



그리미 포팅메뉴얼

I. 빌드 및 배포

1. 개발 환경
2. 설정 파일 목록
 - [React](#)
 - [Spring](#)
 - [FastApi-AI](#)
 - [FastApi-Weather API](#)
 - [Docker](#)
 - [Nginx](#)
3. 설정 파일 및 환경 변수 정보
 - [React](#)
 - [Spring](#)
 - [Docker](#)
 - [Nginx](#)
4. Docker 설치
5. DB 및 Infra 배포
7. CI/CD
 - [BackEnd - 설정파일](#)
 - [BackEnd - Jenkins 설정](#)
 - [FrontEnd - 설정파일](#)
 - [FrontEnd - Jenkins 설정](#)
8. FastApi-AI
9. FastApi-WeatherAPI

II. 외부 서비스

1. 소셜 로그인
 - [Google\(구글\)](#)
 - [Kakao\(카카오\)](#)
2. 가비아 도메인 적용
3. AWS S3
 - [1. S3 버킷 생성](#)
 - [2. IAM 추가](#)

I. 빌드 및 배포

1. 개발 환경

Server : AWS EC2 Ubuntu 20.04 LTS
Visual Studio Code : 1.70.1
IntelliJ IDEA : 2022.2(Ultimate Edition) 17.0.3+7-b469.32 amd64
JDK : openjdk-11
Docker : 20.10.20
Node.js : 18.9.1
MySQL : 8.0.30-1.el8
Nginx : 1.23.1
Jenkins : 2.361.1
Python: 3.9.13
FastApi: 0.85.1

2. 설정 파일 목록

React

- .env : /jenkins/workspace/frontend/frontend
- Dockerfile : /jenkins/workspace/frontend/frontend

Spring

- application.yml : /jenkins/workspace/backend/backend/src/main/resources
- Dockerfile : /jenkins/workspace/backend/backend
- deploy.sh : /jenkins/workspace/backend/backend

FastApi-AI

- Dockerfile: /home/ubuntu/ai/FastApi
- docker-compose.yml: /home/ubuntu/ai/FastApi
- requirements.txt: /home/ubuntu/ai/FastApi

FastApi-Weather API

- Dockerfile: /home/ubuntu/weather/weatherApi
- docker-compose.yml: /home/ubuntu/weather/weatherApi
- requirements.txt: /home/ubuntu/weather/weatherApi

Docker

- docker-compose.yml : /home/ubuntu

Nginx

- app.conf : /home/ubuntu/nginx/conf.d

3. 설정 파일 및 환경 변수 정보

React

- .env

```
WDS_SOCKET_PORT=0

REACT_APP_BASE_URL={REST API 요청 URL : ex) 서비스 도메인/api}

REACT_APP_GOOGLE_CLIENT_ID={구글 클라이언트 ID}
REACT_APP_GOOGLE_CLIENT_SECRET={구글 클라이언트 SECRET}
REACT_APP_GOOGLE_REDIRECT_URI={구글 리다이렉트 URI}

REACT_APP_NAVER_CLIENT_ID={네이버 클라이언트 ID}
REACT_APP_NAVER_CLIENT_SECRET={네이버 클라이언트 SECRET}
REACT_APP_NAVER_REDIRECT_URI={네이버 리다이렉트 URI}

REACT_APP_KAKAO_JAVASCRIPT_KEY={카카오 자바스크립트 KEY}
REACT_APP_KAKAO_REST_API_KEY={카카오 REST API KEY}
REACT_APP_KAKAO_CLIENT_ID={카카오 클라이언트 ID}
REACT_APP_KAKAO_CLIENT_SECRET={카카오 클라이언트 SECRET}
REACT_APP_KAKAO_REDIRECT_URI={카카오 리다이렉트 URI}
```

- Dockerfile

```
FROM node:alpine
WORKDIR /usr/src/app
COPY ./package* /usr/src/app/
RUN npm install --save --legacy-peer-deps
COPY ./ /usr/src/app/
CMD ["npm", "run", "start"]
```

Spring

- application.yml

```

server:
  port: 8000
  servlet:
    context-path: /api

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{DB컨테이너이름}:3306/{SCHEMA이름}?serverTimezone=Asia/Seoul&useUnicode=yes&characterEncoding=UTF-8&zeroDateTmeB
ehavior=convertToNull
    username: {사용자계정ID}
    password: {사용자계정PASSWORD}
  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: false
        dialect: org.hibernate.dialect.MySQL8Dialect
        generate_statistics: false
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    open-in-view: false
    show-sql: false
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
  servlet:
    multipart:
      max-file-size: 10MB
  security:
    oauth2:
      client:
        registration:
          # GOOGLE LOGIN
          google:
            client-id: {구글 클라이언트 ID}
            client-secret: {구글 클라이언트 SECRET}
            redirect-uri: {구글 리다이렉트 URI}
            scope: profile, email
          # KAKAO LOGIN
          kakao:
            client-id: {카카오 클라이언트 ID}
            client-secret: {카카오 클라이언트 SECRET}
            redirect-uri: {카카오 리다이렉트 URI}
            client-authentication-method: POST
            authorization-grant-type: authorization_code
            client-name: Kakao
            scope: profile_nickname, account_email
          # KAKAO PROVIDER
          provider:
            kakao:
              authorization-uri: https://kauth.kakao.com/oauth/authorize
              token-uri: https://kauth.kakao.com/oauth/token
              user-info-uri: https://kapi.kakao.com/v2/user/me
              user-name-attribute: id
# bucket4j ip주소 캐시
cache:
  jcache:
    provider: com.github.benmanes.caffeine.jcache.spi.CaffeineCachingProvider
    config: classpath:ehcache.xml
  cache-names:
    - rate-limit-buckets
  caffeine:
    spec: maximumSize=100000,expireAfterAccess=3600s

decorator:
  datasource:
    p6spy:
      enable-logging: false

# bucket4j 설정
bucket4j:
  enabled: true
  filters:
    - cache-name: rate-limit-buckets
      filter-method: servlet
      url: .*
      strategy: all
      http-response-body: "{ \"status\": 429, \"error\": \"Too Many Requests\", \"message\": \"You have exhausted your Request Quot
a\" }"
  rate-limits:
    - expression: "getRemoteAddr()"
      bandwidths:
        - capacity: 50

```

```

        time: 1
        unit: seconds
        fixed-refill-interval: 1
        fixed-refill-interval-unit: seconds

logging:
  level:
    root: info

# AWS S3
cloud:
  aws:
    credentials:
      access-key: {AWS ACCESS KEY}
      secret-key: {AWS SECRET KEY}
    s3:
      bucket: {BUCKET NAME}
      region:
        static: ap-northeast-2
      stack:
        auto: falseserver.port=8000

```

- Dockerfile

```

FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "/app.jar"]

```

- deploy.sh

```

echo '실행 시작'
echo 'git pull'
echo 'jar 파일 삭제'
rm build/libs/*.jar
echo '빌드 시작'
./gradlew build
echo '도커파일 이미지 빌드'
docker build -t springbootapp .
echo '컨테이너 중지'
docker stop springbootapp
echo '컨테이너 삭제'
docker rm springbootapp
echo '컨테이너 실행'
docker run -p {포트 번호}:{포트 번호} --name springbootapp --network ubuntu_default -d springbootapp

```

Docker

- docker-compose.yml

```

version: "3"
services:
  mysql:
    image: mysql
    container_name: mysql
    environment:
      MYSQL_DATABASE: {SCHEME 이름}
      MYSQL_ROOT_PASSWORD: {ROOT 계정 PASSWORD}
    volumes:
      - /mysql:/var/lib/mysql
    ports:
      - 3306:3306
  nginx:
    image: nginx
    container_name: nginx
    ports:
      - 80:80
      - 443:443
    volumes:
      - /etc/letsencrypt:/etc/letsencrypt
      - ./nginx/conf.d:/etc/nginx/conf.d
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home

```

```

ports:
  - 9090:8080
privileged: true
user: root

fastapi:
  image: fastapi
  container_name: fastapi
  command: uvicorn app.main:app --host 0.0.0.0 --port 8010 --reload
  ports:
    - 8010:8010
  volumes:
    - ./app:/code/app

```

Nginx

- app.conf

```

server {
    listen 80;
    server_name {서비스 도메인} www.{서비스 도메인};
    return 301 https://$server_name$request_uri;
}
server {
    listen 443 ssl;
    server_name {서비스 도메인};
    access_log off;

    ssl_certificate
    /etc/letsencrypt/live/{서비스 도메인}/fullchain.pem;
    ssl_certificate_key
    /etc/letsencrypt/live/{서비스 도메인}/privkey.pem;

    location / {
        proxy_pass http://{서비스 도메인}:3000;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;

        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 Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_redirect off;
    }
    location /api/ {
        proxy_pass http://{서비스 도메인}:8000/api/;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Host $server_name;
        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 Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_redirect off;
        # add_header 'Access-Control-Allow-Origin' '*' always;
        # add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DELETE, OPTIONS' always;
        # add_header 'Access-Control-Allow-Headers' 'content-type, authorization, x-requested-with' always;
    }
    location /ai/ {
        proxy_pass http://{서비스 도메인}:8010/ai/;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Host $server_name;
        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 Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_redirect off;
    }
    location /weather/ {
        proxy_pass http://{서비스 도메인}:8020/weather;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Host $server_name;
        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 Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}

```

```

    proxy_redirect off;
}
}
}

```

4. Docker 설치

- Docker 설치 (출처 : <https://docs.docker.com/engine/install/ubuntu/>)

```

sudo apt-get update

sudo apt-get install \
ca-certificates \
curl \
gnupg \
lsb-release

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) signedby=/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

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

```

- Docker Compose 설치 (출처 : <https://docs.docker.com/compose/install/other/>)

```

sudo curl -SL https://github.com/docker/compose/releases/download/v2.12.1/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose

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

```

5. DB 및 Infra 배포

1. docker-compose 작성 ([docker-compose.yml](#))

위치 : /home/ubuntu

2. docker-compose 실행

/home/ubuntu (docker-compose.yml 있는 경로에서)

```

sudo docker-compose up --build -d

```

3. Nginx 설정 ([app.conf](#))

위치 : /home/ubuntu/nginx/conf.d

4. SSL 인증서 발급 - Certbot 설치 (Nginx on Ubuntu 20)

(출처 : <https://certbot.eff.org/instructions?ws=nginx&os=ubuntu>)

```

1. snapd 설치
sudo apt-get install snapd

2. 설치되었는지 확인
sudo snap install core; sudo snap refresh core

3. 이미 설치된 certbot이 있다면 삭제
sudo apt-get remove certbot

4. certbot 설치
sudo snap install --classic certbot

5. certbot command 준비하기
sudo ln -s /snap/bin/certbot /usr/bin/certbot

6. 인증서 받아 nginx 자동설정
sudo certbot --nginx
이메일 입력 후 이용약관 동의(Y)
certbot 정보 이메일 받고 싶으면 동의
domain 주소 입력 (ex : k7a506.p.ssafy.io)

```

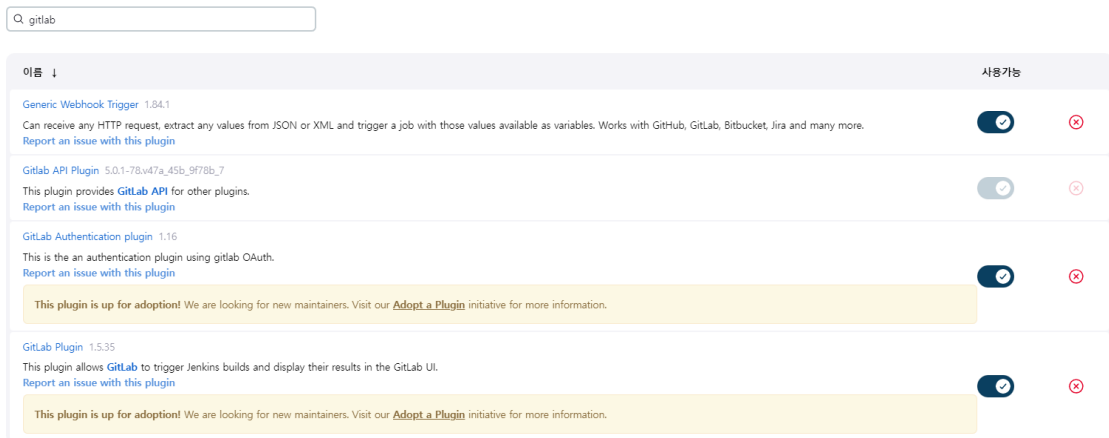
7. 자동 갱신 테스트

```
sudo certbot renew --dry-run
```

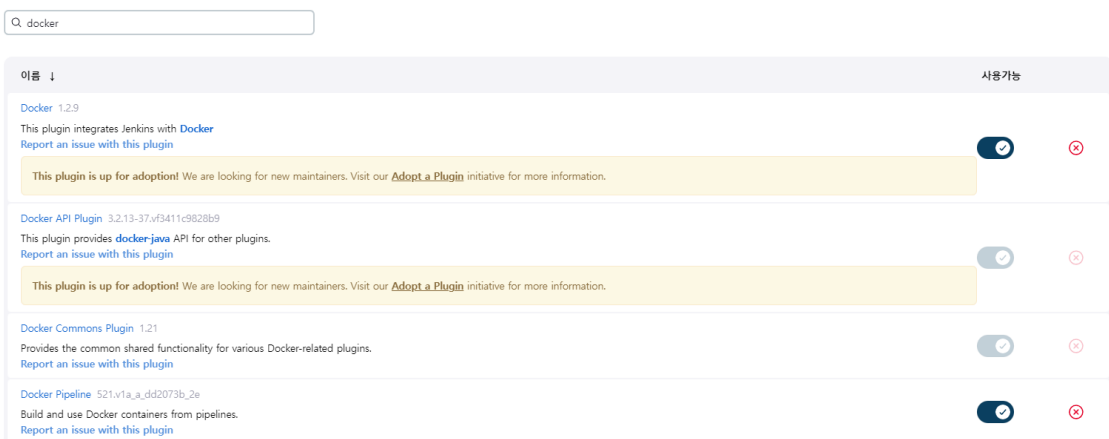
(여기서 오류난다면 실행 중인 nginx 컨테이너 있는지 보고 중지 시킨 후 테스트
> 컨테이너 재시작 하면 됨)

5. Jenkins 설정

- <http://{서비스 도메인}:9090> 으로 접속하여 jenkins 페이지 진입
- 서버 콘솔에서 `sudo docker logs {Jenkins 컨테이너 이름}`으로 Administrator password를 확인하고 입력
- Install suggested plugins를 선택하여 권장 플러그인 설치
- 생성할 관리자 계정 정보를 입력하고 Save and Continue
- Jenkins 접속 URL 확인 후 Save and Finish
- 메인 화면에서 DashBoard > Manager Jenkins > Plugin Manager
gitlab 검색 후 스크린샷과 같이 설치



docker 검색 후 스크린샷과 같이 설치



vii. 서버 콘솔에서 Jenkins 내부에 docker를 설치

```
$ sudo docker exec -it {jenkins 컨테이너명} /bin/bash
$ apt-get update -y
$ apt-get install -y
$ apt-get install docker.io -y
$ docker -v
```

7. CI/CD

BackEnd - 설정파일

- application.yml작성 ([application.yml](#))
- `docker cp /home/ubuntu/exec/application.yml jenkins:/var/jenkins_home/workspace/`
- Dockerfile 작성 ([Dockerfile](#))
- build.gradle 파일에 `jar { enabled=false }` 추가

BackEnd - Jenkins 설정

1. Jenkins 메인 화면 > Dashboard > 새로운 Item
2. FreeStyle project 선택하고 item name은 backend로 설정 후 OK
3. Repositories URL에는 프로젝트 레포지토리의 HTTPS 주소 입력
 - Credentials > Add 클릭
 - Domain > Global credentials
 - Kind > Username with password
 - Username > 레포지토리 접근 권한이 있는 GitLab 계정 아이디
 - Password > 레포지토리 접근 권한이 있는 GitLab 계정 비밀번호
4. 생성된 Credential을 선택
5. 스크린샷과 같이 설정

Branches to build ?

Branch Specifier (blank for 'any') ?

refs/heads/{브랜치 이름}

{브랜치 이름} 브랜치의 내용만을 받아와서 빌드

빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: `http://j7a701.p.ssafy.io:9090/project/backend` ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Push, Merge Request가 발생할 때마다 빌드 유발

Build Steps

Execute shell ?

Command

See [the list of available environment variables](#)

```
cd backend
chmod +x gradlew
chmod +x deploy.sh
./deploy.sh
```

고급...

backend 폴더로 이동하여 gradlew 와 deploy.sh 파일에 실행할 수 있는 권한 부여

```
cp /var/jenkins_home/workspace/application.yml /var/jenkins_home/workspace/backend/backend/src/main/resources
cd backend
chmod +x gradlew
chmod +x deploy.sh
./deploy.sh
```

6. Jenkins에 sudo 권한 부여 (중요한 파일이기 때문에 오타 없는지 한번 더 확인)

```
$ sudo vim /etc/sudoers
```

```
# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL
jenkins ALL=(ALL) NOPASSWD: ALL
```

해당 부분에 `jenkins ALL=(ALL) NOPASSWD: ALL` 작성

7. GitLab Webhook 작성 (Settings > Webhook)

Q Search page

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

☒ Push events

Push to the repository.

FrontEnd - 설정파일

- .env 작성 ([.env](#))
- Dockerfile 작성 ([Dockerfile](#))

FrontEnd - Jenkins 설정

1. Jenkins 메인 화면 > Dashboard > 새로운 Item
2. FreeStyle project 선택하고 item name은 frontend로 설정 후 OK
3. Repositories URL에는 프로젝트 레포지토리의 HTTPS 주소 입력
4. 이전 과정에서 생성했던 Credential을 선택

5. 스크린샷과 같이 설정

Branches to build ?

Branch Specifier (blank for 'any') ?

refs/heads/{브랜치 이름}

{브랜치 이름} 브랜치의 내용만을 받아와서 빌드

빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j7a701p.ssaftyio:9090/project/backend> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Push, Merge Request가 발생할 때마다 빌드 유발

Build Steps

Execute shell ?

Command

See [the list of available environment variables](#)

```
cd frontend
docker build -t reactapp .
docker stop reactapp
docker rm reactapp
```

고급...

frontend 폴더로 이동하여 도커 컨테이너 생성, 기존 컨테이너 중지 후 삭제, 새로운 컨테이너 run

```
cp /var/jenkins_home/workspace/.env /var/jenkins_home/workspace/frontend/frontend/eurime
cd frontend/eurime
docker build -t reactapp .
docker stop reactapp
docker rm reactapp
docker run -p 3000:3000 --name reactapp --network ubuntu_default -d reactapp
```

위의 방법으로 안될 경우 EC2 서버 내에서 아래 명령어를 실행 후 다시 시도해본다.

```
docker run -p 3000:3000 --name reactapp --network ubuntu_default -d reactapp
```

7. GitLab Webhook 작성 (Settings > Webhook)

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

☒ Push events

Push to the repository.

8. FastApi-AI

1. Dockerfile 실행 (위치: /home/ubuntu/ai/FastApi)

```
docker build -t fastapi .
```

2. docker-compose.yml 백그라운드로 실행 (위치: /home/ubuntu/ai/FastApi)

```
docker-compose up&
```

9. FastApi-WeatherAPI

1. Dockerfile 실행 (위치: /home/ubuntu/weather/weatherApi)

```
docker build -t fastapi .
```

2. docker-compose.yml 백그라운드로 실행 (위치: /home/ubuntu/weather/weatherApi)

```
docker-compose up&
```

II. 외부 서비스

1. 소셜 로그인

Google(구글)

1. <https://console.cloud.google.com/home/dashboard>로 접속한다.



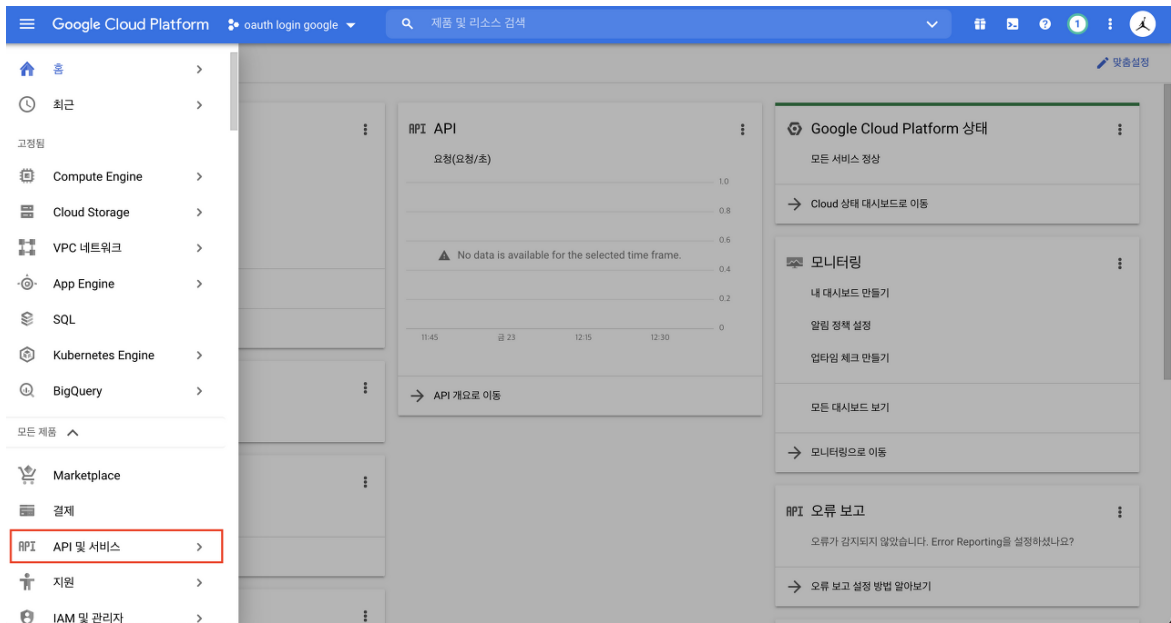
2. 프로젝트 선택을 클릭하고 모달 우측 상단의 새 프로젝트 클릭

프로젝트 선택

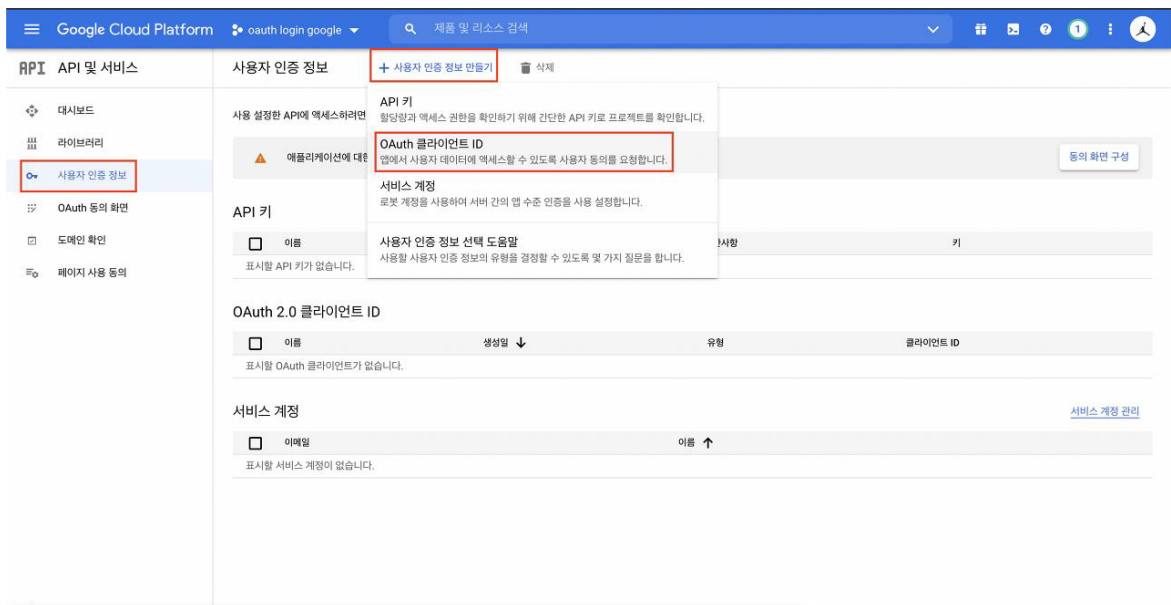
새 프로젝트

🔍 프로젝트 및 폴더 검색	
최근	별표표시된 프로젝트
전체	
이름	ID
📁 조직 없음	0

3. API 및 서비스



4. 사용자 인증 정보 > 사용자 인증 정보 만들기 > OAuth 클라이언트 ID



5. 동의 화면 구성을 클릭하고 해당 화면에서 외부로 선택하여 외부 사용자들이 사용할 수 있도록 설정한다.

OAuth 동의 화면

대상 사용자를 비롯해 앱을 구성하고 등록하려는 방식을 선택하세요. 프로젝트에는 하나의 앱만 연결할 수 있습니다.

User Type

☐ 내부 ?

Google Workspace 사용자가 아니기 때문에 앱을 외부(일반 잠재고객) 사용자에게 제공하는 것만 가능합니다. [외부 앱을 제출할 필요는 없습니다.](#)

☒ 외부 ?

Google 계정이 있는 모든 테스트 사용자가 사용할 수 있습니다. 앱이 테스트 모드로 시작되고 테스트 사용자 목록에 추가된 사용자에게만 제공됩니다. 앱을 프로젝트에 푸시할 준비가 되면 앱을 인증해야 할 수도 있습니다. [사용자 유형 자세히 알아보기](#)

만들기

6. 앱 정보를 입력하고, 개발자 연락처 정보에 이메일을 입력한다.

앱 정보

동의 화면에 표시되어 최종 사용자가 개발자를 확인하고 문의할 수 있습니다.

앱 이름 *

그리미

동의를 요청하는 앱의 이름

사용자 지원 이메일 *

0atx.yh@gmail.com

사용자가 동의 관련 질문을 위해 문의할 때 이용합니다.

앱 로고

logo120.png

✕ [찾아보기](#)

사용자가 앱을 알아보는 데 도움이 되도록 동의 화면에 대한 이미지(1MB 이하 크기)를 업로드합니다. 허용되는 이미지 형식은 JPG, PNG, BMP입니다. 최적의 결과를 위해서는 로고가 120x120픽셀 크기의 정사각형이어야 합니다.



개발자 연락처 정보

이메일 주소 *

0atx.yh@gmail.com ✕

이 이메일 주소는 Google에서 프로젝트 변경사항에 대해 알림을 보내기 위한 용도입니다.

7. 범위 추가 또는 삭제 > email, profile 선택 후 저장
민감하지 않은 범위에 추가된 것 확인

선택한 범위 업데이트

아래에는 사용 설정된 API의 범위만 나와 있습니다. 이 화면에 누락된 범위를 추가하려면 [Google API 라이브러리](#)에서 API를 찾아 사용 설정하거나 아래의 **불여널은 범위** 텍스트 상자를 사용하세요. 라이브러리에서 사용 설정한 새 API를 확인하려면 페이지를 새로고침하세요.

필터: **속성 이름 또는 값 입력**

API	범위	사용자에게 표시되는 설명
<input checked="" type="checkbox"/>	../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
<input checked="" type="checkbox"/>	../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기
<input type="checkbox"/>	openid	Google에서 내 개인 정보를 나와 연결
<input type="checkbox"/>	BigQuery API ../auth/bigquery	View and manage your data in Google BigQuery and see the email address for your Google Account.
<input type="checkbox"/>	BigQuery API ../auth/cloud-platform	Google Cloud 데이터 확인, 수정, 구성, 삭제 및 Google 계정의 이메일 주소 확인
<input type="checkbox"/>	BigQuery API ../auth/bigquery.readonly	Google BigQuery에서 데이터를 봅니다.
<input type="checkbox"/>	BigQuery API ../auth/cloud-platform.readonly	Google Cloud 서비스 전체의 데이터 조회 및 Google 계정의 이메일 주소 확인
<input type="checkbox"/>	BigQuery API ../auth/devstorage.full_control	Manage your data and permissions in Cloud Storage and see the email address for your Google Account
<input type="checkbox"/>	BigQuery API ../auth/devstorage.read_only	Google 클라우드 저장소에서 데이터 조회
<input type="checkbox"/>	BigQuery API ../auth/devstorage.read_write	Cloud Storage의 데이터 관리 및 Google 계정의 이메일 주소 확인

페이지당 항목 수: 10 ▼ 1 - 10 (전체 25점) < >

직접 범위 추가

추가할 범위가 위 표에 표시되지 않으면 여기에 입력할 수 있습니다. 각 범위를 새 줄에 입력하거나 실행으로 구분해야 합니다. <https://>로 시작하는 전체 범위 문자열을 입력하세요. 완료되면 표에 추가를 클릭하세요.

8. 사용자 인증정보 > 사용자 인증정보 만들기 > OAuth 클라이언트 ID 승인된 리디렉션 URI 작성

이름 *

GEURIME

OAuth 2.0 클라이언트의 이름입니다. 이 이름은 콘솔에서 클라이언트를 식별하는 용도로만 사용되며 최종 사용자에게 표시되지 않습니다.



아래에 추가한 URI의 도메인이 [승인된 도메인](#)으로 [OAuth 동의 화면](#)에 자동으로 추가됩니다.

승인된 자바스크립트 원본 ?

브라우저 요청에 사용

URI 1 *

https://k7a506.p.ssafy.io

+ URI 추가

승인된 리디렉션 URI ?

웹 서버의 요청에 사용

URI 1 *

https://k7a506.p.ssafy.io/api/login/oauth2/code/google

+ URI 추가

Kakao(카카오)

1. <https://developers.kakao.com/>로 접속한다.



2. 내 애플리케이션 > 애플리케이션 추가하기 > 앱 이름 입력 > 사업자명 입력 > 저장

애플리케이션 추가하기

앱 아이콘



파일 선택

JPG, GIF, PNG

권장 사이즈 128px, 최대 250KB

앱 이름

그리미

사업자명

A506

- 입력된 정보는 사용자가 카카오 로그인할 때 표시됩니다.
- 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

☒ [서비스 이용이 제한되는 카테고리](#), [금지된 내용](#), [금지된 행동](#) 관련 운영정책을 위반하지 않는 앱입니다.

취소

저장

3. 좌측 Nav 바에서

앱 설정 > 요약 정보 > 앱 키 > REST API 키와

제품 설정 > 카카오 로그인 > 보안 > Client Secret 의 코드를 기록해둔다.



그리미



ID 812480

OWNER

Biz

Web

앱 키

플랫폼	앱 키		재발급
네이티브 앱 키	<div></div>	<button>복사</button>	<button>재발급</button>
REST API 키	<div></div>	<button>복사</button>	<button>재발급</button>
JavaScript 키	<div></div>	<button>복사</button>	<button>재발급</button>
Admin 키	<div></div>	<button>복사</button>	<button>재발급</button>

- 네이티브 앱 키: Android, iOS SDK에서 API를 호출할 때 사용합니다.
- JavaScript 키: JavaScript SDK에서 API를 호출할 때 사용합니다.
- REST API 키: REST API를 호출할 때 사용합니다.
- Admin 키: 모든 권한을 갖고 있는 키입니다. 노출이 되지 않도록 주의가 필요합니다.

Client Secret

삭제

토큰 발급 시, 보안을 강화하기 위해 Client Secret을 사용할 수 있습니다. (REST API인 경우에 해당)

코드	<div></div>	재발급
활성화 상태	사용함	설정

4. 좌측 Nav 바의

앱 설정 > 플랫폼 > Web > Web 플랫폼 등록에서 필요 정보를 기입한다.

Web 플랫폼 등록

사이트 도메인

JavaScript SDK, 카카오톡 공유, 카카오톡, 메시지 API 사용시 등록이 필요합니다.

여러개의 도메인은 줄바꿈으로 추가해주세요. 최대 10까지 등록 가능합니다. 추가 등록은 포럼(데브톡)으로 문의주세요.

예시: (O) <https://example.com> (X) <https://www.example.com>

<https://k7a506.p.ssafy.io>

기본 도메인

기본 도메인은 첫 번째 사이트 도메인으로, 카카오톡 공유와 카카오톡 메시지 API를 통해 발송되는 메시지의 Web 링크 기본값으로 사용됩니다.

<https://k7a506.p.ssafy.io>

취소

저장

5. 제품 설정 > 카카오 로그인 클릭 후 Redirect URI 를 입력한다.

Redirect URI

Redirect URI

카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다.

여러개의 URI를 줄바꿈으로 추가해주세요. (최대 10개)

REST API로 개발하는 경우 필수로 설정해야 합니다.

예시: (O) <https://example.com/oauth> (X) <https://www.example.com/oauth>

<https://k7a506.p.ssafy.io/api/login/oauth2/code/kakao>
|

취소

저장

6. 제품 설정 > 카카오 로그인 > 동의항목에서 아래와 같이 설정한다.

동의항목

카카오 로그인으로 서비스를 시작할 때 동의 받는 항목을 설정합니다. 미리 보기를 통해 사용자에게 보여질 화면을 확인할 수 있습니다.
비즈니스 채널을 연결하면 권한이 없는 동의 항목에 대한 검수 신청을 할 수 있습니다.

[카카오비즈니스 관리자센터에서 비즈니스 채널 연결 →](#)

개인정보

항목 이름	ID	상태	
닉네임	profile_nickname	● 필수 동의	설정
프로필 사진	profile_image	● 사용 안함	설정
카카오계정(이메일)	account_email	● 필수 동의 [수집]	설정
성별	gender	● 사용 안함	설정
연령대	age_range	● 사용 안함	설정

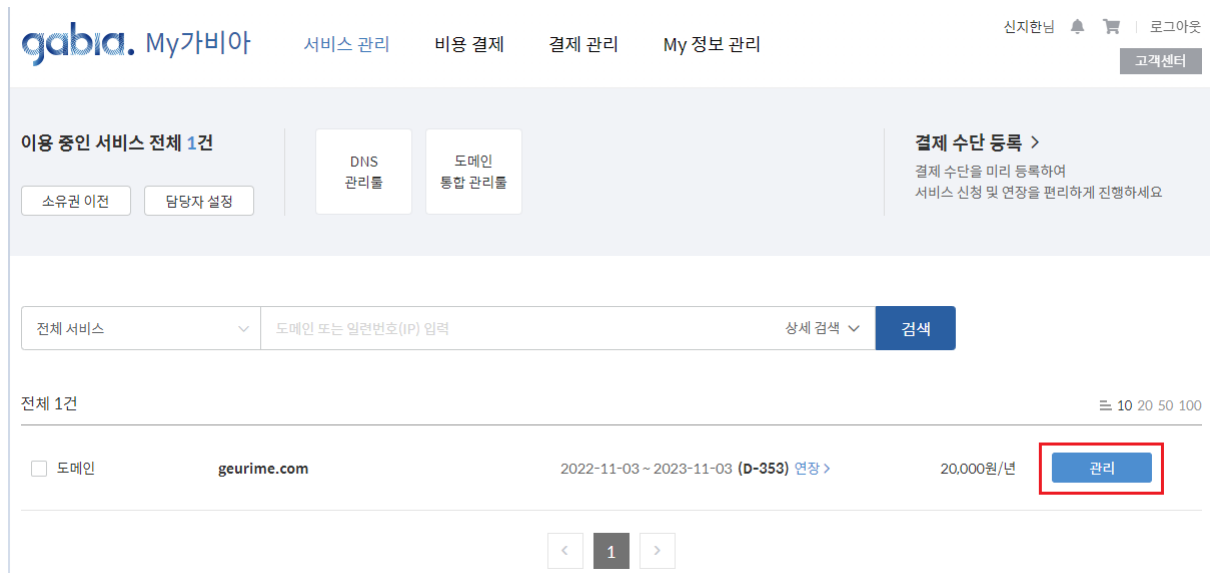
- application.yml 작성 ([application.yml](#))

2. 가비아 도메인 적용

1. <https://www.gabia.com/>로 접속한다.



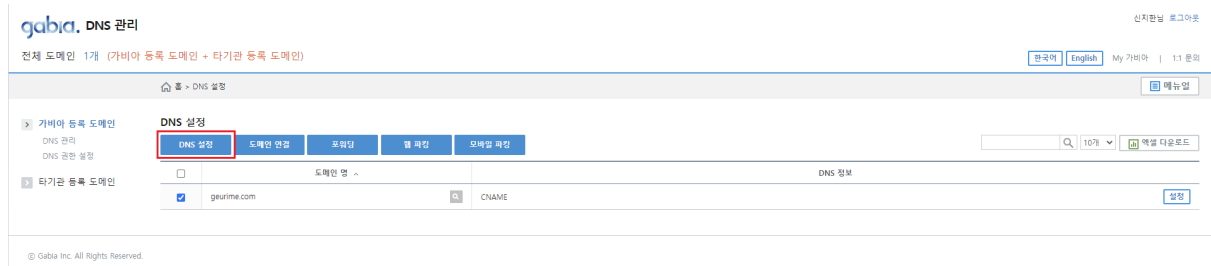
2. My가비아의 서비스 관리 탭을 클릭하여 서비스 관리 탭으로 이동한다.



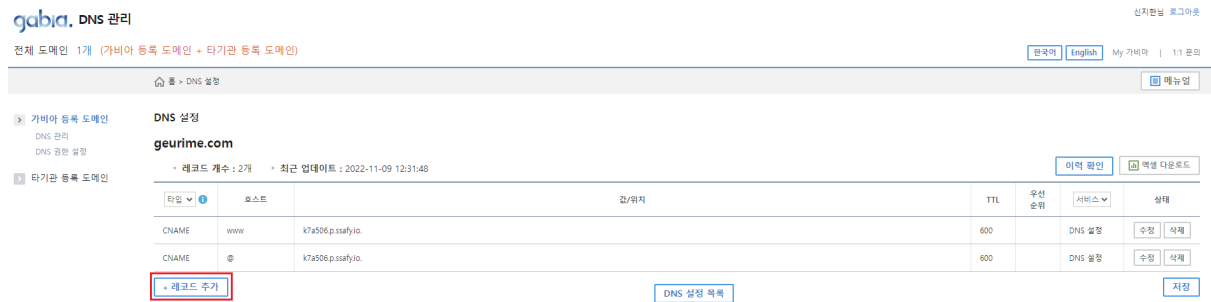
3. 사용할 도메인의 관리 탭을 클릭하여 관리 페이지로 이동한다.



4. DNS 정보 탭에서 DNS 관리 버튼을 눌러 DNS 관리툴로 이동한다.



5. 설정하고자 하는 도메인을 클릭한 후 DNS 설정 버튼을 클릭한다.



6. 레코드 추가 버튼을 클릭하여 위와 같이 기존 도메인 값을 넣어준다.



7. 이후 설정을 마친 도메인으로 접속하면 기존 서버와 연결이 잘 되어있는 것을 확인할 수 있다.

3. AWS S3

1. S3 버킷 생성

버킷 만들기 Info

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

일반 구성

버킷 이름

버킷 이름은 전역에서 고유해야 하며 공백 또는 대문자를 포함할 수 없습니다. [버킷 이름 지정 규칙 참조](#)

AWS 리전

기존 버킷에서 설정 복사 - 선택 사항
다음 구성의 버킷 설정만 복사됩니다.

AWS Console > S3 서비스 > 버킷 만들기

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 모든 퍼블릭 액세스 차단을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 모든 퍼블릭 액세스 차단을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 이 버킷 또는 내부 객체에 대한 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

☐ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

☐ 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.

☐ 임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.

☐ 새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

☐ 임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.



모든 퍼블릭 액세스 차단을 비활성화하면 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다.

정적 웹 사이트 호스팅과 같은 구체적으로 확인된 사용 사례에서 퍼블릭 액세스가 필요한 경우가 아니면 모든 퍼블릭 액세스 차단을 활성화하는 것이 좋습니다.

☐ 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다.

모든 퍼블릭 액세스 차단 체크해제후 버킷 생성

버킷 정책 편집 [Info](#)

버킷 정책
JSON으로 작성된 버킷 정책은 버킷에 저장된 객체에 대한 액세스 권한을 제공합니다. 버킷 정책은 다른 계정이 소유한 객체에는 적용되지 않습니다. 자세히 알아보기 [↗](#)

정책 예제 [↗](#) 정책 생성기 [↗](#)

버킷 ARN
arn:aws:s3:::geurime-a506

생성된 버킷의 설정 > 권한 > 버킷 정책 > 편집

버킷 ARN 복사하기



AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [key concepts in Using AWS Identity and Access Management](#). Here are [sample policies](#).

Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an SNS Topic Policy, a VPC Endpoint Policy, and an SQS Queue Policy.

Select Type of Policy S3 Bucket Policy

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal *

Use a comma to separate multiple values.

AWS Service Amazon S3 ☐ All Services ("*")

Use multiple statements to add permissions for more than one service.

Actions Select Actions ☒ All Actions ("*")

Amazon Resource Name (ARN) arn:aws:s3:::geurime-a506

ARNs should follow the following format: arn:aws:s3:::{BucketName}/{Key(Name)}.
Use a comma to separate multiple values.

Add Conditions (Optional)

Add Statement

Type of Policy : S3 Bucket Policy

principal : *

Actions : All Actions

ARN : 복사한 ARN 붙여넣기

You added the following statements. Click the button below to Generate a policy.

Principal(s)	Effect	Action	Resource	Conditions
*	Allow	s3:*	arn:aws:s3:::geurime-a506	None

Step 3: Generate Policy

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Generate Policy Start Over

Generate Policy 클릭

사용자 추가

1 2 3 4 5

사용자 세부 정보 설정

동일한 액세스 유형 및 권한을 사용하여 한 번에 여러 사용자를 추가할 수 있습니다. [자세히 알아보기](#)

사용자 이름* mario

[+ 다른 사용자 추가](#)

AWS 액세스 유형 선택

이러한 사용자가 주로 AWS에 액세스하는 방법을 선택합니다. 프로그래밍 방식의 액세스만 선택하면 사용자가 위임된 역할을 사용하여 콘솔에 액세스하는 것을 방지할 수 없습니다. 액세스 키와 자동 생성된 암호가 마지막 단계에서 제공됩니다. [자세히 알아보기](#)

AWS 자격 증명 유형 선택*


- ☒ 액세스 키 – 프로그래밍 방식 액세스
AWS API, CLI, SDK 및 기타 개발 도구에 대해 액세스 키 ID 및 비밀 액세스 키 을(를) 활성화합니다.
- ☐ 암호 – AWS 관리 콘솔 액세스
사용자가 AWS Management Console에 로그인할 수 있도록 허용하는 비밀번호 을(를) 활성화합니다.


사용자 이름을 입력하고 액세스 유형 체크


사용자 추가

1 2 3 4 5

▼ 권한 설정

 그룹에 사용자 추가

 기존 사용자에서 권한 복사

 기존 정책 직접 연결

정책 생성



정책 필터 ▼

1 결과 표시

▶ 권한 경계 설정

기존 정책 직접 연결 > AmazonS3FullAccess 체크

사용자 추가

1 2 3 4 5



성공

아래에 표시된 사용자를 생성했습니다. 사용자 보안 자격 증명을 보고 다운로드할 수 있습니다. AWS Management Console 로그인을 위한 사용자 지침을 이메일로 보낼 수도 있습니다. 지금이 이 자격 증명을 다운로드할 수 있는 마지막 기회입니다. 하지만 언제든지 새 자격 증명을 생성할 수 있습니다.

AWS Management Console 액세스 권한이 있는 사용자가 <https://113353718176.signin.aws.amazon.com/console>에 로그인할 수 있습니다.

.csv 다운로드

	사용자	액세스 키 ID	비밀 액세스 키
▶	✓ mario	AKIARUZDIPGQ07V2TXG7	***** 표시

생성된 사용자의 key 정보를 저장해놓기 (사용자 생성 직후에만 다운로드 가능)