

포팅 매뉴얼

1. 개발 환경

형상 관리

- GITLAB

이슈 관리

- JIRA
 - 매주 목표량을 설정하여 스프린트 진행
 - In-Progress 및 Done으로 각자 지금 하고있는 업무와 완료된 업무를 공유

소통 관리

- Mattermost
 - 프로젝트 자료 공유
 - 의견 작성
 - 미완료 업무 공유
- Notion
 - 요구사항 명세서 작성 및 공유
 - API 명세서 작성 및 공유
 - 컨벤션 문서 관리

UI/UX

- Figma

IDE

- IntelliJ IDEA 2022.3.2
- Visual studio code 1.75

DATABASE

- MYSQL 8.0.31
- MYSQL workbench 8.0 CE

SERVER

- AWS EC2
 - UBUNTU 20.04 LTS
 - MobaXterm_Personal_22.3.exe
 - DOCKER 20.10.23
 - NGINX 1.18.0
 - S3

협업툴

- SWAGGER 2.9.2
- POSTMAN for Windows Version 10.9.4

BACK-END

- Java Open-JDK azul 11
- SpringBoot Gradle 2.7.7
 - Spring Data JPA
 - Lombok
 - Swagger 2.9.2
- Python 3.9.13
- Django 4.1.7
- jupyter notebook 6.4.12
- anaconda 1.11.0

FRONT-END

- React 18.2.0
 - react-dom 18.2.0
 - react-icons 4.8.0
 - react-responsive 9.0.2
 - react-easy-swipe 0.0.23
 - react-icons 4.8.0
 - react-responsive 9.0.2
 - react-simple-star-rating 5.1.7
 - react-useanimations 2.10.0
- next 13.2.4
- emotion
 - @emotion/css 11.10.6
 - @emotion/react 11.10.6
 - @emotion/styled 11.10.6
- apexcharts 3.37.1
- "axios 1.3.4",

- "axios 1.3.4",
- "jotai 2.0.3",
- "lodash 4.17.21",
- universal-cookie 4.0.4
- "cookie 0.5.0"

2. 개발 설정

인텔리제이 - application.yml 설정

- application.yml을 개인 정보 보호를 위해 서브 모듈로 관리중

```
spring:
  mvc:
    pathmatch:
      matching-strategy: ANT_PATH_MATCHER
  profiles.active: local
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://j8d203.p.ssafy.io:3306/emosaac?useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false&allowPublicKeyRetrieval=true
    username: {mysql 사용자 이름}
    password: {mysql 비밀번호}

  hikari:
    pool-name: jpa-hikari-pool
    maximum-pool-size: 5
    jdbc-url: ${spring.datasource.url}
    username: ${spring.datasource.username}
    password: ${spring.datasource.password}
    driver-class-name: ${spring.datasource.driver-class-name}
    data-source-properties:
      rewriteBatchedStatements: true

# JPA
jpa:
  generate-ddl: true
  hibernate:
    ddl-auto: update
    show-sql: true
  properties:
    hibernate:
      jdbc:
        time_zone: Asia/Seoul
      dialect: org.hibernate.dialect.MySQL8Dialect
      hbm2ddl.import_files_sql_extractor: org.hibernate.tool.hbm2ddl.MultipleLinesSqlCommandExtractor
      current_session_context_class: org.springframework.orm.hibernate5.SpringSessionContext
      default_batch_fetch_size: ${chunkSize:100}
      jdbc.batch_size: 20
      order_inserts: true
      order_updates: true
      format_sql: true

servlet:
  multipart:
    max-file-size: 50MB #50MB
    max-request-size: 50MB
    enabled: true

# Security OAuth
security:
  oauth2.client:
    registration:
      kakao:
        clientId: '13041c7bbff9b85dc69c4ae6023649f3'
        clientSecret: 't8gXZBCBzfddkdH6HZD0bl5ZV7bFiN4'
        clientAuthenticationMethod: post
        authorizationGrantType: authorization_code
        redirect-uri: 'http://j8d203.p.ssafy.io:8081/oauth2/callback/kakao'
```

```

    scope:
      - profile_nickname
      - profile_image
      - account_email
    clientName: Kakao

  naver:
    client-id: '9xgq3GBZflzSRp5DSCWu'
    client-secret: 'LiQcUfDv9n'
    # redirect-uri: 'https://j8d203.p.ssafy.io/oauth2/callback/naver' #배포 프론트, nginx
    redirect-uri: 'http://j8d203.p.ssafy.io:8081/oauth2/callback/naver' #배포 프론트

    authorization-grant-type: authorization_code
    scope: name, email, profile_image
    client-name: Naver

  provider:
    kakao:
      authorizationUri: https://kauth.kakao.com/oauth/authorize
      tokenUri: https://kauth.kakao.com/oauth/token
      userInfoUri: https://kapi.kakao.com/v2/user/me
      userNameAttribute: id
    naver:
      authorization_uri: https://nid.naver.com/oauth2.0/authorize
      token_uri: https://nid.naver.com/oauth2.0/token
      user-info-uri: https://openapi.naver.com/v1/nid/me
      user_name_attribute: response

  mail:
    host: smtp.gmail.com
    port: 587
    username: {구글 메일 아이디}
    password: {구글 앱 비밀번호}
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true

  jwt:
    header: Authorization
    secret: 23kljgljewlfjldjsfklj3lkjklegjkdjkqjljslfjdkfjdkajgk3ejhkgdxaj3ljljelkdjl3hleflkajdfkljhadfasdfadsfasg4ehhegfdddkljeljdjfld
    token-validity-in-seconds: 86400
    access:
      expire-length: 864000
    refresh:
      expire-length: 864000
    token:
      secret-key: 23kljgljewlfjldjsfklj3lkjklegjkdjkqjljslfjdkfjdkajgk3ejhkgdxaj3ljljelkdjl3hleflkajdfkljhadfasdfadsfasg4ehhegfdddkljeljdjfld

  app:
    auth:
      tokenSecret: 926D96C90030DD58429D2751AC1BDBBC23kljgljewlfjldjsfklj3lkjklegjkdjkqjljslfjdkfjdkajgk3ejhkgdxaj3ljljelkdjl3hleflkajdfkljhadfasdfadsfasg4ehhegfdddkljeljdjfld
      tokenExpirationMsec: 864000000
    oauth2:
      authorizedRedirectUris:
        - http://j8d203.p.ssafy.io:8081/oauth/token
        - https://j8d203.p.ssafy.io/oauth/token
        - http://localhost:3000/oauth/redirect
        - https://j8d203.p.ssafy.io/oauth2/redirect
        - http://localhost:3000/oauth2/redirect
        - http://j8d203.p.ssafy.io:3000/oauth2/redirect
        - https://j8d203.p.ssafy.io
        - myandroidapp://oauth2/redirect
        - myiosapp://oauth2/redirect
    cors:
      allowedOrigins: http://localhost:3000, https://j8d203.p.ssafy.io:8081, http://localhost:8080, http://localhost:8081, http://j8d203

# multipart 용량 지정
servlet:
  multipart:
    max-file-size: 50MB
    max-request-size: 50MB
    enabled: true

cloud:
  aws:
    credentials:
      access-key: {s3 사용자 access-key}
      secret-key: {s3 사용자 secret-key}
    s3:
      bucket: emosaacbucket
      region:

```

```

        static: ap-northeast-2
        stack:
            auto: false

server:
    port: 8081

```

인텔리제이 - gradle.build 설정

```

buildscript {
    ext {
        queryDslVersion = "5.0.0"
    }
}
plugins {
    id 'java'
    id 'org.springframework.boot' version '2.7.9'
    id 'io.spring.dependency-management' version '1.0.15.RELEASE'

    //querydsl 추가
    id 'com.ewerk.gradle.plugins.querydsl' version '1.0.10'
}

group = 'com.emosaac'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    //JPA를 사용하기 위한 스타터 라이브러리
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'

    //OAuth 2.0 클라이언트를 사용하기 위한 스타터 라이브러리
    implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'

    //스프링 시큐리티를 사용하기 위한 스타터 라이브러리
    implementation 'org.springframework.boot:spring-boot-starter-security'

    //lombok설정
    compileOnly 'org.projectlombok:lombok'

    //스프링 부트 개발 도구 라이브러리 (코드 수정시 자동으로 재시작 등)
    developmentOnly 'org.springframework.boot:spring-boot-devtools'

    //인메모리 데이터베이스인 H2를 사용하기 위한 라이브러리
    runtimeOnly 'com.h2database:h2'

    //스프링 웹 애플리케이션을 개발하기 위한 스타터 라이브
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    testImplementation 'org.springframework.security:spring-security-test'

    //스프링 부트가 바로 꺼지는 문제 해결을 위해 추가
    implementation 'org.springframework.boot:spring-boot-starter-web'

    //querydsl 추가
    implementation "com.querydsl:querydsl-jpa:${queryDslVersion}"
    implementation "com.querydsl:querydsl-apt:${queryDslVersion}"

    // https://mvnrepository.com/artifact/mysql/mysql-connector-java
    implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.22'

    //코드 수정시 서버 재시작
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'

    //스웨거 설정
    implementation 'io.springfox:springfox-swagger2:2.9.2'

```

```

implementation 'io.springfox:springfox-swagger-ui:2.9.2'

//validation설정
implementation 'org.springframework.boot:spring-boot-starter-validation'

//JSON Web Token(JWT)을 사용하기 위한 자바 라이브러리
implementation 'io.jsonwebtoken:jjwt-api:0.11.2'

//Java Architecture for XML Binding(JAXB) API
implementation 'jakarta.xml.bind:jakarta.xml.bind-api:2.3.2'

//JWT API의 구현체
runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.2'

//JWT를 Jackson(자바용 JSON 라이브러리)과 통합하기 위한 라이브러리
runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.11.2'

//JSON 데이터를 처리하기 위한 자바 라이브러리
implementation 'com.google.code.gson:gson:2.8.6'

//스프링 부트 구성 클래스의 메타데이터를 생성하는 라이브러리
annotationProcessor "org.springframework.boot:spring-boot-configuration-processor"

//AWS 서비스를 사용하기 위한 스프링 부트 스타터
implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'

// 하이버네이트(Hibernate)를 사용하여 Java 객체를 SQL 유형에 매핑하기 위한 라이브러리
implementation 'com.vladmihalcea:hibernate-types-55:2.20.0'

//Google Cloud Vision API를 사용하기 위한 자바 라이브러리
implementation 'com.google.cloud:google-cloud-vision:3.11.0'

//JavaMail API를 사용하여 이메일을 보내기 위한 스프링 부트 스타터
implementation 'org.springframework.boot:spring-boot-starter-mail'

}

tasks.named('test') {
    useJUnitPlatform()
}

//querydsl 추가
def querydslDir = "$buildDir/generated/querydsl"

querydsl {
    jpa = true
    querydslSourcesDir = querydslDir
}
sourceSets {
    main.java.srcDir querydslDir
}
compileQuerydsl {
    options.annotationProcessorPath = configurations.querydsl
}
configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
    querydsl.extendsFrom compileClasspath
}

// 서버 모듈 emosaac2에 있는 파일을 현재 스프링부트 파일로 복사하는 설정
task copyPrivate(type: Copy) {
    copy {
        //서브 모듈 emosaac2에 있는 application.yml을 현재 파일의 지정된 위치로 복사
        from '../emosaac2'
        include "application.yml"
        into 'src/main/resources'
    }
    copy {
        //서브 모듈 emosaac2에 application.properties를 현재 파일의 지정된 위치로 복사
        from '../emosaac2'
        from '../emosaac2'
        include "application.properties"
        into 'src/main/resources'
    }
    copy {
        //서브 모듈 emosaac2에 구글 클라우드 버전의 시크릿키를 가진
        // woven-name-382111-f4f2ed422bd9.json 파일을 현재 파일의 지정된 위치로 복사
        from '../emosaac2'
        from '../emosaac2'
        include "woven-name-382111-f4f2ed422bd9.json"
        into 'src/main/resources'
    }
}
}
}

```

인텔리제이 - Dockerfile

```
FROM openjdk:11
VOLUME /tmp
EXPOSE 8081
ARG JAR_FILE=build/libs/*.jar
```

장고 - emosaac/settings.py

```
from pathlib import Path
from emosaac import my_settings
import os

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = my_settings.SECRET_KEY

DEBUG = True

ALLOWED_HOSTS = [
    "j8d203.p.ssafy.io",
    "127.0.0.1"
]

CORS_ORIGIN_WHITELIST = [
    "https://j8d203.p.ssafy.io",
]

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'userbasedcf',
    'userbasedpredict',
    'django_crontab', # 크론 설정
    'itembasedcf',
    'corsheaders',
]

CRONJOBS = [
    ('0 0 * * *', 'emosaac.cron.crontab_job_cf', '>>' + os.path.join(BASE_DIR, 'config/log/cron.log') + ' 2>&1 '),
    # 매일 정각
    ('0 0 * * *', 'emosaac.cron.crontab_job_age_gen', '>>' + os.path.join(BASE_DIR, 'config/log/cron.log') + ' 2>&1 '),
    ('* * * * *', 'emosaac.cron.crontab_job_cf', '>>' + os.path.join(BASE_DIR, 'config/log/cron.log') + ' 2>&1 '),
    # 매 분마다 실행
]

# CRONTAB_DJANGO_SETTINGS_MODULE = 'emosaac.settings.local_settings'
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'corsheaders.middleware.CorsMiddleware',
]

ROOT_URLCONF = 'emosaac.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
```

```

        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
]

WSGI_APPLICATION = 'emosaac.wsgi.application'

DATABASES = my_settings.DATABASES

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia/Seoul'

USE_I18N = True

USE_TZ = False # False 로 설정해야 DB에 변경 된 TIME_ZONE 이 반영 됨

STATIC_URL = 'static/'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

장고 - emosaac/my_settings.py

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'emosaac',
        'USER': 'emosaac',
        'PASSWORD': '',
        'HOST': 'j8d203.p.ssafy.io',
        'PORT': '3306',
    }
}

SECRET_KEY = ''

```

장고 - Dockerfile

```

# ./Dockerfile
FROM python:3
# ENV PYTHONUNBUFFERED 1
# # 가상환경 생성
# RUN python -m venv /opt/venv
# RUN . /opt/venv/bin/activate

# ENV PATH="/opt/venv/bin:$PATH"

# RUN source /opt/venv/bin/activate
RUN mkdir /usr/src/app

# cron 설정
RUN apt-get update
RUN apt-get -y install cron
RUN apt-get -y install vim

```



```

WORKDIR /usr/src/app

RUN touch /var/log/cron.log

# 컨테이너 작업경로(requirements.txt, manage.py 위치)
RUN rm -rf /etc/localtime
RUN ln -s /usr/share/zoneinfo/Asia/Seoul /etc/localtime

COPY . .

## Install packages
RUN apt-get install -y python3-dev python3-pip
RUN pip3 install Django
RUN pip install --upgrade pip
RUN pip install django-crontab
RUN pip install django-cors-headers
COPY requirements.txt ./
RUN pip install -r requirements.txt

# cron 설정
RUN service cron start && service cron status
RUN #service cron status
RUN python manage.py crontab add

## Run the application on the port 8080
EXPOSE 8080
CMD ["python", "manage.py", "runserver", "0.0.0.0:8080"]

```

젠킨스 파이프라인

스프링 서버

```

pipeline {
    agent any

    stages{
        stage('Prepare') {
            steps {
                checkout scmGit(
                    branches: [[name: '*be/develop']],
                    extensions: [submodule(parentCredentials: true, recursiveSubmodules: true, reference: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22D2', userRemoteConfigs: [[credentialsId: 'gms9424_sub', url: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22D2']]]
                )
            }
            post {
                success {
                    echo 'Successfully Cloned Repository'
                }
                failure {
                    error 'This pipeline stops here...'
                }
            }
        }

        stage('build'){
            steps{
                dir('server'){
                    sh '''
                        chmod +x gradlew
                        echo 'start bootJar'
                        ./gradlew clean bootJar
                    '''
                }
            }
        }

        stage('dockerizing'){
            steps{
                dir('server'){

                    //////////////////////////////////////////////////젠킨스 실패하면 애네를 주석처리해보자////////////////////////////////////
                    sh 'docker stop test2'
                    sh 'docker rm test2'
                    sh 'docker rmi -f $(docker images -f "dangling=true" -q)'
                    //////////////////////////////////////
                }
            }
        }
    }
}

```

```

        sh 'docker build -t o6e1/emosaac-server:0.0.1 .'
    }
}

stage('Deploy') {
    steps {
        sh 'docker run -d -p 8081:8081 --name test2 o6e1/emosaac-server:0.0.1 .'
    }

    post {
        success {
            echo 'success'
        }

        failure {
            echo 'failed'
        }
    }
}
}
}

```

장고 서버

```

pipeline {
    agent any

    stages{
        stage('Prepare') {
            steps {
                checkout scmGit(
                    branches: [[name: '*/data/develop']],
                    userRemoteConfigs: [[credentialsId: 'gms9424_sub', url: 'https://lab.ssafy.com/s00-bigdata-recom-sub2/S08P22D2
                ]
            }
            post {
                success {
                    echo 'Successfully Cloned Repository'
                }
                failure {
                    error 'This pipeline stops here...'
                }
            }
        }
    }

    stage('dockerizing'){
        steps{
            dir('server_django'){

                ///////////////젠킨스 실패하면 애네를 주석처리해보자/////////////////
                sh 'docker stop django'
                sh 'docker rm django'
                ///////////////////////////////////////////////////

                sh 'docker build -t o6e1/django-server:0.0.1 .'
            }
        }
    }

    stage('Deploy') {
        steps {
            sh 'ls /usr/src -al'
            sh 'docker run -d -p 8000:8000 --name django o6e1/django-server:0.0.1'
        }

        post {
            success {
                echo 'success'
            }

            failure {
                echo 'failed'
            }
        }
    }
}

```

```
}  
}
```

EC2 내부 Nginx 설정

/etc/nginx/conf.d/emosaac.conf

```
server {  
    #listen [::]:443 ssl ipv6only=on; # managed by Certbot  
    listen 443 ssl; # managed by Certbot  
  
    server_name j8d203.p.ssafy.io; # managed by Certbot  
  
    ## SSL 인증서 적용  
    ssl_certificate /etc/letsencrypt/live/j8d203.p.ssafy.io/fullchain.pem; # managed by Certbot  
    ssl_certificate_key /etc/letsencrypt/live/j8d203.p.ssafy.io/privkey.pem; # managed by Certbot  
  
    location / {  
        proxy_pass http://localhost:3000;  
    }  
  
    location /oauth2/redirect {  
        proxy_pass http://localhost:3000;  
    }  
  
    location /api { # location 이후 특정 url을 처리하는 방법을 정의  
        proxy_pass http://localhost:8081/api; # Request에 대해 어디로 리다이렉트하는지  
        proxy_redirect off;  
        charset utf-8;  
  
        proxy_http_version 1.1;  
        proxy_set_header Connection "upgrade";  
        proxy_set_header Upgrade $http_upgrade;  
        proxy_set_header Host $http_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-NginX-Proxy true;  
    }  
  
    location /oauth2/callback{  
        proxy_pass http://localhost:8081;  
    }  
}
```

/etc/nginx/nginx.conf

```
http {  
    client_max_body_size 5M; // 추가하기  
  
    ...  
}
```

Next.js 배포 설정

AWS에 gitlab runner 설치

```
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.rpm.sh" | sudo bash  
  
sudo yum install gitlab-runner  
  
sudo gitlab-runner register  
  
// 후작업  
// 1. 깃랩 주소 입력  
// 2. 깃랩 ci/cd 토큰 값 입력
```

```
// 3. runner 설명 작성
// 4. runner tag명 등록: 이후 .gitlab-ci.yml에서 러너 선택에 활용됨
// 5. 부가 노트 등록
// 6. 실행 형식 선택 : shell
// 7. 등록 끝

//runner 실행
gitlab-runner start
gitlab-runner run
```

gitlab CI/CD pipeline - .gitlab-ci.yml 작성

```
test-job: # 임의로 지정한 JOB 이름
  stage: deploy # build, test, deploy 3단계 스테이지 중 하나
  only:
    - fe/develop # develop 브랜치 이벤트 발생시 실행
  script:
    - git pull origin fe/develop

    - sudo chown -R `whoami` ~/.npm
    - sudo chown -R `whoami` /usr/local/lib/node_modules
    - cd client
    - cp .env.production .env

    - npm i
    - npm install universal-cookie
    - npm install cookie
    - sudo npm run build

    - sudo chown -R gitlab-runner:gitlab-runner .next
    - pm2 list

    # - pm2 start npm --name "emosaac" -w -i max -- start // 초기 한번만 실행
    - pm2 restart emosaac

  tags:
    - nextjs # build-server 태그가 달린 runner에서 실행
```

pm2 설치

```
// pm2 글로벌 설치
npm install pm2 -g
```

외부 서비스 문서


소셜 로그인

Naver

- Oauth 기반 소셜 로그인 API 제공
 - redirect URI : <http://j8d203.p.ssafy.io:8081/oauth2/callback/google>

네이버 로그인 개발가이드 - LOGIN

네이버 로그인 개발가이드 1. 개요 2. 네이버 로그인 서비스 소개 2.1 네이버 로그인 서비스에 대하여 2.2 제공하는 기능 2.2.1 소셜 로그인 2.2.2 로그인 연동 회원 프로필 조회 2.2.3 네이버의 로그인 오픈API 이용 2.2.4 서비스 이용 통계

 <https://developers.naver.com/docs/login/devguide/devguide.md#2-2-3-네이버의-로그인-오픈api-이용>

Kakao

- Oauth 기반 소셜 로그인 API 제공
 - redirect URI : <http://j8d203.p.ssafy.io:8081/oauth2/callback/kakao>

Kakao Developers

이 문서는 REST API를 사용한 로그인 구현 방법을 안내합니다. 이 문서에 포함된 기능 일부는 [도구] > [REST API 테스트]를 통해 사용해 볼 수 있습니다. 카카오 로그인 구현에 필요한 로그인 버튼 이미지는 [도구] > [리소스 다운로드]에서 제공합니다. 해당 로그인 버튼은 디자인 가이드를 참고하여 서비스 UI에 적합한 크기로 수정하여 사용할 수 있습니다. <https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api>

kakao developers

S3 설정

IAM 사용자 생성

IAM > 사용자 > 사용자 생성

1단계
사용자 세부 정보 지정

2단계
권한 설정

3단계
검토 및 생성

사용자 세부 정보 지정

사용자 세부 정보

사용자 이름

emosaad

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

☐ AWS Management Console에 대한 사용자 액세스 권한 제공 - 선택 사항
사용자에게 콘솔 액세스 권한을 제공하는 경우 IAM Identity Center에서 액세스를 관리하는 것은 모범 사례입니다.

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

• 권한 옵션 : 직접 정책 연결

IAM > 사용자 > 사용자 생성

1단계
사용자 세부 정보 지정

2단계
권한 설정

3단계
검토 및 생성

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 (1038)
새 사용자에게 연결할 정책을 하나 이상 선택합니다.

검색 텍스트 또는 값을 기준으로 배포 필터링

< 1 2 3 4 5 6 7 ... 52 > ⚙

☐ 정책 이름

▲ | 유형

▼ | 연결된 엔터티

▼

☐ AccessAnalyzerServiceRolePolicy

AWS 관리형

0

• 권한 정책 :

- AmazonS3FullAccess 선택
- AWSLambda_fullAccess 선택

포팅 매뉴얼

13

권한 정책 (1/1038)

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

🔄

정책 생성

🔍

텍스트 또는 값을 기준으로 배포 필터링

11 개 일치

< 1 >

s3

✕

필터 지우기

	정책 이름	유형	연결된 엔터티
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	AWS 관리형	0
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS 관리형	2
<input type="checkbox"/>	AmazonS3ObjectLambdaExecution...	AWS 관리형	0
<input type="checkbox"/>	AmazonS3OutpostsFullAccess	AWS 관리형	0
<input type="checkbox"/>	AmazonS3OutpostsReadOnlyAccess	AWS 관리형	0
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	AWS 관리형	0
<input type="checkbox"/>	AWSBackupServiceRolePolicyForS3...	AWS 관리형	0
<input type="checkbox"/>	AWSBackupServiceRolePolicyForS3...	AWS 관리형	0
<input type="checkbox"/>	IVSRecordToS3	AWS 관리형	0
<input type="checkbox"/>	QuickSightAccessForS3StorageMan...	AWS 관리형	0

권한 정책 (2/1038)

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

🔄

정책 생성

🔍

텍스트 또는 값을 기준으로 배포 필터링

13 개 일치

< 1 > ⚙️

awslambda

✕

필터 지우기

	정책 이름	유형	연결된 엔터티
<input checked="" type="checkbox"/>	AWSLambda_FullAccess	AWS 관리형	1
<input type="checkbox"/>	AWSLambda_ReadOnlyAccess	AWS 관리형	0
<input type="checkbox"/>	AWSLambdaBasicExecutionRole	AWS 관리형	0

액세스 키 생성

• 사용자 선택 후 액세스 키 생성(엑세스키 만들기)

엑세스 키 (0)

엑세스 키를 사용하여 AWS CLI, AWS Tools for PowerShell, AWS SDK 또는 직접 AWS API 호출을 통해 AWS에 프로그래밍 방식 호출을 전송합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다. [Learn more](#)

엑세스 키 만들기

엑세스 키 없음

엑세스 키와 같은 장기 자격 증명을 사용하지 않는 것이 모범 사례입니다. 대신 단기 자격 증명을 제공하는 도구를 사용하세요. [Learn more](#)


엑세스 키 만들기

1단계
액세스 키 모범 사례 및 대안2단계 - 선택 사항
설명 태그 설정3단계
액세스 키 검색

액세스 키 모범 사례 및 대안

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

- ☐ Command Line Interface(CLI)
AWS CLI를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ 로컬 코드
로컬 개발 환경의 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☒ AWS 컴퓨팅 서비스에서 실행되는 애플리케이션
Amazon EC2, Amazon ECS 또는 AWS Lambda와 같은 AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ 서버 파티 서비스
AWS 리소스를 모니터링 또는 관리하는 서버 파티 애플리케이션 또는 서비스에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ AWS 외부에서 실행되는 애플리케이션
애플리케이션을 온프레미스 호스트에서 실행하거나 로컬 AWS 클라이언트 또는 서버 파티 AWS 플러그 인을 사용할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ 기타
귀하의 사용 사례가 여기에 나열되어 있지 않습니다.

 **권장되는 대안**
EC2 인스턴스 또는 Lambda 함수와 같은 컴퓨팅 리소스에 IAM 역할을 할당하여 액세스를 위한 임시 보안 인증을 자동으로 제공합니다. [Learn more](#)

☐ 위의 권장 사항을 이해했으며 액세스 키 생성을 계속하려고 합니다.

취소 다음

3. 버킷 생성

- 버킷 이름은 소문자만 가능
- 지역은 서울로 설정

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.

버킷 선택

객체 소유권 Info

다른 AWS 계정에서 이 버킷에 작성한 객체의 소유권 및 액세스 제어 목록(ACL)의 사용을 제어합니다. 객체 소유권은 객체에 대한 액세스를 지정할 수 있는 사용자를 결정합니다.

☐ ACL 비활성화됨(권장)

이 버킷의 모든 객체는 이 계정이 소유합니다. 이 버킷과 그 객체에 대한 액세스는 정책을 통해서만 지정됩니다.

☒ ACL 활성화됨

이 버킷의 객체는 다른 AWS 계정에서 소유할 수 있습니다. 이 버킷 및 객체에 대한 액세스는 ACL을 사용하여 지정할 수 있습니다.

객체 소유권

☐ 버킷 소유자 선호

이 버킷에 작성된 새 객체가 bucket-owner-full-control 삽입 ACL을 지정하는 경우 새 객체는 버킷 소유자가 소유합니다. 그렇지 않은 경우 객체 라이터가 소유합니다.

☒ 객체 라이터

객체 라이터는 객체 소유자로 유지됩니다.



ACL 활성화 관련 향후 권한 변경 사항

2023년 4월부터는 S3 콘솔을 사용하여 버킷을 생성할 때 ACL을 활성화하기 위해 s3:PutBucketOwnershipControls 권한이 있어야 합니다. [자세히 알아보기](#)

• 퍼블릭으로 설정

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(액세스 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 지정에만 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 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.



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.

- **정책 설정**

Amazon S3 > 버킷 > emosaacbucket > 버킷 정책 편집

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](#)

버킷 ARN
arn:aws:s3:::emosaacbucket

정책

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "AddPerm",
6       "Effect": "Allow",
7       "Principal": "*",
8       "Action": "s3:*",
9       "Resource": [
10        "arn:aws:s3:::emosaacbucket/*",
11        "arn:aws:s3:::emosaacbucket"
12      ]
13    }
14  ]
15 }
```

```

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

- **cors 설정**

CORS(cross-origin 리소스 공유) 편집 [Info](#)**CORS(Cross-origin 리소스 공유)**

JSON으로 작성된 CORS 구성은 한 도메인에 로드되어 다른 도메인의 리소스와 상호 작용하는 클라이언트 웹 애플리케이션에 대한 방법을 정의합니다. [자세히 알아보기](#)

```

1  [
2  {
3    "AllowedHeaders": [
4      "*"
5    ],
6    "AllowedMethods": [
7      "GET",
8      "PUT",
9      "POST",
10     "HEAD"
11   ],
12   "AllowedOrigins": [
13     "*"
14   ],
15   "ExposeHeaders": [
16     "x-amz-server-side-encryption",
17     "x-amz-request-id",
18     "x-amz-id-2"
19   ],
20   "MaxAgeSeconds": 3000
21 }
22 ]

```

취소

변경 사항 저장

```

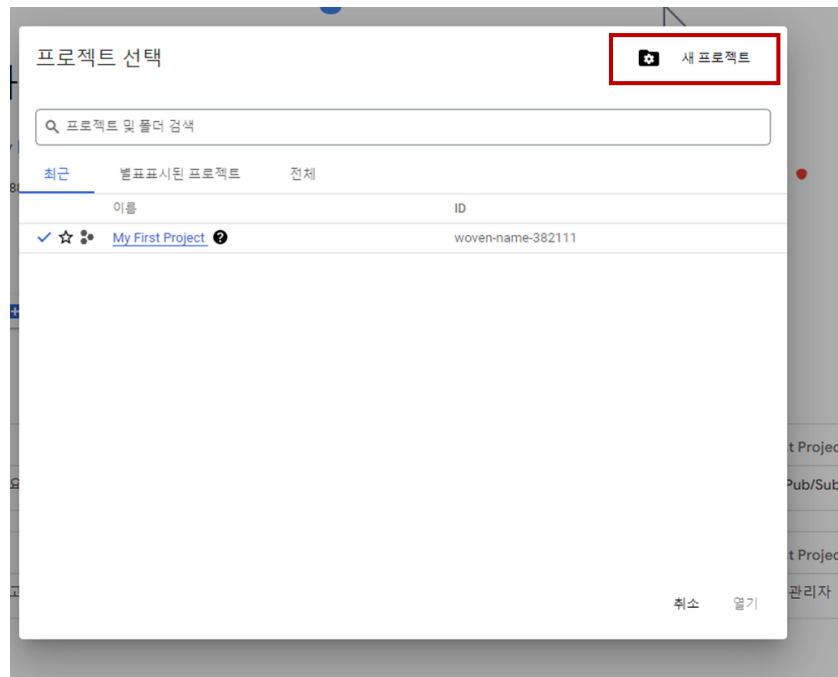
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "PUT",
      "POST",
      "HEAD"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "ExposeHeaders": [
      "x-amz-server-side-encryption",
      "x-amz-request-id",
      "x-amz-id-2"
    ],
    "MaxAgeSeconds": 3000
  }
]

```

google cloud vision api

1. google cloud platform 등록하기

- 새 프로젝트



- 새 프로젝트 이름 기입

Google Cloud

새 프로젝트

projects 할당량이 23개 남았습니다. 할당량 증가를 요청하거나 프로젝트를 삭제하세요. [자세히 알아보기](#)

[MANAGE QUOTAS](#)

프로젝트 이름 *

emosaacTest

프로젝트 ID: emosaactest입니다. 나중에 변경할 수 없습니다. [수정](#)

위치 *

조직 없음

[찾아보기](#)

상위 조직 또는 폴더

만들기

취소

2. 프로젝트 결제 등록

무료 평가판 상태: 크레딧은 ₩391,397.00, 무료 평가판 기간은 84일 남았습니다. 완전한 계정을 사용하면 Google Cloud Platform의 모든 기능에 무제한 액세스할 수 있습니다.

Google Cloud emosaacTest 리소스, 문서, 제품 등 검색(/)

Cloud 개요 > 모든 제품 보기

고정됨

API API 및 서비스 >

결제 >

IAM 및 관리자 >

Marketplace

Compute Engine >

Kubernetes Engine >

Cloud Storage >

BigQuery >

VPC 네트워크 >

Cloud Run >

SQL >

보안 >

Google Maps Plat... >

제품 더보기 >

대시보드 활동 권장사항

프로젝트 정보

프로젝트 이름
emosaacTest

프로젝트 번호
8665441995

프로젝트 ID
emosaacTest

[이 프로젝트에 사용자 추가](#)

[프로젝트 설정으로 이동](#)

리소스

BigQuery
데이터 웨어하우스/분석

SQL
관리형 MySQL, PostgreSQL, SQL Server

Compute Engine
VM, GPU, TPU, 디스크

Storage
멀티클래스 멀티 리전 객체 스토리지

Cloud 함수
이벤트 기반 서버리스 함수

App Engine
관리 앱 플랫폼

API API

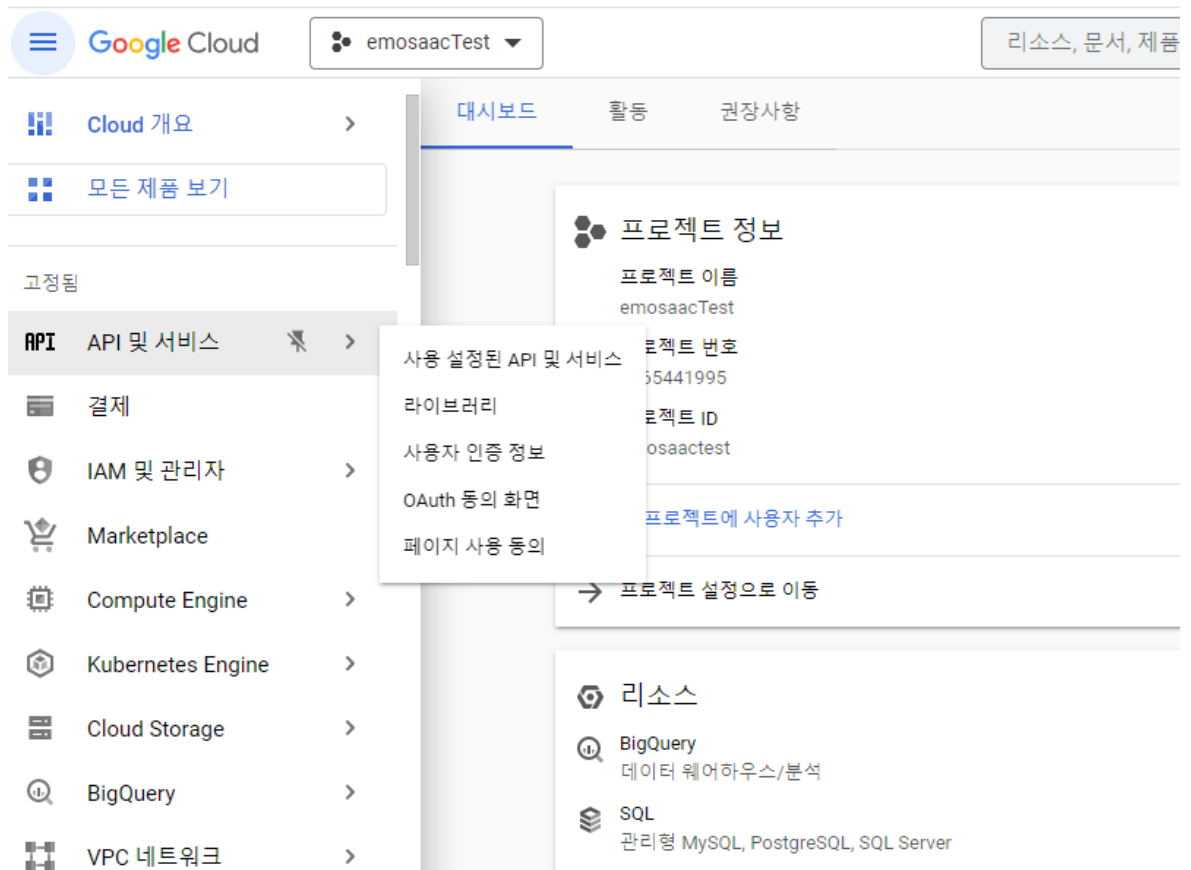
요청(요청/초)

No data is avail


9:30

[API 개요로 이동](#)


3. API 사용 설정하기



- google vision api 등록

 emosaacTest

[←](#) 제품 세부정보



Cloud Vision API

[Google Enterprise API](#)

Image Content Analysis

[사용](#) [API 사용해 보기](#)

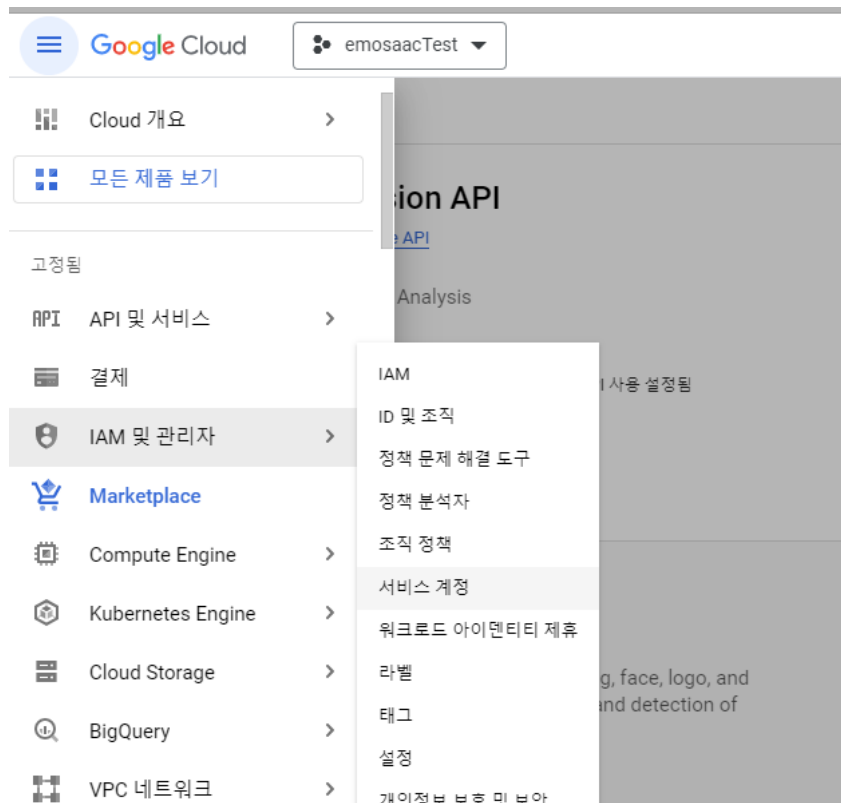
[개요](#) [가격 책정](#) [문서](#) [지원](#)

개요

Integrates Google Vision features, including image labeling, face, logo, and landmark detection, optical character recognition (OCR), and detection of explicit content, into applications.

4. 서비스 계정 등록하기

- IAM 및 관리자 > 서비스 계정



• 서비스 계정 만들기

IAM 및 관리자

서비스 계정

+ 서비스 계정 만들기

삭제

액세스 관리

새로고침

IAM
ID 및 조직
정책 문제 해결 도구
정책 분석자
조직 정책
서비스 계정
워크로드 아이덴티티 제휴
라벨
태그
설정
개인정보 보호 및 보안

'emosaacTest' 프로젝트의 서비스 계정

서비스 계정은 Compute Engine VM에서 실행되는 코드, App Engine 앱 또는 Google 외부에서 실행되는 시스템과 조직 정책은 서비스 계정을 보호하고 자동 IAM 부여, 키 생성/업로드, 다른 서비스 계정의 생성과 같은 위험한 서

필터

속성 이름 또는 값 입력

<input type="checkbox"/>	이메일	상태	이름 ↑	설명	키 ID	키 생성일	OAuth
표시할 행이 없습니다.							

← 서비스 계정 만들기

1 서비스 계정 세부정보

서비스 계정 이름

emosaac

이 서비스 계정의 표시 이름입니다.

서비스 계정 ID *

emosaac-803

X ↺

이메일 주소: emosaac-803@emosaactest.iam.gserviceaccount.com

서비스 계정 설명

이 서비스 계정에서 수행할 작업을 설명하세요.

만들고 계속하기

2 이 서비스 계정에 프로젝트에 대한 액세스 권한 부여 (선택사항)

|

3 사용자에게 이 서비스 계정에 대한 액세스 권한 부여 (선택사항)

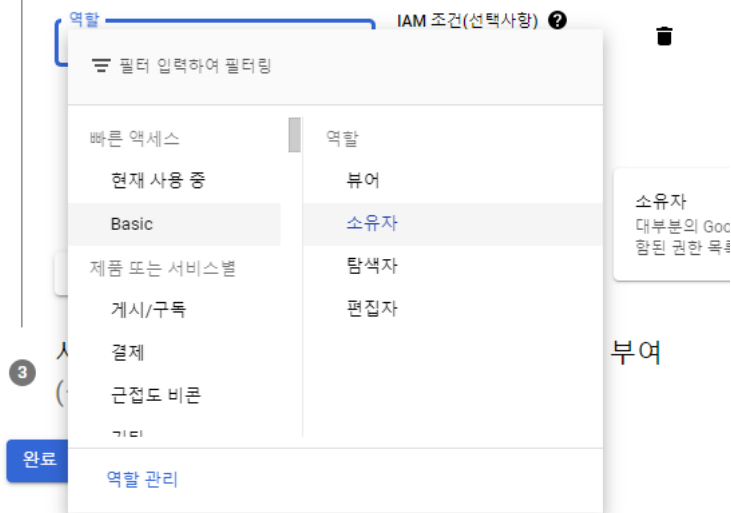
완료

취소

- 권한은 전체 권한을 가진 소유자로 설정

2 이 서비스 계정에 프로젝트에 대한 액세스 권한 부여 (선택사항)

프로젝트의 리소스에서 특정 작업을 완료할 수 있도록 이 서비스 계정에 emosaacTest에 대한 액세스 권한을 부여합니다. [자세히 알아보기](#)



• 생성 확인

'emosaacTest' 프로젝트의 서비스 계정

서비스 계정은 Compute Engine VM에서 실행되는 코드, App Engine 앱 또는 Google 외부에서 실행되는 시스템과 같은 Google Cloud 서비스 ID를 나타냅니다. [서비스 계정 자세히 알아보기](#)

조직 정책은 서비스 계정을 보호하고 자동 IAM 부여, 키 생성/업로드, 다른 서비스 계정의 생성과 같은 위험한 서비스 계정 기능을 차단하는 데 사용될 수 있습니다. [서비스 계정 조직 정책 자세히 알아보기](#)

필터 속성 이름 또는 값 입력								
<input type="checkbox"/>	이메일	상태	이름 ↑	설명	키 ID	키 생성일	OAuth 2 클라이언트 ID ⓘ	작업
<input checked="" type="checkbox"/>	연 emosaac-803@emosaac-test.iam.gserviceaccount.com	✓	emosaac		키 없음		112337130761955935897	⋮

5. 서비스 계정 비공개 키 받기

• 키 관리 클릭

'emosaacTest' 프로젝트의 서비스 계정

서비스 계정은 Compute Engine VM에서 실행되는 코드, App Engine 앱 또는 Google 외부에서 실행되는 시스템과 같은 Google Cloud 서비스 ID를 나타냅니다. [서비스 계정 자세히 알아보기](#)

조직 정책은 서비스 계정을 보호하고 자동 IAM 부여, 키 생성/업로드, 다른 서비스 계정의 생성과 같은 위험한 서비스 계정 기능을 차단하는 데 사용될 수 있습니다. [서비스 계정 조직 정책 자세히 알아보기](#)

이름	상태	이름 ↑	설명	키 ID	키 생성일	OAuth 2 클라이언트 ID	작업
emosaac-803@emosaactest.iam.gserviceaccount.com	✓	emosaac		키 없음		112337130761955935897	세부정보 관리 권한 관리 키 관리 측정항목 보기 로그 보기 사용 중지 삭제

- 키 추가 > 새 키 만들기

세부정보 권한 **키** 측정항목 로그

키

⚠ 보안 침해 시 서비스 계정 키로 인해 보안 위험이 발생할 수 있습니다. 서비스 계정 키를 다운로드하는 대신 [워크로드 아이덴티티 제휴](#)를 사용하십시오.

새 키 쌍을 추가하거나 기존 키 쌍의 공개키 인증서를 업로드하세요.

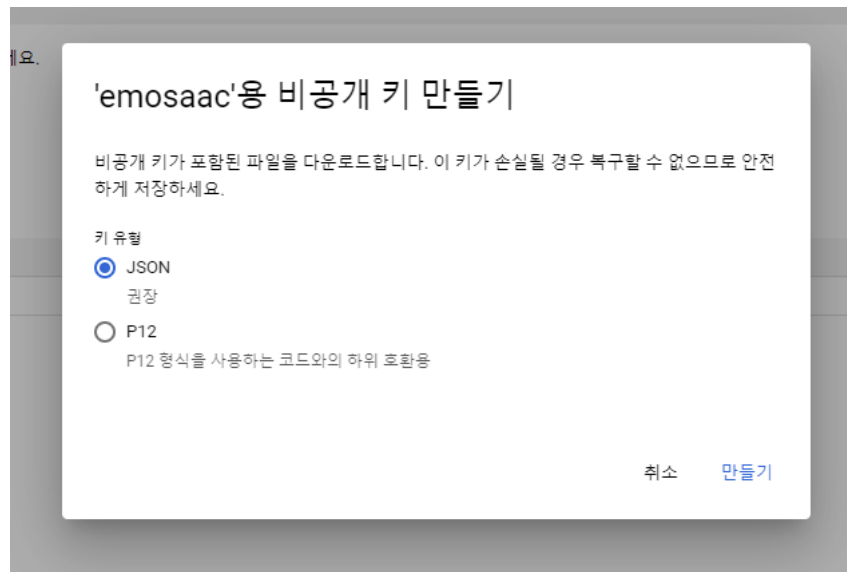
[조직 정책](#)을 사용하여 서비스 계정 키 생성을 차단합니다.
[서비스 계정의 조직 정책 설정 자세히 알아보기](#)

키 추가 ▾

새 키 만들기
 기존 키 업로드

키	키 생성일	키 만료일

- json 유형 선택 > 만들기



- 발급 성공

