

# 포팅매뉴얼

## I. 빌드 및 배포

1. 실행환경
2. 설정 파일 목록 ( 경로 및 파일은 배포 과정 중 생성 될 수 있음 )
3. 설정 파일 및 환경 변수 정보
4. 포트 개방
5. SSL 인증서 발급
6. Jenkins에 Gitlab Webhook 연결
7. MySql 컨테이너 설정

## II. 외부 서비스

1. Dockerhub
  2. AWS S3
  3. KAKAO DEVELOPER
  4. OpenAI ChatGPT
  5. 기상청 단기예보 API
- 

## 목차

### I. 빌드 및 배포

1. 실행 환경
2. 설정 파일 목록
3. 설정 파일 및 환경 변수 정보
4. 포트 개방
5. SSL 인증서 발급
6. Jenkins에 Gitlab Webhook 연결

### II. 외부 서비스

1. Dockerhub
2. AWS S3
3. KAKAO DEVELOPER
4. OpenAI ChatGPT

## I. 빌드 및 배포

### 1. 실행환경

1. Server: AWS EC2 Ubuntu 20.04.6 LTS
2. Visual Studio Code: 1.81
3. IntelliJ IDEA: 2023.1.3 (Ultimate Edition) 17.0.7+10-b829.16 amd64
4. JVM: Zulu OpenJDK 11
5. Docker: 24.0.5
6. Node.js: 20.8.1 LTS
7. MySQL: 8.0.31
8. Jenkins: 2.428
9. Cloud Storage: AWS S3
10. Python: 3.10

## 11. Go: 1.21

## 2. 설정 파일 목록 ( 경로 및 파일은 배포 과정 중 생성 될 수 있음 )

### 1. React

- a. Dockerfile : ./frontend

### 2. Spring Boot

- a. Dockerfile : ./backend
- b. application.properties : ./backend/src/main/resources
- c. application-prod.properties : ./backend/src/main/resources
- d. application-local.properties : ./backend/src/main/resources

### 3. Fast API

- a. Dockerfile (rembg) : ./RemoveBackground
- b. Dockerfile (checkClothes) : ./ClothesComment

### 4. Go

- a. Dockerfile (recommendClothes) : ./RecommendOutfit

### 5. Docker

- a. docker-compose.yml : (Docker의 Jenkins 컨테이너 내) /var/jenkins\_home
- b. .env : (EC2 인스턴스 내) /home/ubuntu/config
- c. .env : (EC2 인스턴스 내) /home/ubuntu/frontconfig
- d. .env : (EC2 인스턴스 내) /home/ubuntu/commentconfig

### 6. Nginx

- a. default.conf : (EC2 인스턴스 내) /etc/nginx/sites-available

## 3. 설정 파일 및 환경 변수 정보

### 1. React

- a. Dockerfile :

```
FROM node:20.8.1
# http -> https 시 웹소켓 오류 때문에 ( 웹 소켓 사용시 지워야함 )
ENV WDS_SOCKET_PORT 0
WORKDIR /app
COPY package.json .
RUN npm install --force
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

### 2. Spring Boot

- a. Dockerfile :

```
FROM adoptopenjdk/openjdk11
EXPOSE 8081
ARG JAR_FILE=build/libs/moeutto-0.0.1-SNAPSHOT.jar
ADD ${JAR_FILE} app.jar
# Tomcat에서 blocking issue 때문에
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/app.jar"]
```

- b. application.properties :

```

server.servlet.context-path=/api
server.port=${SERVER_PORT}
spring.profiles.default=prod
spring.jackson.time-zone=Asia/Seoul
spring.jackson.serialization.fail-on-empty-beans=false

#jwt
jwt.secret = ${JWT_SECRET}

#S3
cloud.aws.credentials.accessKey=${S3_ACCESS_KEY}
cloud.aws.credentials.secretKey=${S3_SECRET_KEY}
cloud.aws.stack.auto=false

# AWS S3 Service bucket
cloud.aws.s3.bucket=${BUCKET_NAME}
cloud.aws.region.static=ap-northeast-2

# AWS S3 Bucket URL
cloud.aws.s3.bucket.url=https://s3.ap-northeast-2.amazonaws.com/${BUCKET_NAME}

#multipartfile max file size setting
spring.servlet.multipart.max-file-size=30MB
spring.servlet.multipart.max-request-size=30MB

#kakao
kakao.api.key = ${KAKAO_API_KEY}
kakao.secret = ${KAKAO_SECRET}

```

c. application-prod.properties :

```

# jdbc
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.username=${PROD_USERNAME}
spring.datasource.hikari.password=${MYSQL_ROOT_PASSWORD}
spring.datasource.url=${PROD_DATASOURCE_URL}
spring.jpa.hibernate.ddl-auto=create

#database
spring.jpa.hibernate.naming.implicit-strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
spring.jpa.hibernate.naming.physical-strategy=org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy
spring.data.web.pageable.one-indexed-parameters=true

#jpa query log
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.highlight_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect

#logger setting
logging.level.root=info
logging.level.com.ssafy.moeutto=debug

#spring security
spring.security.user.name=${PROD_SECURITY_USERNAME}
spring.security.user.password=${PROD_SECURITY_PASSWORD}

#prod python URL
python.check.request.url = ${PROD_CHECK_REQUEST_URL}

go.recommend.request.url = ${PROD_RECOMMEND_REQUEST_URL}

#prod kakao redirect
kakao.redirect.url = ${PROD_KAKAO_REDIRECT_URL}

```

d. application-local.properties :

```

# jdbc
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.username=${LOCAL_USERNAME}
spring.datasource.hikari.password=${LOCAL_PASSWORD}
spring.datasource.url=${LOCAL_DATASOURCE_URL}
spring.jpa.hibernate.ddl-auto=create

#database
spring.jpa.hibernate.naming.implicit-strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
spring.jpa.hibernate.naming.physical-strategy=org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy
spring.data.web.pageable.one-indexed-parameters=true

```

```
#jpa query log
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.highlight_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect

#logger setting
logging.level.root=info
logging.level.com.ssafy.moeutto=debug

#spring security
spring.security.user.name=${LOCAL_SECURITY_USERNAME}
spring.security.user.password=${LOCAL_SECURITY_PASSWORD}

#local python URL
python.check.request.url = ${LOCAL_CHECK_REQUEST_URL}

go.recommend.request.url = ${LOCAL_RECOMMEND_REQUEST_URL}

#local kakao redirect
kakao.redirect.url = ${LOCAL_KAKAO_REDIRECT_URL}
```

### 3. Fast API

#### a. Dockerfile (ClothesComment) :

```
FROM python:3.10
WORKDIR /comment

COPY ./requirements.txt /comment/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /comment/requirements.txt

COPY . /comment
EXPOSE 9011

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "9011"]
```

#### b. Dockerfile (RemoveBackground) :

```
FROM python:3.9
WORKDIR /rembg

COPY ./requirements.txt /rembg/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /rembg/requirements.txt

COPY . /rembg
EXPOSE 9010

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "9010"]
```

### 4. Go

#### a. Dockerfile (RecommendOutfit) :

```
FROM golang:1.21
RUN go env -w G0111MODULE=auto
WORKDIR /app
COPY . /app

RUN echo "$PWD"
RUN ls

WORKDIR /app/myapp/cmd/myapp

RUN echo "$PWD"
RUN ls

RUN go env -w G0111MODULE=auto
RUN go build -o main.go

EXPOSE 9000

CMD ["/main.go"]
```

## 5. Docker

### a. docker-compose.yml :

```
version: '3.8'
services:
  backend:
    image: pohangman/moeutto-back:latest
    build:
      context: ./workspace/moeutto-pipeline/backend
      dockerfile: Dockerfile

  frontend:
    image: pohangman/moeutto-front:latest
    environment:
      - WDS_SOCKET_PORT=0
    build:
      context: ./workspace/moeutto-pipeline/frontend
      dockerfile: Dockerfile

  rembg:
    image: pohangman/moeutto-remove:latest
    build:
      context: ./workspace/moeutto-pipeline/RemoveBackground
      dockerfile: Dockerfile

  comment:
    image: pohangman/moeutto-check:latest
    build:
      context: ./workspace/moeutto-pipeline/ClothesComment
      dockerfile: Dockerfile

  recommend:
    image: pohangman/moeutto-recommend:latest
    build:
      context: ./workspace/moeutto-pipeline/RecommendOutfit
      dockerfile: Dockerfile
```

### b. ./config/.env :

```
PROD_DATASOURCE_URL=jdbc:mysql://172.26.13.183:3306/moeutto?useUnicode=true&cha>
PROD_USERNAME=root
MYSQL_ROOT_PASSWORD=maridonghyeon
PROD_SECURITY_USERNAME=moeutto
PROD_SECURITY_PASSWORD=maridonghyeon
PROD_KAKAO_REDIRECT_URL=http://k9a604.p.ssafy.io/login-redirect
PROD_CHECK_REQUEST_URL=https://k9a604.p.ssafy.io/clothes
PROD_RECOMMEND_REQUEST_URL=https://k9a604.p.ssafy.io/recommend
KAKAO_SECRET=${ KAKAO_SECRET_KEY }
KAKAO_API_KEY=${ KAKAO_API_KEY }
WDS_SOCKET_PORT=0
JWT_SECRET=${ JWT_SECRET }
S3_ACCESS_KEY=${ S3_ACCESS_KEY }
S3_SECRET_KEY=${ S3_SECRET_KEY }
BUCKET_NAME=moeutto-bucket
SERVER_PORT=8081
```

### c. ./frontconfig/.env

```
SKIP_PREFLIGHT_CHECK=true
REACT_APP_REDIRECT_URL=http://k9a604.p.ssafy.io/login-redirect
REACT_APP_API=https://k9a604.p.ssafy.io
REACT_APP_AI=https://k9a604.p.ssafy.io
REACT_APP_KAKAO_CLIENT_ID=${ KAKAO_DEVELOPER_CLIENT_ID }
REACT_APP_LOGOUT_REDIRECT_URL=http://k9a604.p.ssafy.io/logout-redirect
```

### d. ./commentconfig/.env

```
apikey=${your chat gpt api key}
```

### e. mysql ( Container )

```
docker run --name mysql -e MYSQL_ROOT_PASSWORD=${PROD_PASSWORD} -d -p 3306:3306 mysql:latest --character-set-server=utf8 --col
```

## 6. Nginx

### a. default.conf :

```
server {

    autoindex_localtime on;

    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/j9a410.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j9a410.p.ssafy.io/privkey.pem;

    server_name j9a410.p.ssafy.io;

    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-Proto https;
    proxy_headers_hash_bucket_size 512;
    proxy_redirect off;

    # Websockets
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

    location / {
        proxy_pass http://localhost:3000;
        proxy_connect_timeout 300s;
        proxy_read_timeout 600s;
        proxy_send_timeout 600s;
        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
        #mixed error
        add_header Content-Security-Policy "upgrade-insecure-requests";
    }

    location /api {
        proxy_pass http://localhost:8081;
    }

    location /ws {
        proxy_pass http://localhost:3000;
    }

    location /predict {
        proxy_pass http://localhost:9010;
    }

    location /clothes {
        proxy_pass http://localhost:9011;
    }

    location /recommend {
        proxy_pass http://localhost:9000;
    }

}
server {
    listen 80;
    server_name j9a410.p.ssafy.io;

    return 301 https://$host$request_uri;
}
```

## 7. Jenkins Pipeline Script

```
pipeline {
    agent any
    tools{
        gradle 'gradle'
        nodejs 'nodejs'
    }
}
```

```

environment {
    DOCKER_REPO_BACK = "pohangman/moeutto-back"
    DOCKER_REPO_FRONT = "pohangman/moeutto-front"
    DOCKER_REPO_REMBG = "pohangman/moeutto-remove"
    DOCKER_REPO_COMMENT = "pohangman/moeutto-check"
    DOCKER_REPO_RECOMMEND = "pohangman/moeutto-recommend"
    DOCKER_CREDENTIAL = credentials('dockerhub-credential')
    BACK_CONTAINER = "moeutto-back"
    FRONT_CONTAINER = "moeutto-front"
    REMBG_CONTAINER = "moeutto-rembg"
    SSH_CONN = "ubuntu@k9a604.p.ssafy.io"
    ENV_DIR = "./config/.env"
    FRONT_ENV_DIR = "./frontconfig/.env"
    COMMENT_ENV_DIR = "./commentconfig/.env"
    BACK_PORT = "8081"
    FRONT_PORT = "3000"
    REMBG_PORT = "9010"
    COMMENT_PORT = "9011"
    RECOMMEND_PORT = "9000"
}

stages{
    stage('Git Clone') {
        steps {
            sh "#### Git Clone Start ####"
            git branch: 'develop', credentialsId: 'gitlabId', url: 'https://lab.ssafy.com/s09-final/S09P31A604.git'
            sh "#### Git Clone Success ####"
        }
    }
    stage('Build Backend'){
        steps{
            dir('backend'){
                sh "echo '#### Build Backend Start ####'"
                sh "chmod +x gradlew"
                sh "./gradlew clean compileJava bootJar"
                sh "echo '#### Build Backend Success ####'"
            }
        }
    }
    stage('Build Frontend'){
        steps{
            sh "echo '#### Build Frontend Start ####'"
            dir('frontend'){
                sh "npm install --force"
                sh "CI=false npm run build"
            }
            sh "echo '#### Build Frontend Success ####'"
        }
    }
    stage('Build Images'){
        steps{
            sh "echo '#### Build Image Start ####'"
            sh "docker-compose build"
            sh "echo '#### Build Image Success ####'"
        }
    }
    stage('Push Images'){
        steps{
            sh "echo '#### Push Images Start ####'"
            sh "echo $DOCKER_CREDENTIAL_PSW | docker login -u $DOCKER_CREDENTIAL_USR --password-stdin"
            sh "docker push pohangman/moeutto-back:latest"
            sh "docker push pohangman/moeutto-front:latest"
            sh "docker push pohangman/moeutto-remove:latest"
            sh "docker push pohangman/moeutto-check:latest"
            sh "docker push pohangman/moeutto-recommend:latest"
            sh "echo '#### Push Images Success ####'"
        }
    }
    stage('Deploy Backend Server'){
        steps{
            sh "echo '#### Deploy Backend Server Start ####'"
            sshagent(credentials:['ec2-user']){
                sh "pwd"
                sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker rm -f moeutto-back'"
            }
        }
    }
}

```





```

sudo apt-get install ufw

sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https

sudo ufw allow 8080
sudo ufw allow 8080/tcp
sudo ufw allow 3000
sudo ufw allow 3000/tcp
sudo ufw allow 8081
sudo ufw allow 8081/tcp
sudo ufw allow 8000
sudo ufw allow 8000/tcp
sudo ufw allow 80
sudo ufw allow 20
sudo ufw allow 80/tcp
sudo ufw allow 22/tcp
sudo ufw allow 443/tcp
sudo ufw allow 9000
sudo ufw allow 9000/tcp
sudo ufw allow 9010
sudo ufw allow 9010/tcp
sudo ufw allow 9011
sudo ufw allow 9011/tcp
sudo ufw allow 3306
sudo ufw allow 3306/tcp

```

## 5. SSL 인증서 발급

```

sudo apt-get install letsencrypt
sudo apt-get install certbot python3-certbot-nginx
sudo certbot --nginx
# 이메일 입력
# 약관 동의 - Y
# 이메일 발송 동의 - Y or N
# 도메인 입력

```

## 6. Jenkins에 Gitlab Webhook 연결

1. Jenkins 대시보드>Jenkins 관리>Plugins
2. Gitlab 플러그인 설치
3. Gitlab API Token 발급

**User Settings**

- Profile
- Account
- Applications
- Chat
- Access Tokens**
- Emails
- Password
- Notifications
- SSH Keys
- GPG Keys
- Preferences
- Comment Templates
- Active Sessions
- Authentication Log

**Personal Access Tokens**

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

**Add a personal access token**

Enter the name of your application, and we'll return a unique personal access token.

**Token name**

gitlab-jenkins-token

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

**Expiration date**

2023-09-17

**Select scopes**

Scopes set the permission levels granted to the token. [Learn more.](#)

- ☒ **api**  
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ **read\_api**  
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ **read\_user**  
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- ☐ **read\_repository**  
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- ☐ **write\_repository**  
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

**Create personal access token**

4. 발급 받은 Token을 Jenkins에 등록

5. Jenkins관리>시스템 설정으로 이동 후, gitlab 경로와 위에서 등록한 API Token Credential을 사용하여 Gitlab과 Jenkins를 연동

6. 파이프라인 프로젝트 생성 및 Webhook 경로 확인

- CI/CD를 수행할 Pipeline 아이템 생성

## Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://example.com/project/Dasoni> ?

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 ?

### 7. Pipeline Secret Token 발급

- Build Triggers 메뉴 아래 '고급' 탭 클릭

고급 ▼

✎ Edited

- Generate를 통해 Secret Token 발급 ( 꼭, 따로 저장하도록 해야함 )

Secret token ?

Generate

### 8. Repository Webhook Event 설정

- Gitlab의 Repository에서 Settings → Webhooks
- Webhook Event는 Repository 별로 설정해주어야 함
- 앞서 기억해둔 URL과 Secret을 입력해주고 원하는 Trigger를 체크해준다

#### URL

http://example.com/trigger-ci.json

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

☒ Show full URL

☐ Mask portions of URL

Do not show sensitive data such as tokens in the UI.

#### Secret token

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

### 9. 등록 확인

- 등록 후, Test를 통해 연동 확인 테스트를 할 수 있다

☐ Wiki page events  
A wiki page is created or updated.

☐ Deployment events  
A deployment starts, finishes, fails, or is canceled.

☐ Feature flag events  
A feature flag is turned on or off.

☐ Releases events  
A release is created or updated.

#### SSL verification

☒ Enable SSL verification

Add webhook

Push events  
Tag push events  
Issues events  
Confidential issues events  
Comments  
Confidential comments  
Merge request events  
Job events  
Pipeline events  
Wiki page events  
Deployment events  
Feature flag events  
Releases events

Project Hooks (1)

Push events SSL Verification: enabled

Test

Edit

Delete

- Test를 하고 나서 이와 같이 표시되면 연동이 성공 된 것이다

CI/CD > main > team 00 > Manage labeling service > Webhook settings

Hook executed successfully: HTTP 200

## 7. MySql 컨테이너 설정

### 1. Mysql 컨테이너 내부 접속

```
$ docker exec -it --user root ${your mysql container name} bash
```

### 2. 유저 인증

```
mysql -u root -p
```

```
Enter password: ${PROD_PASSWORD}
```

### 3. 스키마 만들기

```
create schema moeutto;
```

### 4. 카테고리 추가

```
insert into large_category values("002", "아우터"), ("001", "상의"),
("003", "하의"), ("011", "아이템");

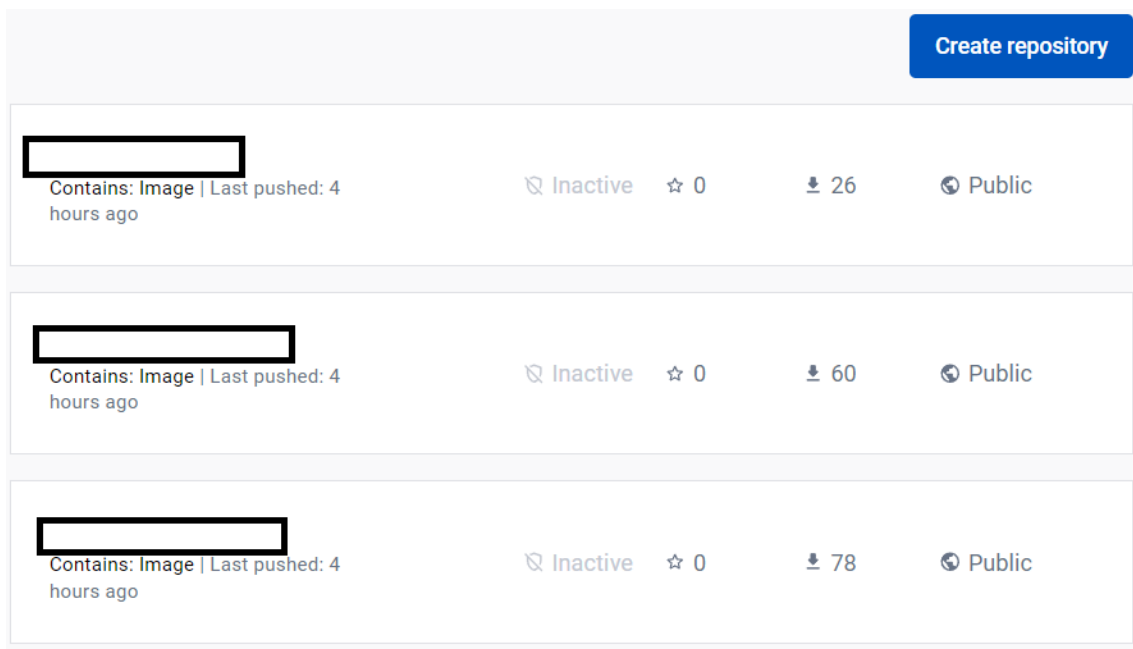
insert into middle_category values("002012", "패딩", "002"), ("002007", "코트", "002"),
("002004", "자켓", "002"), ("001005", "맨투맨", "001"), ("001004", "후드", "001"),
("001001", "반팔", "001"), ("003002", "청바지", "003"), ("003009", "반바지", "003"),
("003004", "카고팬츠", "003"), ("011011", "장갑", "011"), ("011010", "목도리", "011"),
("011006", "귀마개", "011");
```

## II. 외부 서비스

### 1. Dockerhub

#### 1. Dockerhub Repository 만들기

##### a. Dockerhub 로그인 후, 우측 상단 Create repository 클릭



##### b. Repository Name을 채우고 Create ( Private는 Free Tier 에선 하나 밖에 생성하지 못함 )

## Create repository

Namespace

Repository Name \*

user name


your\_repository\_name


Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

### Visibility

Using 0 of 1 private repositories. [Get more](#)

☒
**Public**


☐
**Private**


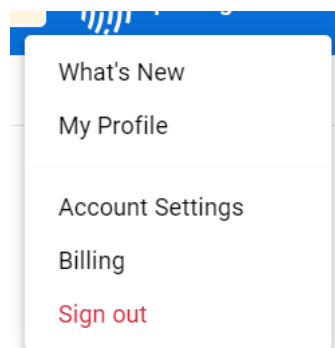
Appears in Docker Hub search results

Only visible to you

Cancel

Create

- c. Jenkins Pipeline Script에서 Dockerhub를 사용할 수 있도록 Dockerhub Access Token 발급 및 등록
- i. Account Settings 클릭



- ii. Security 메뉴 클릭 후, New Access Token 버튼 클릭

General  
**Security**  
Default Privacy  
Notifications  
Convert Account  
Deactivate Account

### Access Tokens

Tokens marked **AUTO-GENERATED** are created on your behalf by Docker Desktop for the CLI to use for authentication. You can have a maximum of 5 auto-generated tokens associated with your account.

Last Used	Created	
Oct 04, 2023 09:09:55	Sep 19, 2023 13:02:06	⋮
Sep 15, 2023 17:35:03	Aug 11, 2023 16:00:44	⋮
Never	Sep 15, 2023 15:46:39	⋮
Never	Aug 09, 2023 15:32:52	⋮

- iii. Access Token Description 작성 후, 부여하고 싶은 액세스 권한 설정 후, Generate 클릭

## New Access Token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more](#)

Access Token Description \*

jenkins-dockerhub-credential

Access permissions

Read, Write, Delete

Read, Write, Delete tokens allow you to manage your repositories.

Cancel

Generate

- iv. 발급된 Access Token을 꼭 저장해두도록 한다

## Copy Access Token

When logging in from your Docker CLI client, use this token as a password. [Learn more](#)

ACCESS TOKEN DESCRIPTION

jenkins-dockerhub-credential

ACCESS PERMISSIONS

Read, Write, Delete

To use the access token from your Docker CLI client:

1. Run `docker login -u Dockerhub username`
2. At the password prompt, enter the personal access token.

access token



**WARNING:** This access token will only be displayed once. It will not be stored and cannot be retrieved. Please be sure to save it now.

Copy and Close

- v. Jenkins의 Dashboard>Jenkins관리>Credentials 에서 Add credentials>Username with password 선택
- vi. 필요한 내용을 채운 뒤, Create 클릭 후 사용

Kind

Username with password

Scope ?

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

Username ?

Dockerhub Username

☐ Treat username as secret ?

Password ?

발급받은 Access Token

ID ?

Pipeline Script 등에서 사용할 아이디 (즉, 변수명)

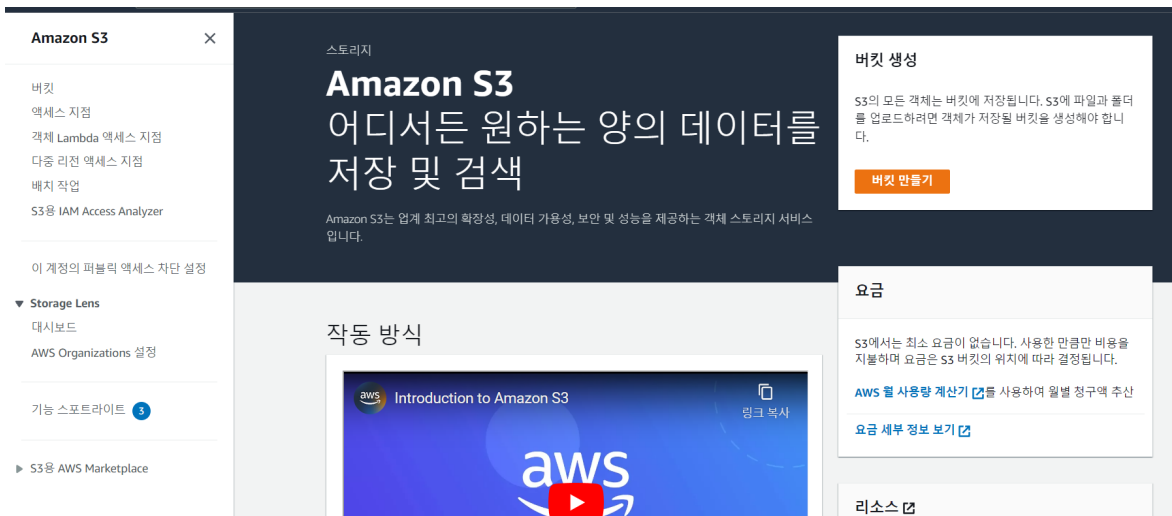
Description ?

Create

## 2. AWS S3

1. <https://aws.amazon.com/ko/> 로 접속
2. 로그인 후, 서비스 창에서 S3 선택
3. 버킷 만들기 클릭





#### 4. 버킷 이름과 리전 설정

- a. 버킷 이름은 고유한 값이어야 함

## 버킷 만들기 정보

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

### 일반 구성

버킷 이름

버킷 이름은 글로벌 네임스페이스 내에서 고유해야 하며 버킷 이름 지정 규칙을 따라야 합니다. [버킷 이름 지정 규칙 보기](#)

AWS 리전

아시아 태평양(서울) ap-northeast-2

기존 버킷에서 설정 복사 - 선택 사항

다음 구성의 버킷 설정만 복사됩니다.

버킷 선택

#### 5. 퍼블릭 액세스 설정 후 버킷 만들기 클릭

- 외부에 S3를 공개할 경우 모든 퍼블릭 액세스 차단에 체크를 해제, 공개하지 않는다면 체크

## 이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(엑세스 제어 목록), 버킷 정책, 액세스 지정 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(엑세스 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(엑세스 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 지정 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

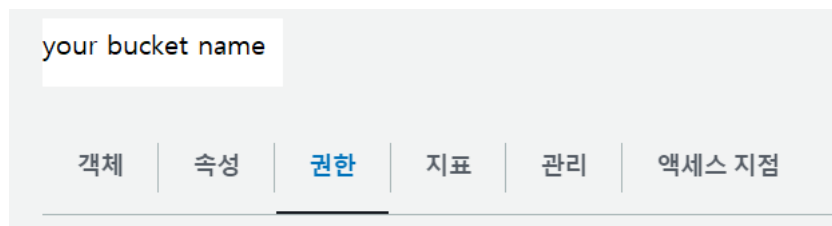
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지정 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

#### ☒ 임의의 퍼블릭 버킷 또는 액세스 지정 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 지정에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.

## 6. 버킷 정책 생성

### a. 생성된 버킷 클릭 → 권한 메뉴 클릭 → 버킷 정책 메뉴의 편집 클릭



#### 버킷 정책

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

편집

삭제

### b. 버킷 ARN을 복사한 뒤, AWS 정책 생성기로 접속( <http://awspolicygen.s3.amazonaws.com/policygen.html> ) 하여 정책 타입과 상태를 추가한 뒤, 정책 생성

### 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 VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy IAM 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

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) your bucket ARN

ARN should follow the following format: `arn:aws:s3:::${BucketName}/${KeyName}`.  
Use a comma to separate multiple values.

[Add Conditions \(Optional\)](#)

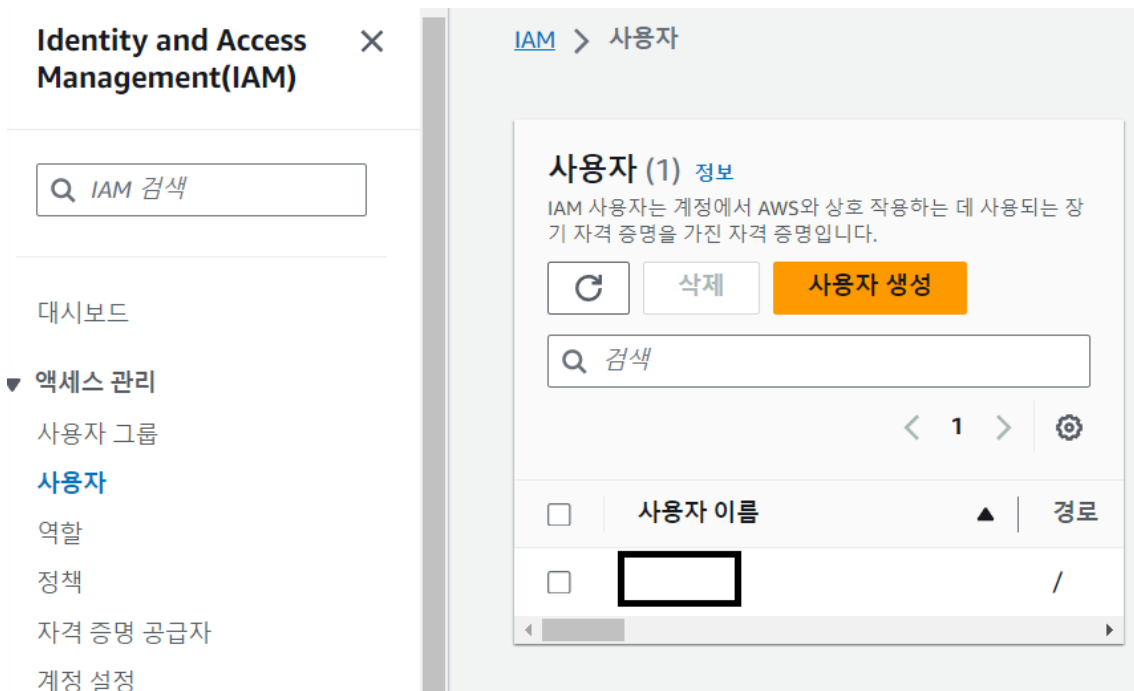
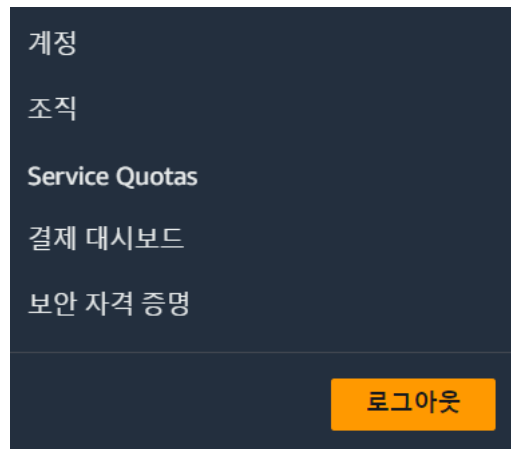
Add Statement

- c. 생성된 정책을 복사 후, 버킷의 정책란에 붙여 넣기 후, 변경 사항 저장

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "[redacted]",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::[redacted]/*",
        "arn:aws:s3:::[redacted]"
      ]
    }
  ]
}
```

7. Spring Boot 서버에서 S3를 사용할 수 있도록, IAM에서 사용자 등록

- a. 보안 자격 증명 → 액세스 관리 하위 사용자 → 사용자 생성



b. 사용자 이름 작성 후 다음

1/3 단계

## 사용자 세부 정보 지정

## 사용자 세부 정보

사용자 이름

사용자 이름은 최대 64자까지 가능합니다. 유효한 문자: A~Z, a~z, 0~9 및 +, =, ., @, \_ -(하이픈)

☐ AWS Management Console에 대한 사용자 액세스 권한 제공 - 선택 사항사용자에게 콘솔 액세스 권한을 제공하는 경우 IAM Identity Center에서 액세스를 관리하는 것은 [모범 사례](#)입니다.

**i** 이 IAM 사용자를 생성한 후 액세스 키 또는 AWS CodeCommit이나 Amazon Keyspaces에 대한 서비스별 보안 인증 정보를 통해 프로그래밍 방식 액세스를 생성할 수 있습니다. [자세히 알아보기](#)

취소

다음

c. 권한 설정에서 직접 정책 연결 선택 후, AmazonS3FullAccess 검색 후 선택 → 다음 클릭 → 사용자 생성 클릭

2/3 단계

## 권한 설정

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 직무별로 사용자의 권한을 관리하려면 그룹을 사용하는 것이 좋습니다. [자세히 알아보기](#)

## 권한 옵션

☐ 그룹에 사용자 추가

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 그룹을 사용하여 직무별로 사용자 권한을 관리하는 것이 좋습니다.

☐ 권한 복사

기존 사용자의 모든 그룹 멤버십, 연결된 관리형 정책 및 인라인 정책을 복사합니다.

☒ 직접 정책 연결

관리형 정책을 사용자에게 직접 연결합니다. 사용자에게 연결하는 대신, 정책을 그룹에 연결한 후 사용자를 적절한 그룹에 추가하는 것이 좋습니다.

**권한 정책 (1/1132)** 🔄 정책 생성 ↗

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

필터링 기준 유형

🔍 S3 ✕ 모든 유형 ▼ 12 개 일치

< 1 > ⚙️

	정책 이름 ↗	유형 ▼	연결된 ... ▼
<input type="checkbox"/>	AmazonDMSRedsh...	AWS 관리형	0
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS 관리형	1

d. 생성한 사용자 클릭 → 보안 자격 증명 → 액세스 키 → 액세스 키 만들기

i. CLI 선택 후 다음

## 액세스 키 모범 사례 및 대안 정보

보안 개선을 위해 액세스 키와 같은 장기 자격 증명을 사용하지 마세요. 다음과 같은 사용 사례와 대안을 고려하세요.

사용 사례

☒ **Command Line Interface(CLI)**

AWS CLI를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

ii. 설명 태그 값 입력 후 액세스 키 생성

**2/3 단계**

## 설명 태그 설정 - 선택 사항 정보

이 액세스 키에 대한 설명은 이 사용자에게 태그로 연결되고, 액세스 키와 함께 표시됩니다.

**설명 태그 값**

이 액세스 키의 용도와 사용 위치를 설명합니다. 좋은 설명은 나중에 이 액세스 키를 자신있게 교체하는 데 유용합니다.

최대 256자까지 가능합니다. 허용되는 문자는 문자, 숫자, UTF-8로 표현할 수 있는 공백 및 \_ . : / = + - @입니다.

취소 이전 액세스 키 만들기

iii. 액세스 키와 비밀 액세스 키 반드시 다른 곳에 저장

- 액세스 키는 다시 확인 가능하지만 비밀 액세스 키는 다시 확인하기 힘들

3/3 단계

## 엑세스 키 검색 정보

### 엑세스 키

분실하거나 잊어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

엑세스 키

비밀 액세스 키



e. 발급받은 액세스 키와 비밀 액세스 키를 사용

### 3. KAKAO DEVELOPER

1. <https://developers.kakao.com/console/app> 접속
2. 내 애플리케이션 > 애플리케이션 추가하기

전체 애플리케이션 (2)

애플리케이션 이름



애플리케이션 추가하기

3. 애플리케이션 추가 후, 내 애플리케이션 > 앱 설정 > 요약 정보 에서 앱 키 확인

### 앱 키

네이티브 앱 키	NATIVE APP KEY
REST API 키	REST API KEY
JavaScript 키	JavaScript KEY
Admin 키	Admin KEY

4. 내 애플리케이션 > 제품 설정 > 카카오 로그인 이동 후, Redirect URI 설정

## Redirect URI

[삭제](#)[수정](#)

Redirect URI

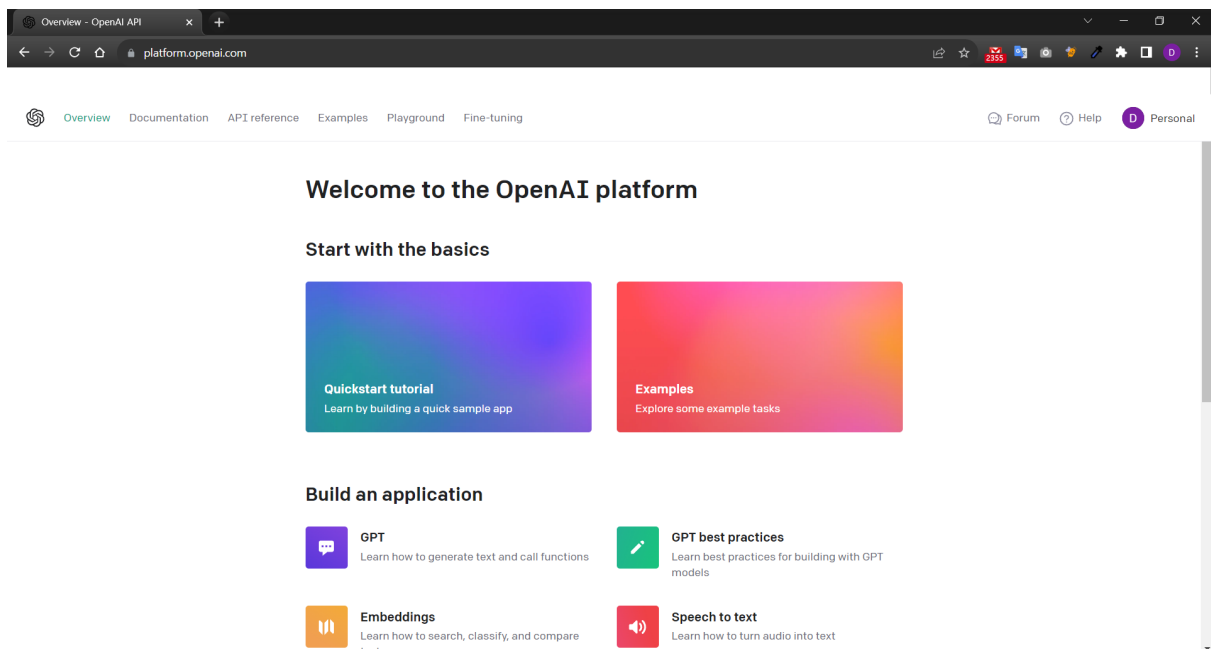
YOUR LOGIN REDIRECT URI

YOUR LOGOUT REDIRECT URI

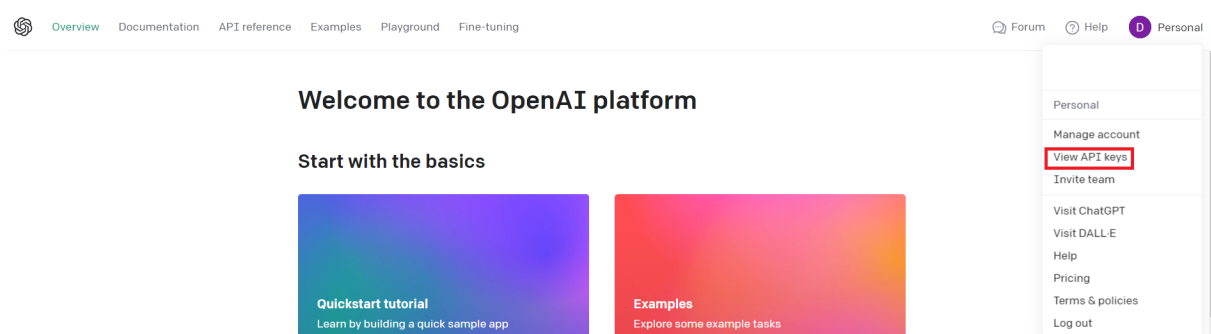
- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

## 4. OpenAI ChatGPT

### 1. [platform.openai.com](https://platform.openai.com) 접속

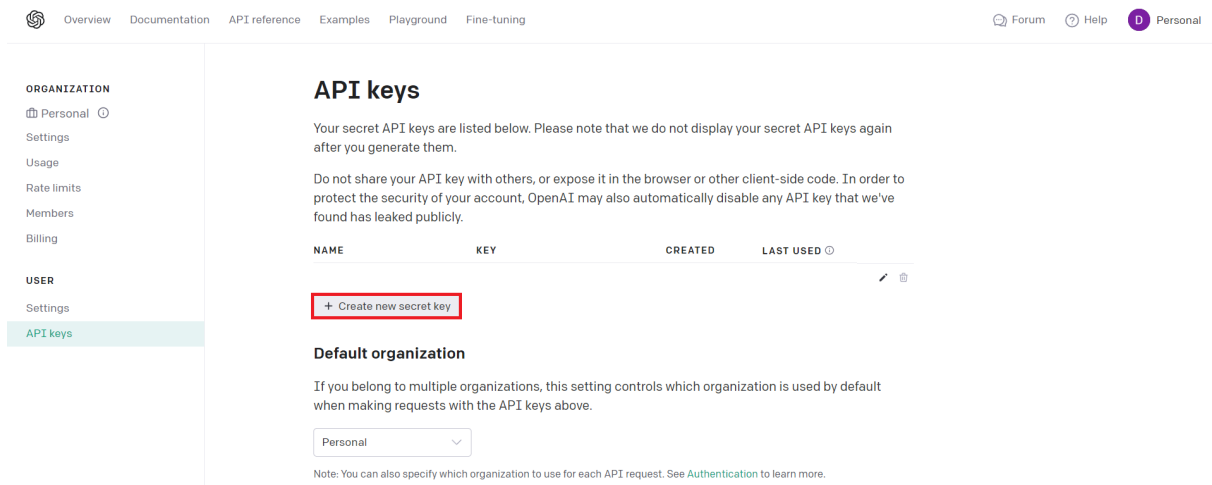


### 2. View API keys 클릭

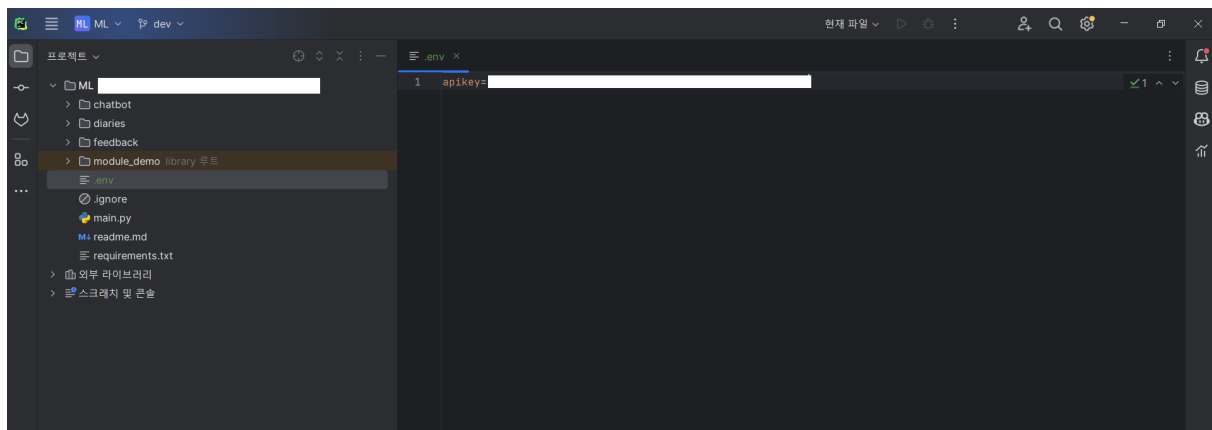


### 3. Create new secret key



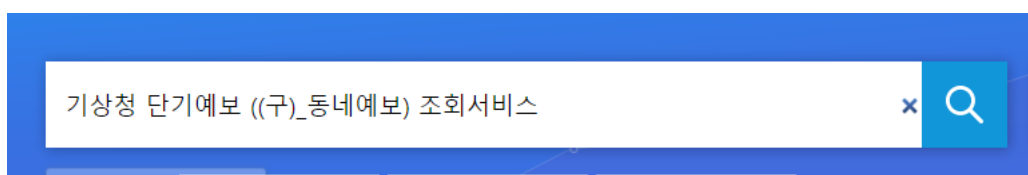


#### 4. ClothesComment 디렉토리에 .env 파일 생성 후 'apikey' 변수 초기화



### 5. 기상청 단기예보 API

1. <https://www.data.go.kr/index.do> 접속
2. 기상청 단기예보 ((구)\_동네예보) 조회서비스 검색



#### 3. 활용 신청

##### 오픈API 상세

**XML** **JSON** 기상청\_단기예보 ((구)\_동네예보) 조회서비스

초단기실황, 초단기예보, 단기((구)동네)예보, 예보버전 정보를 조회하는 서비스입니다. 초단기실황정보는 예보 구역에 대한 대표 AWS 관측값을, 초단기예보는 예보시점부터 6시간까지의 예보를, 단기예보는 예보기간을 끝까지 확장 및 예보단위를 상세화(3시간→1시간)하여 시공간적으로 세분화한 예보를 제공합니다.

👍 139
❤️ 18
🔖 관심

URL 복사

[활용신청](#)

#### 4. 인증 키 복사 후 사용

기본정보

데이터명	기상청_단기예보 ((구)_동네예보) 조회서비스	상세설명	
서비스유형	REST	심의여부	자동승인
신청유형	개발계정   활용신청	처리상태	승인
활용기간	2023-10-19 ~ 2025-10-19		

서비스정보

데이터포맷	JSON+XML
End Point	http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0
API 환경 또는 API 호출 조건에 따라 인증키가 적용되는 방식이 다를 수 있습니다. 포털에서 제공되는 <b>Encoding/Decoding</b> 된 인증키를 적용하면서 구동되는 키를 사용하시기 바랍니다. * 향후 포털에서 더 명확한 정보를 제공하기 위해 노력하겠습니다.	
일반 인증키 (Encoding)	인증키 ↵
일반 인증키 (Decoding)	인증키