

포팅매뉴얼

≡ 태그

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

II. 외부 서비스

1. Dockerhub
2. AWS S3
3. Naver Cloud Clova OCR
4. OpenAI ChatGPT

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. Naver Cloud Clova OCR
4. OpenAI GPT

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: 18.16.1
7. MySQL: 8.0.34
8. Jenkins: 2.416
9. Cloud Storage: AWS S3
10. OCR API: Naver Cloud Clova OCR

11. Python: 3.11.5

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/namani

b. application.properties : ./backend/namani/src/main/resources

c. application-prod.properties : ./backend/namani/src/main/resources

d. application-local.properties : ./backend/namani/src/main/resources

3. Fast API

a. Dockerfile : ./ML

4. Docker

a. docker-compose.yml : (Docker의 Jenkins 컨테이너 내) /var/jenkins_home

b. .env : (EC2 인스턴스 내) /var/jenkins_home/workspace/Namani/config

5. Nginx

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

3. 설정 파일 및 환경 변수 정보

1. React

a. Dockerfile :

```
FROM node:18.16.1
# http -> https 시 웹소켓 오류 때문에
ENV WDS_SOCKET_PORT 0
WORKDIR /app
COPY package.json .
RUN npm install
# 현재 디렉토리의 모든 파일을 도커 컨테이너 워킹 디렉토리에 저장
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

2. Spring Boot

a. Dockerfile :

```
FROM adoptopenjdk/openjdk11
EXPOSE 8081
ARG JAR_FILE=build/libs/namani-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=8081
spring.profiles.default=prod
spring.jackson.time-zone=Asia/Seoul
spring.jackson.serialization.fail-on-empty-beans=false
#jwt
jwt.secret=secretKey=${JWT_SECRET_KEY}
#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=10MB
spring.servlet.multipart.max-request-size=10MB

# ClovaOCR Keys
clova.secret=${CLOVA_SECRET}
clova.api.url=${CLOVA_API_URL}

```

c. application-prod.properties :

```

spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.url=${LOCAL_DATASOURCE_URL}
spring.datasource.username=${LOCAL_DATASOURCE_USERNAME}
spring.datasource.password=${LOCAL_DATASOURCE_PASSWORD}
spring.jpa.hibernate.ddl-auto=update
#jpa query log
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.highlight_sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect
#logger setting
logging.level.root=info
logging.level.com.ssafy.namani=debug

```

d. application-local.properties :

```

spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.url=${PROD_DATASOURCE_URL}
spring.datasource.username=${PROD_DATASOURCE_USERNAME}
spring.datasource.password=${PROD_DATASOURCE_PASSWORD}
spring.jpa.hibernate.ddl-auto=update
#jpa query log
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.highlight_sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect
#logger setting
logging.level.root=info
logging.level.com.ssafy.namani=debug

```

3. Fast API

a. Dockerfile :

```

FROM python:3.11
WORKDIR /fast

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

COPY . /fast
EXPOSE 8000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]

```

4. Docker

a. docker-compose.yml :

```

version: '3.8'
services:
  backend:
    image: ${dockerhub_username}/backend:latest
    build:
      context: ./workspace/Namani/backend/namani

```

```

dockerfile: Dockerfile

frontend:
  image: ${dockerhub_username}/frontend:latest
  environment:
    - WDS_SOCKET_PORT=0
  build:
    context: ./workspace/Namani/frontend
    dockerfile: Dockerfile

ml:
  image: ${dockerhub_username}/ml:latest
  build:
    context: ./workspace/Namani/ML
    dockerfile: Dockerfile
  ports:
    - 8000:8000
  volumes:
    - ./:fast

```

b. .env :

```

PROD_DATASOURCE_URL=${your mysql or mariadb url}
PROD_DATASOURCE_USERNAME=${your db username}
PROD_DATASOURCE_PASSWORD=${your db password}
WDS_SOCKET_PORT=0
JWT_SECRET_KEY=${your jwt secret key}
S3_ACCESS_KEY=${your s3 access key}
S3_SECRET_KEY=${your s3 secret key}
CLOVA_API_URL=${your naver cloud clova api url}
CLOVA_SECRET=${your naver cloud clova secret key}
BUCKET_NAME=${your s3 bucket name}
apikey=${your open-ai api key}

```

5. 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 /ml {
        proxy_pass http://localhost:8000;
    }

}

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

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

```

6. Jenkins Pipeline Script

```

pipeline {
    agent any
    tools{
        gradle 'gradle'
        nodejs 'NodeJS 18.16.1'
    }
    environment {
        DOCKER_REPO_BACK = "${dockerhub username}/backend"
        DOCKER_REPO_FRONT = "${dockerhub username}/frontend"
        DOCKER_REPO_ML = "${dockerhub username}/ml"
        DOCKER_CREDENTIAL = credentials('dockerhub-credential')
        BACK_CONTAINER = "pendi-back"
        FRONT_CONTAINER = "pendi-front"
        ML_CONTAINER = "pendi-ml"
        SSH_CONN = "ubuntu@j9a410.p.ssafy.io"
        ENV_DIR = "./config/.env"
        BACK_PORT = "8081"
        FRONT_PORT = "3000"
        ML_PORT = "8000"
    }
    stages{
        stage('Git Clone') {
            steps {
                sh "#### Git Clone Start ####"
                git branch: 'master', credentialsId: 'gitlabID', url: 'https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22A410.g
                sh "#### Git Clone Success ####"

            }
        }
        stage('Build Backend'){
            steps{
                dir('backend/namani'){
                    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 serve"
                    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/backend:latest"
  sh "docker push pohangman/frontend:latest"
  sh "docker push pohangman/ml: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 pend-backend'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker rmi -f $DOCKER_REPO_BACKEND'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker pull $DOCKER_REPO_BACKEND'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'echo y | docker image prune'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker images'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker run -d --name pend-backend --env-file $ENV_DIR -p $BACK_PORT'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker ps'"
    }
    sh "echo '#### Deploy Backend Server Success ####'"
  }
}

stage('Deploy Frontend Server'){
  steps{
    sh "echo '#### Deploy Frontend Server Start ####'"
    sshagent(credentials:['ec2-user']){
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker rm -f pend-frontend'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker rmi -f $DOCKER_REPO_FRONT'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker pull $DOCKER_REPO_FRONT'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'echo y | docker image prune'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker images'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker run -d --name pend-frontend --env-file $ENV_DIR -p $FRONT_PORT'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker ps'"
    }
    sh "echo '#### Deploy Frontend Server Success ####'"
  }
}

stage('Deploy ML Server'){
  steps{
    sh "echo '#### Deploy ML Server Start ####'"
    sshagent(credentials:['ec2-user']){
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker rm -f pend-ml'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker rmi -f $DOCKER_REPO_ML'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker pull $DOCKER_REPO_ML'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'echo y | docker image prune'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker images'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker run -d --name pend-ml --env-file $ENV_DIR -p $ML_PORT'"
      sh "ssh -o StrictHostKeyChecking=no $SSH_CONN 'docker ps'"
    }
    sh "echo '#### Deploy ML Server Success ####'"
  }
}

}

}

}

```

4. 포트 개방

```
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 8080
sudo ufw allow 8080/tcp
sudo ufw allow 80
sudo ufw allow 20
```

```
sudo ufw allow 80/tcp
sudo ufw allow 22/tcp
sudo ufw allow 443/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

Project Hooks (1)

Push events SSL Verification: enabled

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

Test

Edit

Delete

- Test를 하고 나서 이와 같이 표시되면 연동이 성공 된 것이다

CI/CD > main > team-00 > Deploying service > Webhook settings

Hook executed successfully: HTTP 200

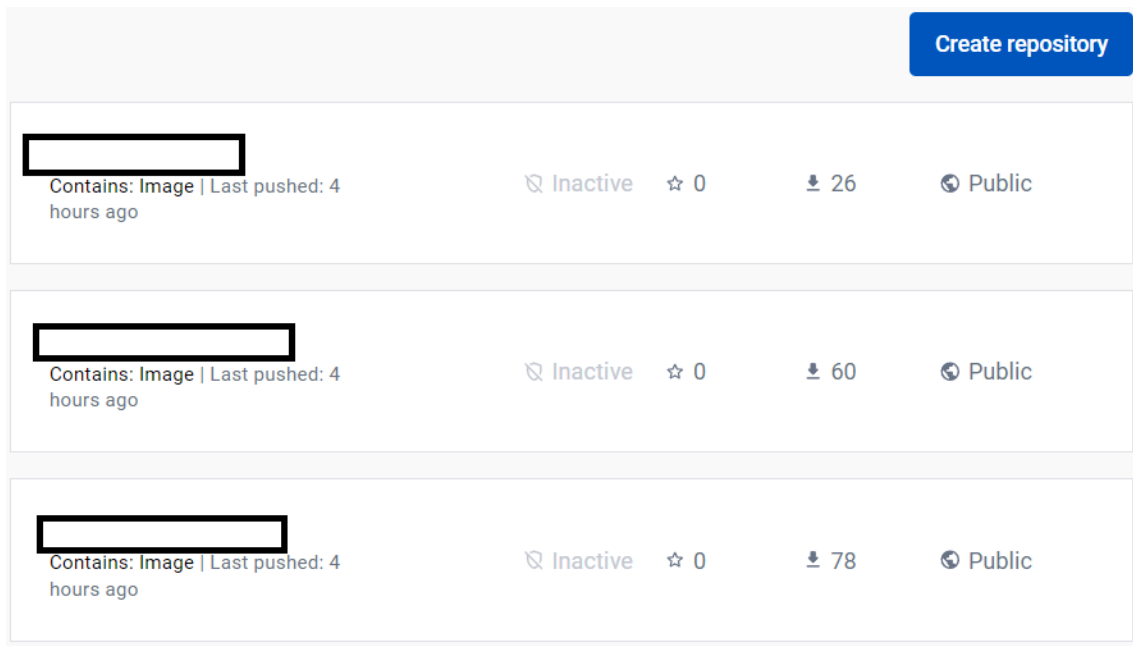
Search settings

II. 외부 서비스

1. Dockerhub

1. Dockerhub Repository 만들기

- 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

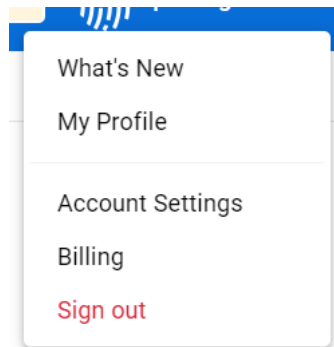
Only visible to you

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

The screenshot shows the Amazon S3 console interface. On the left is a navigation pane with options like '버킷', '액세스 지점', '다중 리전 액세스 지점', '배치 작업', 'S3용 IAM Access Analyzer', '이 계정의 퍼블릭 액세스 차단 설정', 'Storage Lens', '대시보드', 'AWS Organizations 설정', '기능 스포트라이트', and 'S3용 AWS Marketplace'. The main content area has a dark header with 'Amazon S3' and a sub-header '어디서든 원하는 양의 데이터를 저장 및 검색'. Below this is a section titled '작동 방식' with a video player showing 'Introduction to Amazon S3'. On the right, there's a '버킷 생성' (Create Bucket) section with a description: 'S3의 모든 객체는 버킷에 저장됩니다. S3에 파일과 폴더를 업로드하려면 객체가 저장될 버킷을 생성해야 합니다.' and a prominent orange '버킷 만들기' (Create Bucket) button. Below that is a '요금' (Pricing) section stating 'S3에서는 최소 요금이 없습니다. 사용한 만큼만 비용을 지불하며 요금은 S3 버킷의 위치에 따라 결정됩니다.' and links for 'AWS 월 사용량 계산기' and '요금 세부 정보 보기'. At the bottom right is a '리소스' (Resources) section.

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

your bucket name

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 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(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

your bucket name

객체

속성

권한

지표

관리

액세스 지점

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 \(Optional\)](#)

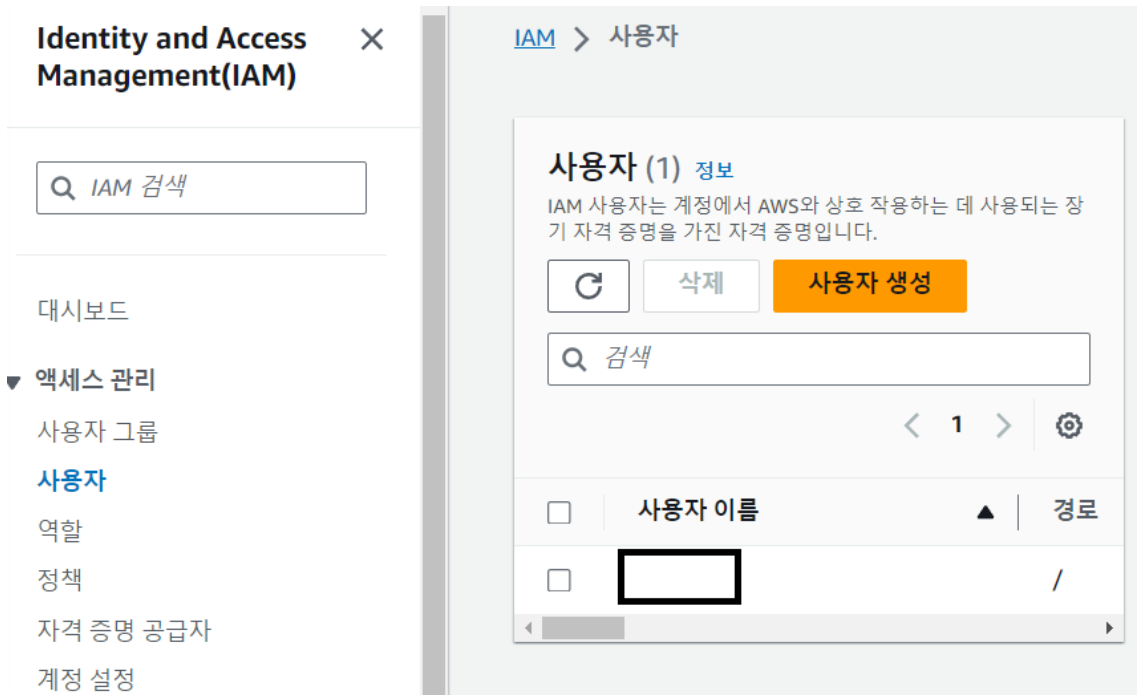
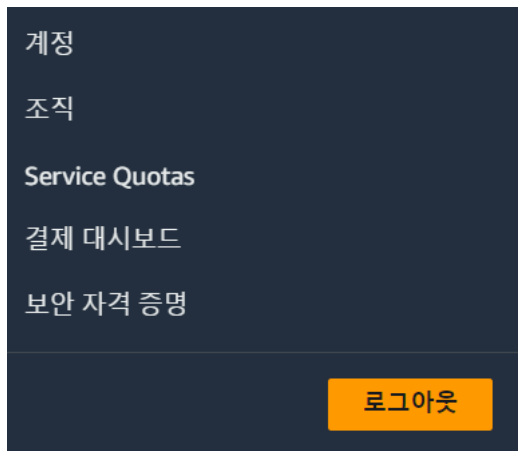
Add Statement

- 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

your_iam_name

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

× 모든 유형 ▼ 12 개 일치

< **1** > ⚙️

<input type="checkbox"/>	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

액세스 키 검색 정보

액세스 키

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

액세스 키

비밀 액세스 키

e. 발급받은 액세스 키와 비밀 액세스 키를 사용

3. Naver Cloud Clova OCR

1. Naver Cloud Platform Console 접속
2. 도메인 생성 → 고유의 도메인명과 도메인 코드 입력

생성

새로운 도메인을 생성합니다.

도메인명

test

중복 확인

도메인 코드

bunbun

중복 확인

지원 언어

☒ 한국어
 ☐ 일본어
 ☐ 중국어(번체)

서비스 타입

☐ 일반
 ☒ 템플릿

인식 모델

☒ Basic
 ☐ Premium

서비스 플랜

☒ Free
 ☐ Basic
 ☐ Standard
 ☐ Advanced

Free 서비스 플랜을 제외한 모든 서비스 플랜은 CLOVA OCR API를 호출하지 않아도 기본 요금이 부과됩니다.

CLOVA OCR은 도메인별 서비스 플랜에 따라 요금이 부과됩니다.
 Free 서비스 플랜을 제외한 모든 서비스 플랜은 CLOVA OCR API를 호출하지 않아도 기본 요금이 부과됩니다.
 서비스 플랜에 따라 기본으로 제공되는 API 호출 수가 다르며, 기본 제공 건수 초과 시 추가 요금이 부과됩니다. [서비스 플랜 요금 안내](#)를 참조해 주십시오.

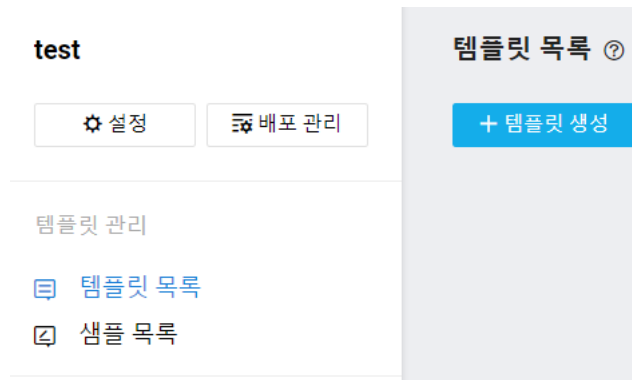
× 취소

✓ 생성

3. 생성 후, 콘솔에서 생성한 도메인의 템플릿 빌더 클릭

<input type="checkbox"/>	25336	test	bunbun g	한국어	Basic	Free	0	2023-10-04 16:59:22	2023-10-04 16:59:22	템플릿 빌더
--------------------------	-------	------	-------------	-----	-------	------	---	------------------------	------------------------	--------

4. 템플릿 목록 → 템플릿 생성



5. 템플릿 명을 입력 → 대표 이미지 등록 → 판독 필드 지정 → 저장

6. 설정 → API Gateway 연동 → Secret Key 복사 → API 게이트웨이 자동 연동 → API URL 복사

- 창을 닫더라도 다시 확인할 수 있음

7. 베타 배포 후 테스트를 거쳐 테스트가 제대로 된다면 서비스 배포 후 사용

배포 관리 ⓘ
템플릿 목록에서 배포할 템플릿의 체크 박스를 클릭한 후 [베타 배포] 버튼을 클릭해 주십시오. 베타 배포를 완료해야 테스트 및 서비스 배포를 할 수 있습니다.

템플릿 목록 **베타 배포** 템플릿 선택 All ▼

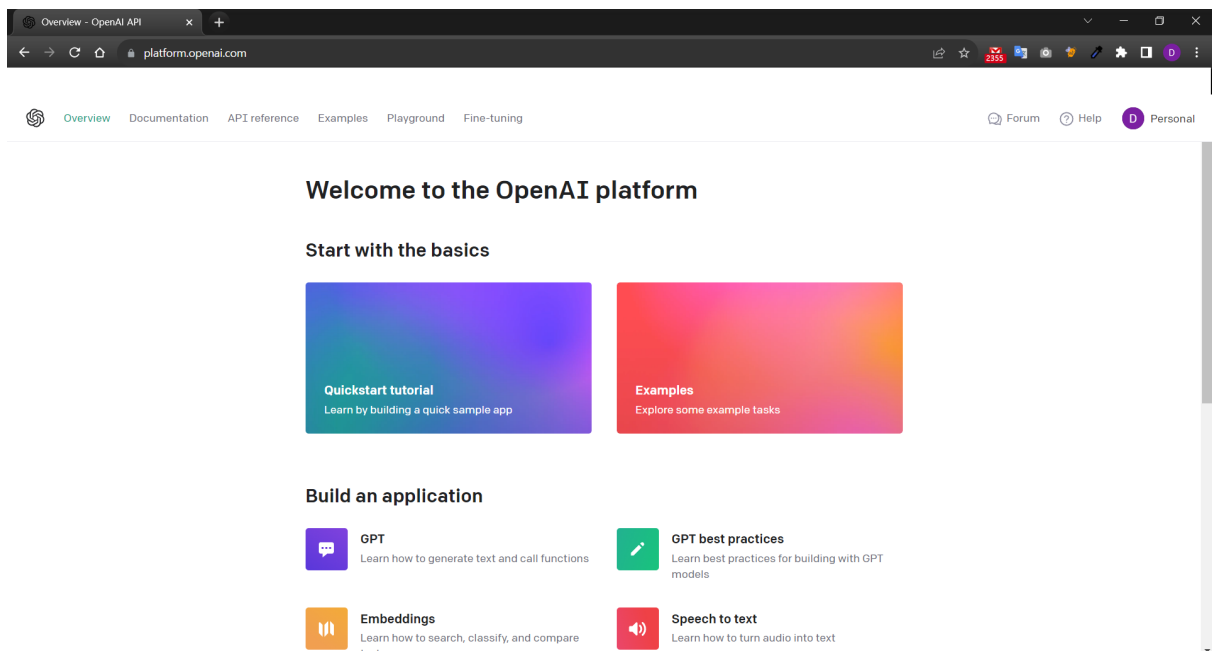
<input type="checkbox"/> 템플릿 ID (Total: 1)	템플릿명	대표 샘플명	배포 종료 시간 (UTC+09:00)	배포 상태	배포
<input type="checkbox"/>	receipt01	sample_receipt01			베타 배포 서비스 배포

8. Request와 Response의 양식을 공식문서를 통해 참조

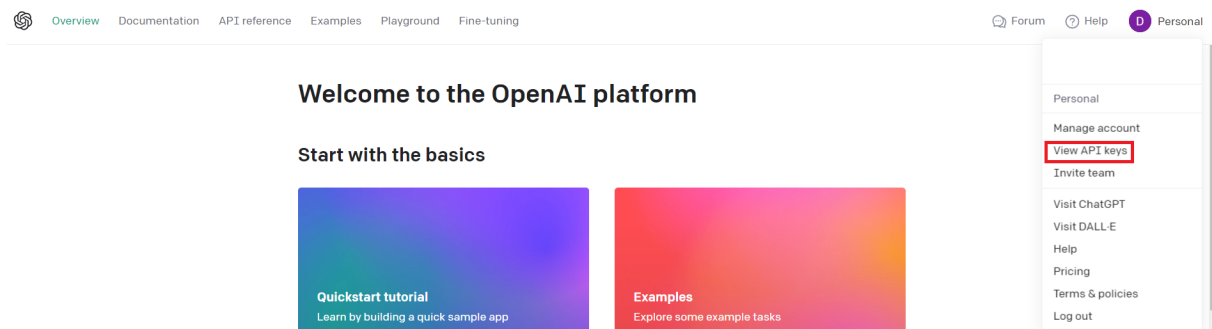
- <https://api.ncloud-docs.com/docs/ai-application-service-ocr-ocr>

4. OpenAI ChatGPT

1. platform.openai.com 접속



2. View API keys 클릭



3. Create new secret key

[Overview](#)
[Documentation](#)
[API reference](#)
[Examples](#)
[Playground](#)
[Fine-tuning](#)

[Forum](#)
[Help](#)

Personal

ORGANIZATION

Personal

Settings

Usage

Rate limits

Members

Billing

USER

Settings

API keys

API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that we've found has leaked publicly.

NAME	KEY	CREATED	LAST USED
<div>+ Create new secret key</div>			

Default organization

If you belong to multiple organizations, this setting controls which organization is used by default when making requests with the API keys above.

Personal

Note: You can also specify which organization to use for each API request. See [Authentication](#) to learn more.

4. ML디렉토리에 .env 파일 생성 후 'apykey' 변수 초기화

