

**A303 : 마시멜로 일기**

삼성 소프트웨어 청년 아카데미 서울캠퍼스 7기

자율 프로젝트: 2022/11/ ~ 2022/11/21

**포팅 매뉴얼**

담당 컨설턴트: 오형남

최윤영(팀장), 김현지, 박현영, 유지슬, 이도연, 전상현

**<목차>**

1. 기술 스택
2. Gitignore 파일
3. 빌드 상세 내용
4. 배포 특이사항
5. DB 계정
6. 외부 서비스

나도 몰랐던 내 감정을 알려주는 마시멜로 일기. 하루를 기록하면 당신의 감정을 분석해드립니다. 매일 일기를 쓰고 행복한 기억을 돌아보며, 달콤하고 귀여운 마시멜로를 모아보세요!

**1. 기술 스택**

가. 이슈 관리: Jira

나. 형상 관리 : Gitlab

다. 커뮤니케이션 : MatterMost, Notion

라. 개발 환경

1) OS : WINDOW 10

2) IDE

가) IntelliJ IDEA 2022.2.2

나) Visual Studio Code 1.70.0

3) DataBase

가) MySQL 8.0.30

나) Redis 5.0.7

다) MongoDB 4.4.17

4) Server : AWS EC2

가) Ubuntu 20.04 LTS

나) Xshell 7.0.0025

다) Nginx

마. 상세

1) Backend

가) Gradle 7.5

나) Java 11.0.17 (corretto-11)

2) Frontend

가) Nodejs LTS 16.16.0

나) Chocolatey 1.2.0

다) Android Studio Dolphin | 2021.3.1 Patch 1

라) SDK 31

**2. .gitignore 파일**

가) BackEnd(Spring): *application-sescure.properties* 파일

1) 저장 위치

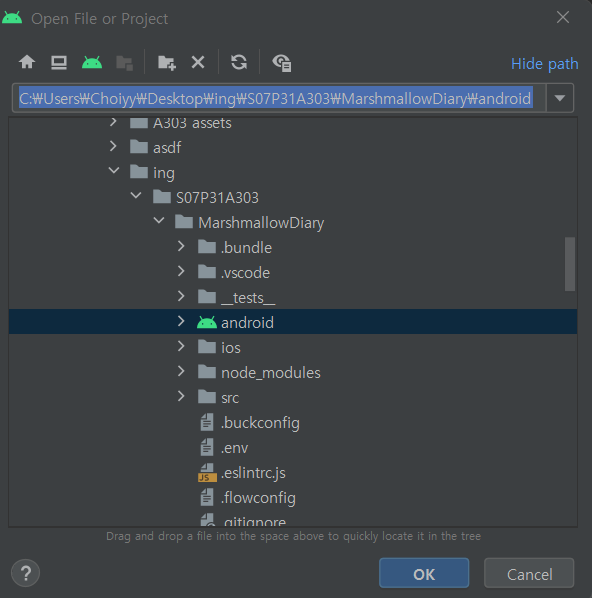
[project root directory]/back/src/main/resources

2) 파일 내용

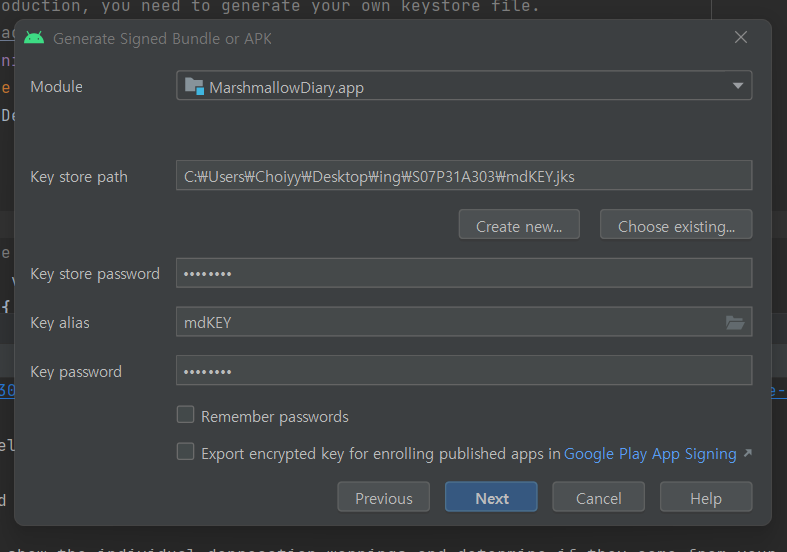
|  |
| --- |
| server.port=9090  *# localhost*  *# spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver*  *# spring.datasource.url=jdbc:mysql://localhost:3306/marshmallow?useUniCode=yes&characterEncoding=UTF-8&serverTimezone=Asia/Seoul*  *# spring.datasource.username=root*  *# spring.datasource.password=ssafy*  *# 리모트*  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  spring.datasource.url=jdbc:mysql://k7a303.p.ssafy.io:3306/marshmallow?useUniCode=yes&characterEncoding=UTF-8&serverTimezone=Asia/Seoul  spring.datasource.username=maro  spring.datasource.password=maro1229  spring.jwt.secretKey=marshmallow  cloud.aws.credentials.access-key=AKIA3P72UB7RH6IULLFG  cloud.aws.credentials.secret-key=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  cloud.aws.stack.auto=false  cloud.aws.s3.bucket=marshmallow-bucket  cloud.aws.region.static=ap-northeast-2  cloud.aws.s3.bucket.url=https://s3.ap-northeast-2.amazonaws.com/marshmallow-bucket  clova.id=pnk8ieleu4  clova.key=44oU5QJ81KCHIOlBsL8Oh8hBLl24idaLNj5Q2oEc  spring.redis.port=6379  spring.redis.host=k7a303.p.ssafy.io  spring.redis.password=maro1229 |

**3. 빌드 상세 내용**

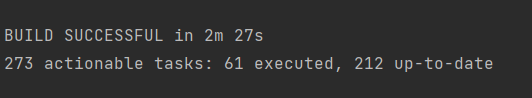
가) FrontEnd 빌드하기

 1) 빌드할 소스코드를 Android Studio에서 열기

2) 사진과 같이 정보 입력

- Key Store Path: jks 파일 위치(컴퓨터의 키 경로)  
 - Key store password & Key password: maro1229

3) next > release > finish

 4) 빌드 성공 시 다음과 같은 메시지와 함께 MarshmallowDiary → android → app →  
 release에 .aab 파일 생성됨(생성된 .aab 파일명을 출시버전명과 동일하게 해야 함)

나) BackEnd 빌드하기

1) [project root directory]/back 위치로 이동

2) 빌드

|  |
| --- |
| ./gradlew clean build -x test |

3) 생성된 jar 파일 실행

|  |
| --- |
| java -jar build/libs/\*.jar |

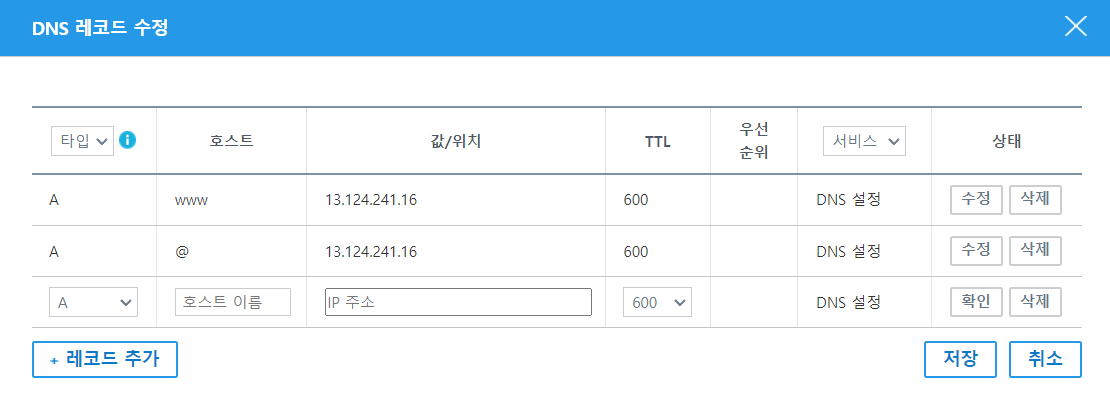
**4. 배포 특이 사항**

가. 가비아 도메인 IP 등록

1) 퍼블릭 IP 확인

|  |
| --- |
| curl http://checkip.amazonaws.com |

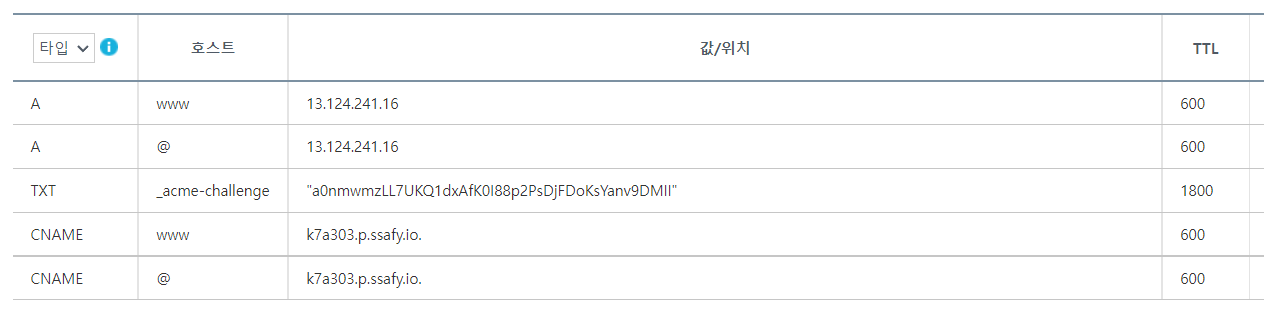
2) My 가비아 > 관리 > DNS 정보 > 도메인 연결 설정 > 레코드 수정



나. DNS 레코드 설정

1) 도메인 소유권 확인

|  |
| --- |
| sudo certbot certonly -d marshmallowdiary.com --manual --preferred-challenges dns |

 2) DNS 레코드 설정 > 도메인 연결 설정 > Gsuite 선택 > 다음과 같이 레코드 추가

다. https 적용을 위한 SSL 인증서 발급

1) Nginx 및 Certbot 설치

|  |
| --- |
| *# 저장소 설정과 업데이트 진행*  sudo apt-get update  sudo apt-get install software-properties-common  sudo add-apt-repository universe  sudo add-apt-repository ppa:certbot/certbot  sudo apt-get update  *# certbot 설치*  sudo apt-get install certbot python3-certbot-nginx |

2) Nginx 설정

|  |
| --- |
| sudo vi /etc/nginx/sites-available/marshmallowdiary |

도메인 정보를 기입

|  |
| --- |
| **server** {  **listen** **80**;  **server\_name** marshmallowdiary.com;  **location** /{  **proxy\_pass** http://marshmallow.com:9090;  }  } |

3) SSL 인증서 발급

|  |
| --- |
| sudo certbot --nginx -d marshmallowdiary.com |

4) openssl로 dhparam.pem키 생성

|  |
| --- |
| openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048 |

5) Nginx 설정 파일 수정

|  |
| --- |
| **server** {  **listen** **443** default\_server ssl;  **listen** [::]:443 default\_server ssl ipv6only=on;  **server\_name** marshmallowdiary.com;  **access\_log** /var/log/nginx/access.log;  **ssl\_certificate** /etc/letsencrypt/live/marshmallowdiary.com/fullchain.pem;  **ssl\_certificate\_key** /etc/letsencrypt/live/marshmallowdiary.com/privkey.pem;  **ssl\_dhparam** "/etc/ssl/certs/dhparam.pem";  **ssl\_session\_cache** shared:SSL:1m;  **ssl\_session\_timeout** **5m**;  **ssl\_ciphers** HIGH:MEDIUM:!SSLv2:!PSK:!SRP:!ADH:!AECDH;  **ssl\_prefer\_server\_ciphers** **on**;  **location** /api {  **proxy\_pass** http://marshmallowdiary.com:9090;  **proxy\_http\_version** **1**.1;  **proxy\_set\_header** Connection "";  **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;  **proxy\_set\_header** X-Forwarded-Host $host;  **proxy\_set\_header** X-Forwarded-Port $server\_port;  }  **location** / {  **root** /usr/share/nginx/html;  **index** index.html index.htm index.nginx-debian.html;  **try\_files** $uri $uri/ /index.html;  }  }  **server** {  **listen** **80**;  **listen** [::]:80 ipv6only=on default\_server;  **server\_name** marshmallowdiary.com;  **return** **301** https://marshmallowdiary.com$request-uri;  } |

6) Nginx 서버 재실행

|  |
| --- |
| sudo service nginx start  sudo service nginx stop  sudo service nginx restart  sudo systemctl status nginx.service |

라. 편리한 배포를 위한 Docker 설치

1) 사전 패키지 설치

|  |
| --- |
| sudo apt update  sudo apt-get install -y ca-certificates **\**  curl **\**  software-properties-common **\**  apt-transport-https **\**  gnupg **\**  lsb-release |

2) gpg키 다운로드

|  |
| --- |
| sudo mkdir -p /etc/apt/keyrings  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  echo \  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null |

3) docker 설치

|  |
| --- |
| sudo apt update  sudo apt install docker-ce docker-ce-cli containerd.io docker-compose |

마. dockerfile 작성

1) 저장 위치

[project root directory]/back

2) 파일 내용

|  |
| --- |
| FROM openjdk:11-jdk as builder  WORKDIR application  ARG JAR\_FILE=build/libs/\*.jar  COPY ${JAR\_FILE} ./  RUN java -Djarmode=layertools -jar marshmallow.jar extract # --destination  FROM openjdk:11-jdk  WORKDIR application  ENV port 9090  ENV spring.profiles.active local  COPY --from=builder application/dependencies/ ./  COPY --from=builder application/spring-boot-loader/ ./  COPY --from=builder application/snapshot-dependencies/ ./  COPY --from=builder application/application/ ./  ENTRYPOINT ["java", "org.springframework.boot.loader.JarLauncher"] |

바. 젠킨스 설치

1) 자바 설치 여부 확인

|  |
| --- |
| java -version |

1-2) 설치되어 있지 않다면 자바 설치

|  |
| --- |
| sudo apt update  sudo apt install openjdk-11-jdk -y |

2) 젠킨스 설치

|  |
| --- |
| wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -  sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \  /etc/apt/sources.list.d/jenkins.list'  sudo apt-get update  sudo apt-get install jenkins |

3) 상태 확인

|  |
| --- |
| systemctl status jenkins |

사. 젠킨스 프로젝트 생성 및 Gitlab 연결

1) 포트 열기

|  |
| --- |
| sudo ufw allow 8080 |

2) {퍼블릭IP/도메인주소}:8080으로 접속하여 다음 명령어의 결과 비밀번호 입력

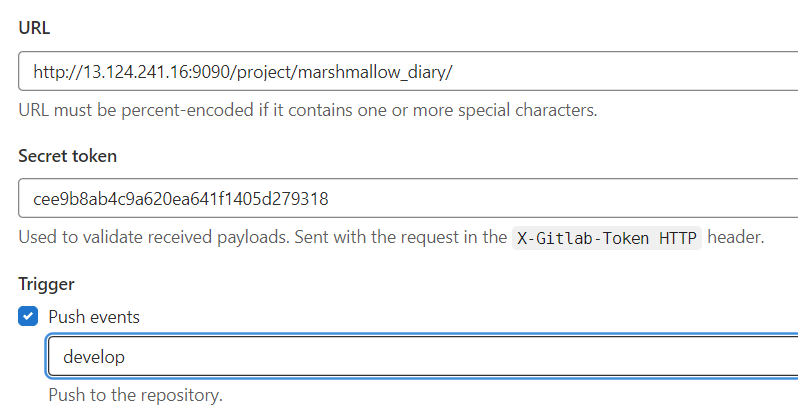
|  |
| --- |
| sudo cat /var/lib/jenkins/secrets/initialAdminPassword |

3) 계정 설정 및 필요한 플러그인 설치

- gitlab, Generic Webhook Trigger, Gitlab API, Gitlab Authentication  
 - Docker, Docker Commons, Docker Pipeline, Docker API  
 - Publish Over SSH

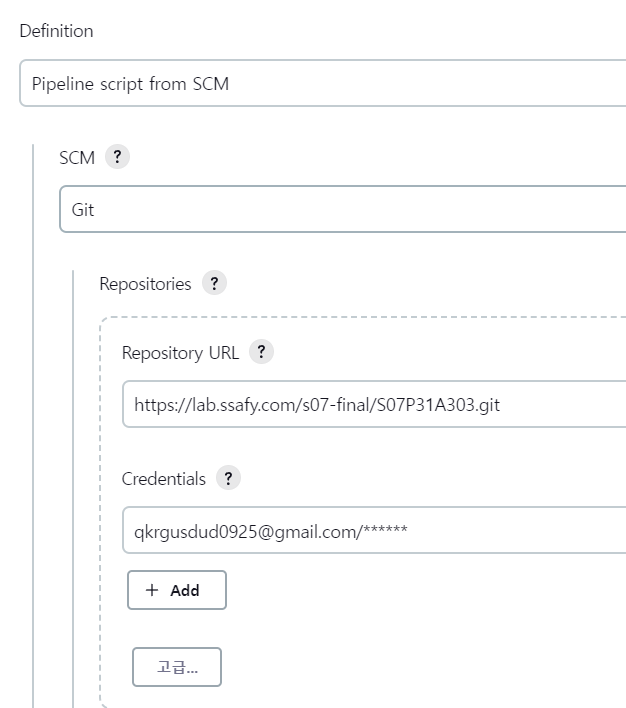
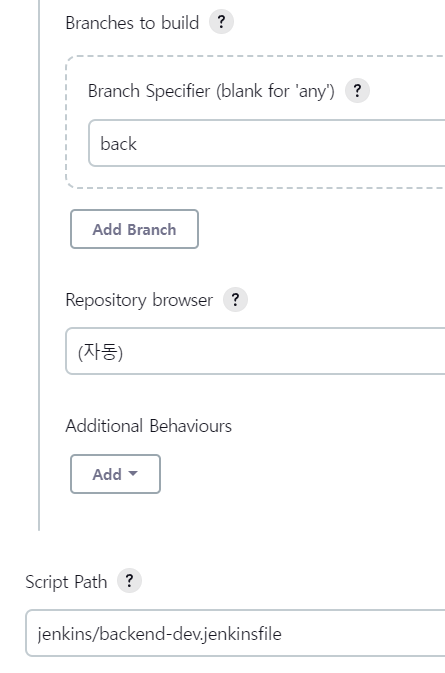
4) Github hook으로 pipeline 프로젝트 생성

5) 고급 > Generate 로 시크릿 토큰 저장(gitlab 연결 시 사용)

 6) Gitlab Webhook 설정

7) Jenkins관리 > 시스템 설정 > Global properties > Environment variable 환경변수 등록

8) 구성 > Advanced Prokect Options에서 Pipeline script from SCM 선택



아. jenkinsfile 작성

1) 저장 위치

[project root directory]/jenkins

2) 파일 내용

|  |
| --- |
| node {  stage ("git checkout") {  git branch: "back", credentialsId: "${env.CREDENTIAL\_ID}", url: "https://lab.ssafy.com/s07-final/S07P31A303.git"  }  stage ("image prune") {  sh "sudo docker image prune -a --force"  }  stage("kill previous container") {  sh "sudo docker rm -f ${env.SPRING\_CONTAINER\_NAME}"  }    stage("build jar") {  dir("back"){  sh "chmod +x ./gradlew"  sh "sudo ./gradlew clean build -x test"  }  }  stage ("save spring image") {  sh "sudo docker build --no-cache -t ${env.SPRING\_IMAGE\_NAME} back"  sh "sudo docker save ${env.SPRING\_IMAGE\_NAME} > marshmallow.tar"  }  stage("deploy"){  sh "sudo docker load -i marshmallow.tar"  sh "sudo docker run -dit -p 80:80 -p ${env.SPRING\_PORT}:${env.SPRING\_PORT} --name ${env.SPRING\_CONTAINER\_NAME} ${env.SPRING\_IMAGE\_NAME}"  script {  def NONE\_CONTAINERS = sh(script: "sudo docker images -f \"dangling=true\" -q", returnStdout:true)  // echo NONE\_CONTAINERS  if(!NONE\_CONTAINERS.isEmpty()){  // echo "containers need to be deleted"  sh "sudo docker rmi -f \$(sudo docker images -f \"dangling=true\" -q)"  }  }  }  } |

**5. DB 계정 설정**

가) 접속 정보

1) Host명: k7a303.p.ssafy.io

2) root 계정 비밀번호: marosiro

3) user 계정 정보

a) ID: maro

b) PWD: maro1229

나) MySQL 설치 및 설정

1) 초기 설정

|  |
| --- |
| *# EC2 접속*  ssh -i K7A303T.pem ubuntu@k7a303.p.ssafy.io  sudo ufw allow ssh  sudo ufw enable  *# 초기 설정*  sudo apt-get update *# 패키지 매니저 업데이트*  sudo apt-get install mysql-server *# mysql 설치*  sudo ufw allow mysql *# 외부 접속 기능 설정(포트 3306 오픈)*  sudo systemctl start mysql *# mysql 실행*  sudo systemctl enable mysql *# mysql 서버 재시작 시 mysql 자동 재시작* |

2) MySQL 접속 및 root 비밀번호 변경

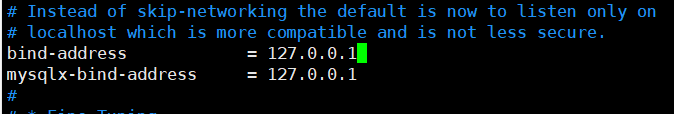
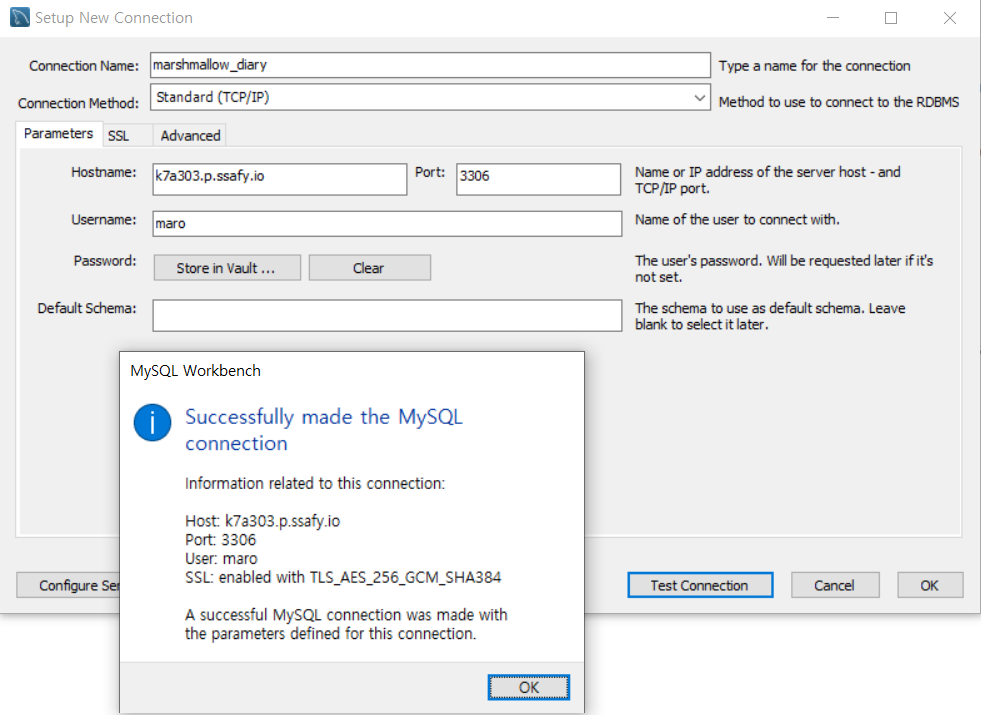
|  |
| --- |
| sudo /usr/bin/mysql -u root -p *# 비밀번호: marosiro*  use mysql;  mysql> **select** Host, User, authentication\_string from user;  mysql> **select** Host, User, authentication\_string from user; |

3) user 계정 생성 및 권한 부여

|  |
| --- |
| mysql> create user 'maro'@'%' identified by 'maro1229';  mysql> grant all privileges on \*.\* to maro@'%';  mysql> update user set Host='%' where User='root';  mysql> FLUSH PRIVILEGES;  mysql> exit  Bye |

4) 외부 접속 허용을 위해 mysqld.cnf 파일 수정

|  |
| --- |
| sudo su # root 권한 부여  cd /etc/mysql/mysql.conf.d # mysqld.cnf 파일이 있는 디렉터리로 이동  vi mysqld.cnf # 파일 실행  # bind-address 부분을 찾아 0.0.0.0으로 변경  service mysql restart # mysql 서버 재실행 |

 5) MySQL Workbench에서 접속(로컬 컴퓨터)

다) Redis 설치 및 설정

1) Redis 설치

|  |
| --- |
| # 우분투에 redis 설치  wget https://download.redis.io/redis-stable.tar.gz  tar xzf redis-stable.tar.gz  cd redis-stable  sudo make install  # 빌드 완료 후 확인  cd src  # conf 파일 상관없이 실행  ./redis-server  # 작성한 conf 파일을 기반으로 redis 실행  sudo ./redis-server /etc/redis/redis.conf  # redis 상태 확인  systemctl status redis-server.service  # src 디렉토리에서 실행  ./redis-cli  # 설정한 비밀번호를 입력해야 접근 가능  AUTH {password} |

2) conf 파일 관리

|  |
| --- |
| # 우분투 redis 설정파일 위치  sudo vi /etc/redis/redis.conf  #### redis.conf 기본 설정하기####  # redis 접속 허용 IP 설정  bind [IP] # ex: bind 127.0.0.1  # redis 허용 포트 설정  port [포트번호] # ex: port 6379  # 접속 패스워드 설정  requirepass [패스워드] # maro1229 설정 |

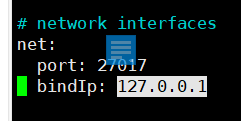
라) MongoDB 설치 및 설정

1) 버전 확인 및 public key 가져오기

|  |
| --- |
| lsb\_release -dc # 현재 우분투 버전 확인하기(20.04)  # 패키지 관리 시스템에서 사용하는 public key 가져오기  wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add - |

2) 패키지 설치

|  |
| --- |
| # list 파일 생성  echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list  # 로컬 패키지 데이터베이스 불러오기  sudo apt-get update  # mongodb 패키지 설치  sudo apt-get install -y mongodb-org |

 3) conf 파일 설정(bindIp를 0.0.0.0으로 수정)

|  |
| --- |
| sudo nano /etc/mongod.conf sudo systemctl restart mongod |

4) admin DB 및 관리자 추가

|  |
| --- |
| mongo  use admin  db.createUser({  user: "계정이름",  pwd: "비밀번호",  roles: [  {  role: "root",  db: "admin"  }  ]}  ) |

5) DB 생성 및 사용자 생성

|  |
| --- |
| use 데이터베이스명  db.createUser({  user: "사용자아이디",  pwd: "사용자비밀번호",  roles: [  {  role: "readWrite",  db: "데이터베이스명"  }  ]}  ) |

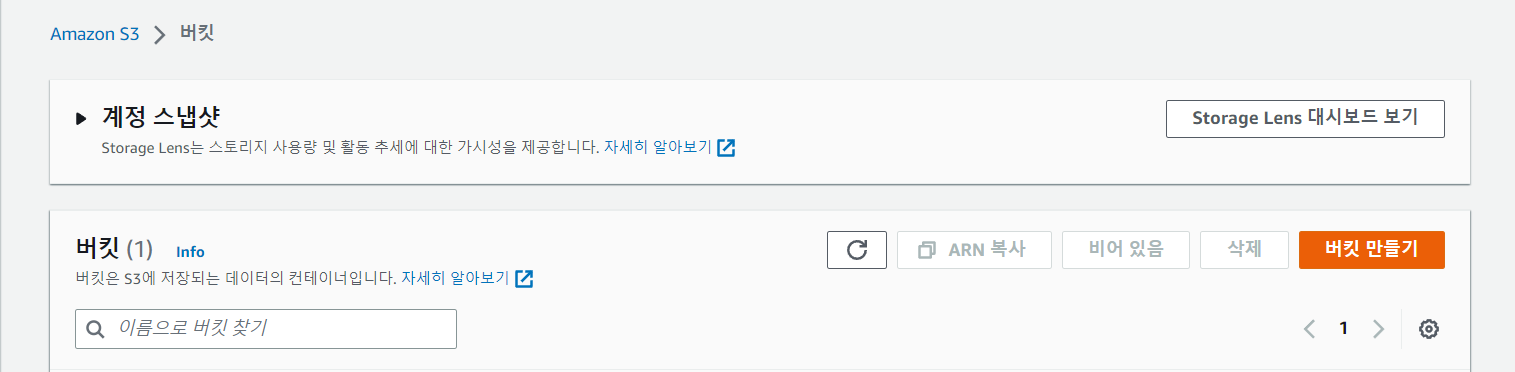
6) 등록 확인

|  |
| --- |
| use 데이터베이스명  db.getUsers() |

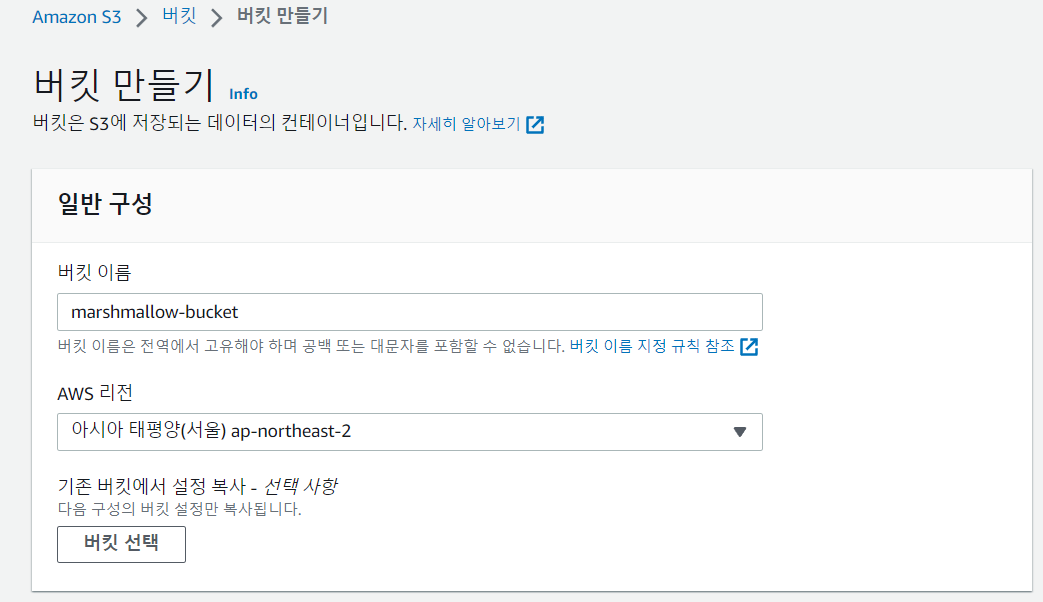
**6. 외부 서비스**

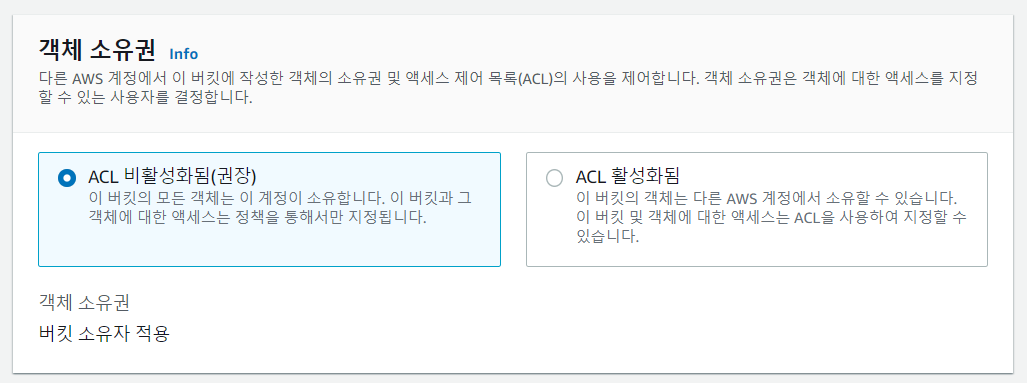
가) 아마존 S3

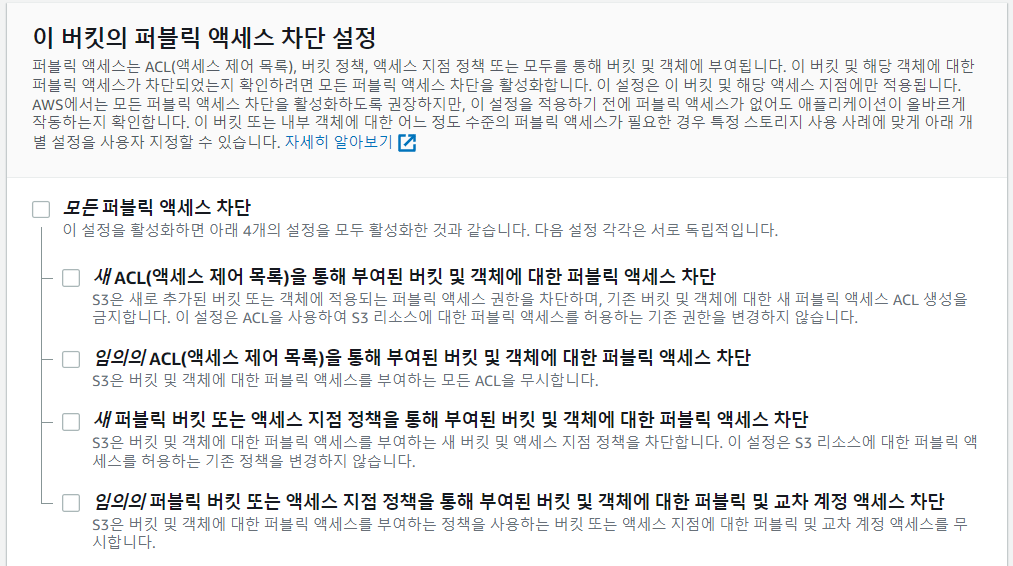
1. AWS 가입 후 S3 검색 후 버킷만들기 클릭

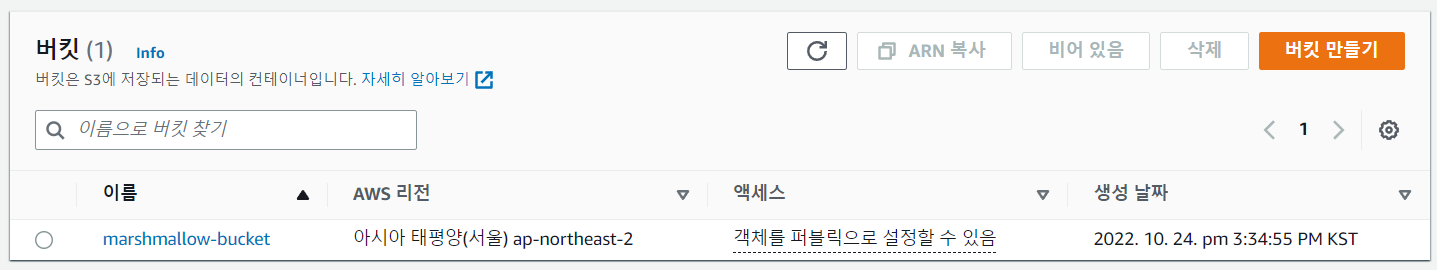


1. 버킷 이름, 지역 설정 후 객체 소유권 설정 후 생성



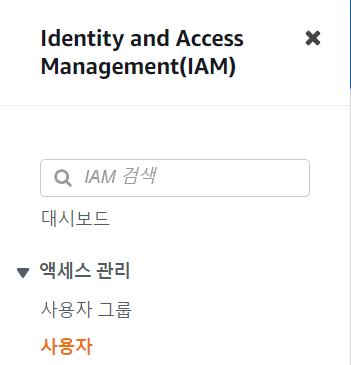
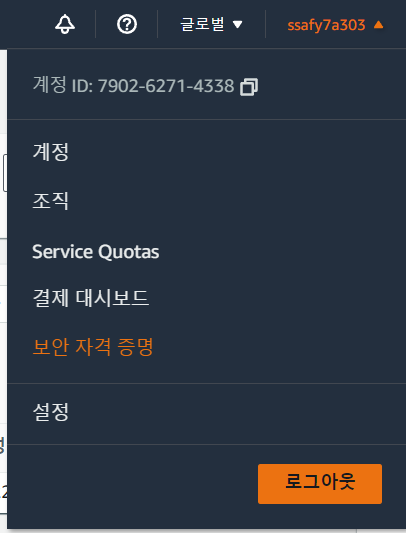


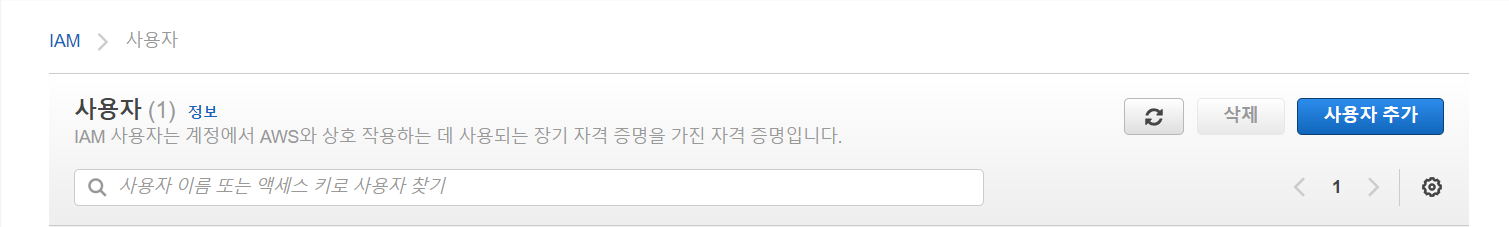




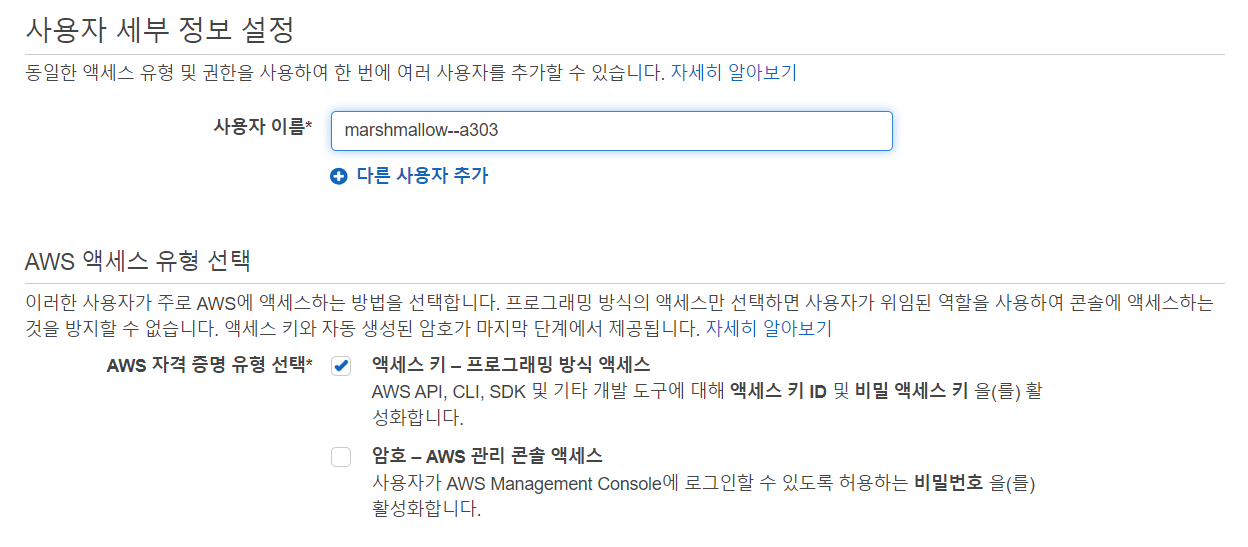
1. 버킷에 접근을 하기 위한 사용자 생성하기

a) 보안 자격 증명 > 사용자 > 사용자 추가 클릭

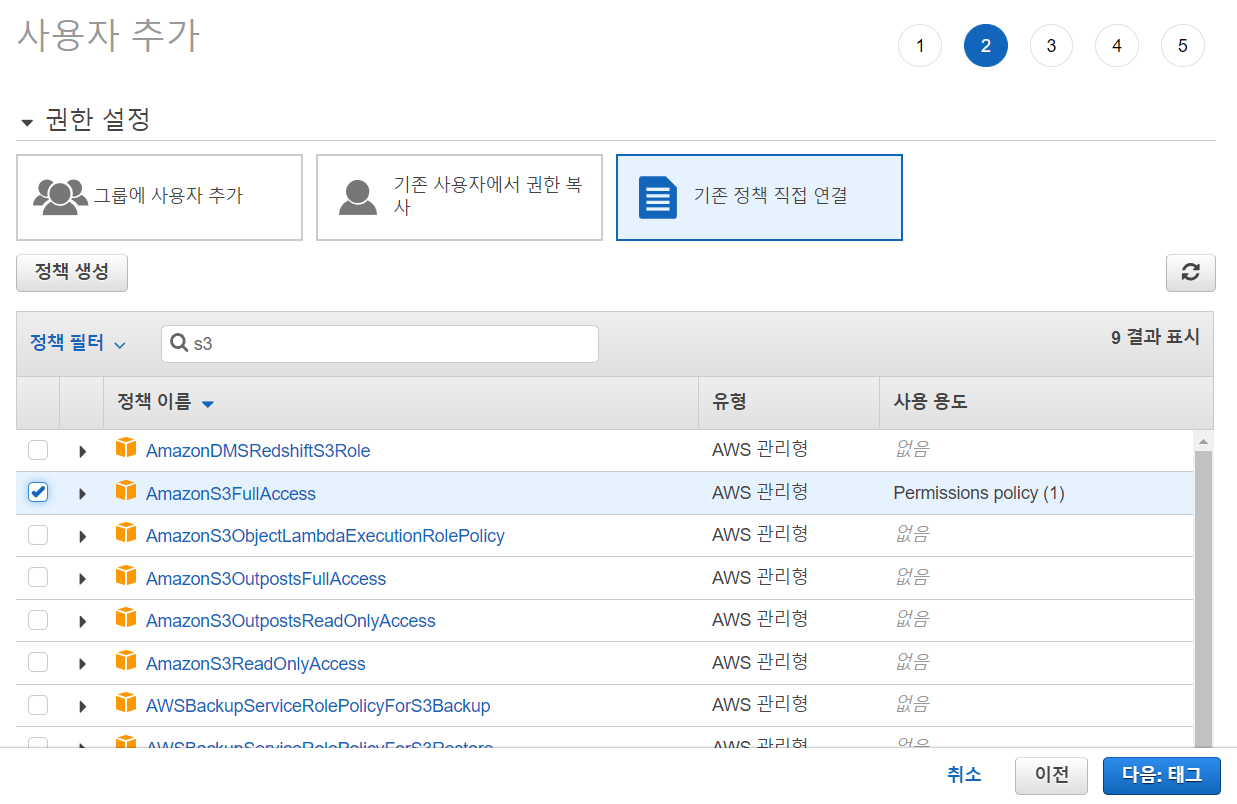




b) 사용자 이름 입력, 자격 증명 유형 – 엑세스 키 – 프로그래밍 방식 선택



c) 권한 설정 : 기존 정책 직접 연결 -> AmazonS3FullAccess 클릭



d) csv 다운로드 필수(access, secret key 파일)

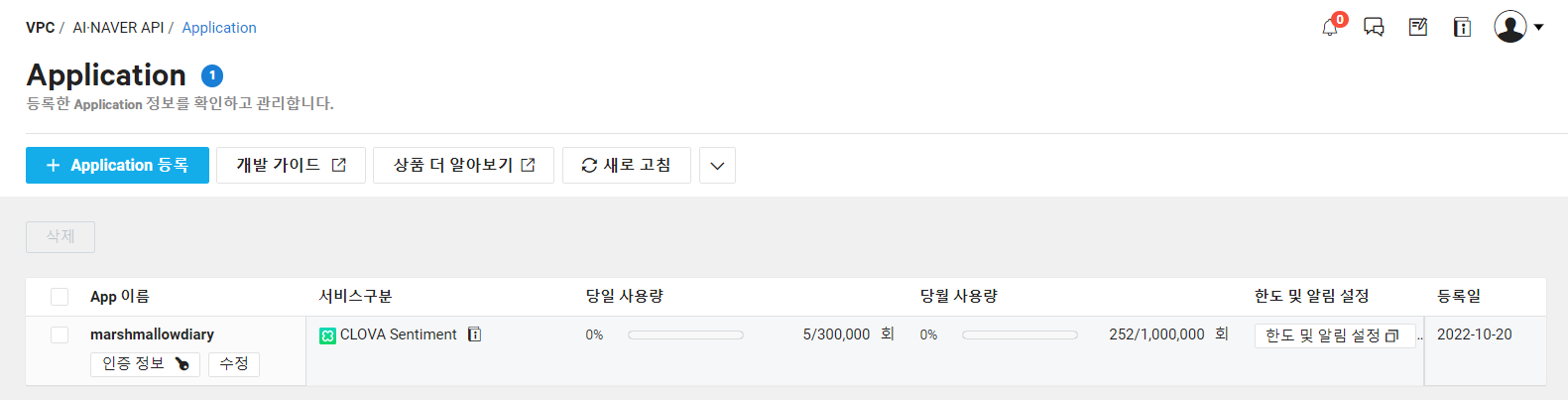


나) 네이버 클로바 감정 분석 api

1) <https://www.ncloud.com/product/aiService/clovaSentiment> 접속 후 이용 신청



2) 본인 네이버 계정으로 로그인 후 카드 등록



등록 완료