 ssl;
    server_name i10a306.p.ssafy.io www.i10a306.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i10a306.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/i10a306.p.ssafy

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers "ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES2
}

```

## 5 외부 서비스 연동

### 5.1 데이터베이스 연동(MySQL)

CLI & GUI(DataGrip) 공통

0. 사전작업 :

- EC2 서버 내부에 Docker가 설치되어 있어야 함.
- spring boot 애플리케이션의 application.yml 파일 내부 `spring.datasource.*` 아래의 접속정보 반영.

1. MySQL Docker Image 다운로드 :

```

# 가장 최근 MySQL 버전 다운로드
$sudo docker pull mysql

# 특정 버전 지정 MySQL 버전 다운로드
$sudo docker pull mysql:???

```

## 2. 다운로드된 Docker Image 확인 :

```
$sudo docker images
```

다운로드된 mysql docker image 캡처 첨부

## 3. MySQL Docker 컨테이너 생성 & 실행 :

```
$sudo docker run --name [컨테이너명]  
-e MYSQL_ROOT_PASSWORD=[패스워드]  
-d -p 3306:3306 mysql:???
```

## 4. Docker 컨테이너 리스트 확인 :

```
$sudo docker ps -a
```

컨테이너 목록 결과 캡처 화면

## MySQL CLI

### 1. EC2 MySQL Docker 컨테이너 및 MySQL 콘솔 접속 :

- EC2 내부에서 실행중인 MySQL에 접속.
- MySQL 접속정보 (root 로그인) 입력.

```
$sudo docker exec -it mysql-container bash
```

```
-bash $ mysql -u root -p  
Enter password: # 패스워드 입력
```

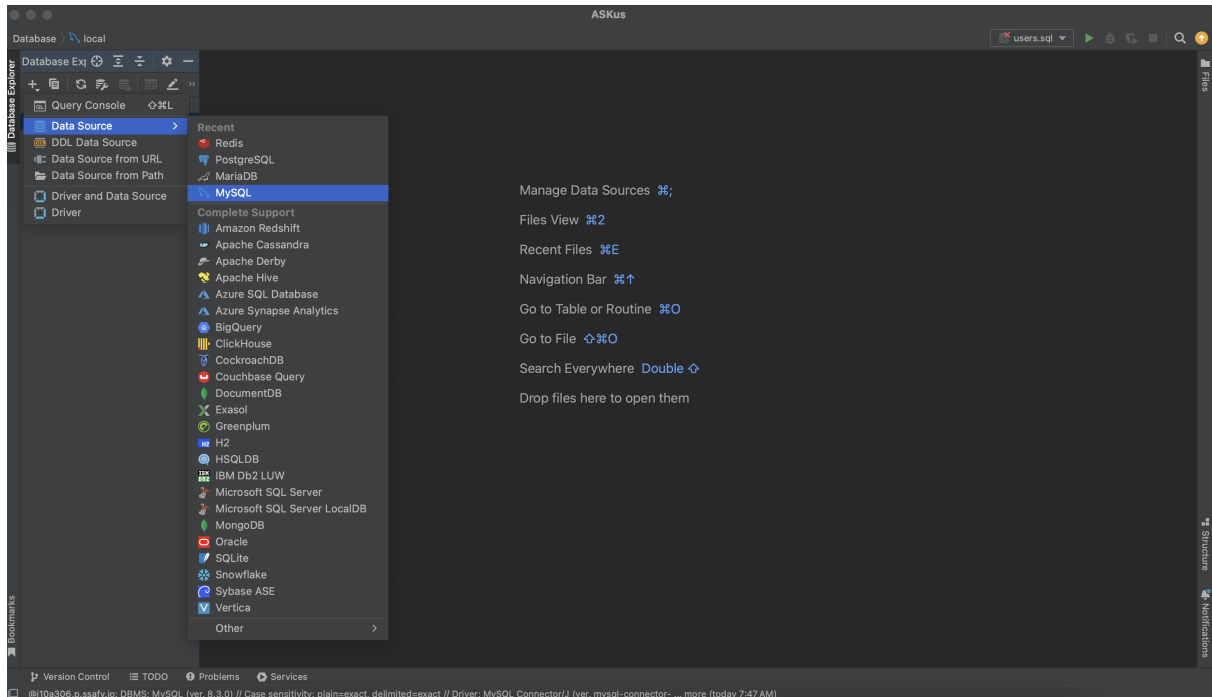
### 2. MySQL 콘솔을 통해 application.yml 에 작성해둔 url의 포트 뒷부분(DB명)과 동일한 이름의 데이터베이스 생성.

```
create database [DB명];
```

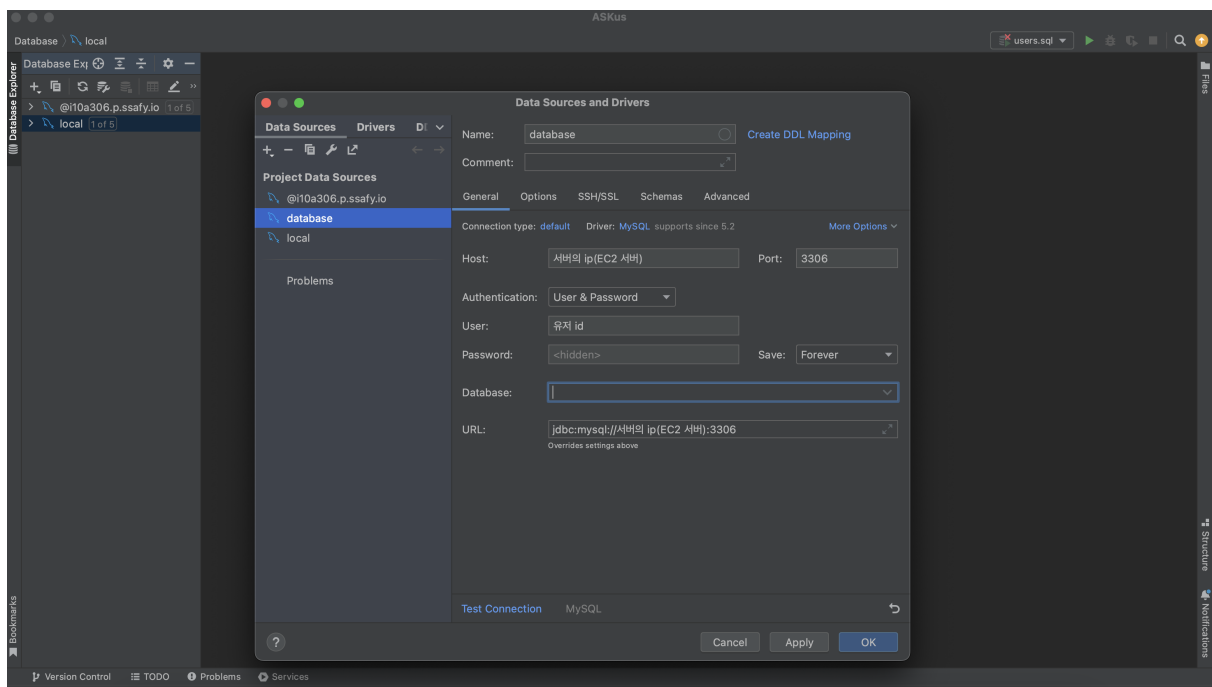
## MySQL GUI(DataGrip)

### 1. EC2 MySQL Docker 컨테이너 접속 및 콘솔 접속.

- 좌측 상단의 + 버튼을 클릭 후 Data Source > MySQL 클릭.



- 아래 이미지의 접속정보를 application.yml 과 동일하게 작성하고 OK.



2. MySQL 콘솔을 통해 application.yml 에 작성해둔 url의 포트 뒷부분(DB명)과 동일한 이름의 데이터베이스 생성.

```
create database [DB명];
```

## 5.2 Redis 연동

### 0. 사전작업 :

- EC2 서버 내부에 Docker가 설치되어 있어야 함.
- spring boot 애플리케이션의 application.yml 파일 내부 `spring.datasource.*` 아래의 접속정보 반영.

### 1. Redis Docker Image 다운로드 :

```
docker pull redis
```

### 2. 다운로드된 Docker Image 확인 :

```
docker images
```

### 3. Redis 컨테이너 실행 :


```
docker run -p 6379:6379 --name [컨테이너명] -d redis
```

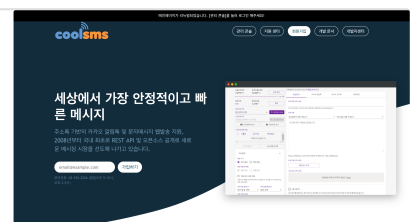
## 5.2 외부 서비스 연동

*cool-sms api : 전화번호 인증코드 전송*

1. 아래의 링크로 접속 후, 계정 생성.
2. 우측 상단 메뉴바의 개발/연동 > API Key 관리 > API 키, Secret 키 발급.
3. application.yml의 `cool-sms.*`와 알맞게 정보 입력.


세상에서 가장 안정적이고 빠른 메시지 발송 플랫폼 - 쿨에스엠에스  
카카오 알림톡 및 문자메시지 연동 발송을 지원해 드립니다.

 <https://coolsms.co.kr/>



### *aws s3 bucket : 이미지 파일 저장*

1. 아래의 링크로 접속 후, 계정 생성 및 로그인.
2. AWS Console > S3 > 버킷 > 버킷 만들기.
  - 버킷 생성시, 액세스 차단 설정을 해제.
3. 사용자 추가 및 접근 권한 부여.
  - 권한 설정시 직접 정책 연결 > AmazonS3FullAccess 선택.
4. AWS Console > IAM > 액세스 관리자 > 사용자 > 생성한 사용자 이름 클릭 > 보안 자격 증명 > 액세스 키 만들기.
5. application.yml의 `cloud.aws.credentials.*` 와 알맞게 정보 입력.

 <https://ap-northeast-2.console.aws.amazon.com/console/home?region=ap-northeast-2>