포팅 매뉴얼

▼ 0.a 각 버전

▼ 백엔드 버전

- JDK: 17
- Spring Boot: 3.3.2
- Apache Tomcat: 10.1.28
- MySQL: 8.0.39
- IntelliJ IDEA ultimate 2024.2.0.1
- redis: 7.4.0
- rabbitMQ: 3.7.28

▼ 프론트엔드 버전

- IDE: vscode 1.90 + / webstorm 2024.1

▼ CICD 버전

- 도커: 27.1.1(API: 1.46)
  - Version: 27.3.1
  - API version: 1.47
- 도커 컴포즈: 2.29.7
- 젠킨스: 2.476
- nginx: 1.27.0

▼ 환경변수

▼ [BE] fcm-key.json

```
{
  "type": "service_account",
  "project_id": "fcm-test-b9231",
  "private_key_id": "2393d93a3549e5c93314ca3cafe85746d93661f9",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwg
gSjAgEAAoIBAQDQbf/8/rXZW/QS\neFD556QzwliogvYmrk32/QdpEMZlvXW3QSkDbm48DdSVnMnCE5fcXes
f1EL0z1sb\ng6cDohoHXeYU53dtfAy9RaKv5GOmoPicy3MkyZH6xFz2LIGVtnv5NC46WdqlMNJF\noVd2aJm
tTT0Mq7iosy/tmjl4PvGEgctl3oW1oRwtlSd3sXKiqO0450y8U9ONyiMk\nDy6ZjIO1PHEecuDyJqWv+ElO8
9H/23UozRxn2HEeF67pzPHmUoeI2ywSoOahSwXi\nv39L5rYqJFD+1y367B4LA3aRwM0gpTP4ndZW+hbOtlQ
FgmdOaSJURagwvG/tdnCJ\nwgwphXq1AgMBAAECggEAH+4GBgQgKBbv3kd7wjjz/PDkI5aCbn0L2Q88rCIDf
rEG\nQEdVaN7pOL52p1/QQbC+NiZWH3dlNXsxjha+Rm52FdRS67dk4osbKYCqMk9m268z\nvtNlBYjymcuso
RLC3ERLHCtb/Jjox58hy0Ry8WMHTdb0CAWjgYaHzFqwvKTJ234+\n/DZ3WbRwF8ToGE7gMIl4FVOcSYdlzgm
IiNLvVbEknBAbRADLreCOIRk+q992yOoI\nV8jCbTXpnvBYzU8dqsaKirCKqQQ3EDy+NQvuYWTmpjuc10F96
g6IGv9r15fwH4F+\nO/164ltV+Uw7cvhchJDzzt856ooF75q/yYVZB0LjaQKBgQDnIH1AMQEepHtNefF3\nR
sXGZhWT6Psbkob3kGXEJx+VzqYsmQPER7DDTY5nO3p67V3SPiThop794yIji1Nz\nBgk/hWZWlmic99jxmDi
psNxsI9RBZY/Wct74U4RqRKgSxvGg/SC/mVFD7UtQ2ESD\nH0iaASjRGS0E9PZZFc3xLXwLTQKBgQDm3DSnW
Hq99K4Fb3ffLCMWIIpftew7iXqT\nsqFzJsps1Yd9O9mJ5cvVbVyzJjFio5OKH7X5bk9dN00Jb82SVZgE9x6
JoBL7L0YY\nc+hPqmvj9oJ4SWoxuNttXgITsGM1131boBBsVmwvmLj2OyMRLAK4r/PMlhEhTUdT\n+TS8oWD
pCQKBgQDEz/64LcKVVF0HIYMEHckGZCGVvTwvQ3wYzeZKvUIBMGKC8Z2q\ng527AYz0zUpD6WMOSArh9LfM3
KBSgi8CcK5pX/BAZX4ZxFgBiq0dmtRoZa42LgTp\nCRJjhcLpNyOkMbLKcyJy5Vy1KhkLc84LlhUdx0T5oaW
bZfls7Mtcx22stQKBgHKP\n0ClDRRXCuCDGYgwDkR3H5CL+2k3pOUTWHjxQgZ9kjynca2V8/ZvZ+2iDiXpTh
kHC\n3sD6trOhX79RFgiKJThBkYvmPcUcfoJf8mamQYgMqk4a0HIANBOf1RQEMhdj6jdv\nxPaeJG2yMgcu7
YX1+xu/ZRKfXbzS5pSLVFpPExJRAoGABPXxw5907hM7XiOZg4Ij\ndV1AR3+E3ny1IS2McDLjPjMyrVegcHb
QwmWecl+48DTkTFungtBe8yHoEuyjKMGU\nFGhbCDJfP8BKCySlzj0mFBSmqujHgHbCqWZ3X2nric1LhpAz+
rFksNZaNDMQoIGG\n914SQGhJTdUXZb6QrnDzauE=\n-----END PRIVATE KEY-----\n",
  "client_email": "firebase-adminsdk-b7x6z@fcm-test-b9231.iam.gserviceaccount.com",
  "client_id": "109159763025613210964",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebas
e-adminsdk-b7x6z%40fcm-test-b9231.iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
```

▼ [BE] application.yml env값

```
BACKEND_DB_URL=jdbc:mysql://j11d209.p.ssafy.io:2020/travelus?serverTimezone=Asia/Seo
ul&useUniCode=yes&characterEncoding=UTF-8
BANK_DB_URL=jdbc:mysql://j11d209.p.ssafy.io:2020/tunabank?serverTimezone=Asia/Seoul&
useUniCode=yes&characterEncoding=UTF-8

SERVER_BASE_URL=https://j11d209.p.ssafy.io/api/v1/bank

DB_USERNAME=travelus
DB_PASSWORD=travelus209

TEAM_PRIVATE_KEY=SHINHAN_HACKATHON_LSH_LJJ_HDW_GL
TEAM_FCM_KEY=fcm-key.json
RABBITMQ_USER=travelus
RABBITMQ_PASSWORD=travelus209
REDIS_PASSWORD=travelus209
```

▼ [BE] application-dev.yml

```
team:
  private:
    key: SHINHAN_HACKATHON_LSH_LJJ_HDW_GL
    baseUrl: http://localhost:8080/api/v1/bank
  #    baseUrl: https://j11d209.p.ssafy.io/api/v1/bank


bank:
  private:
    apikey: t9SUPvckGZQc2ozs4A6IS7FVVAPE4IA6ChO0S4ZNcfEO0tkPdE9q88fkgEa84jYG

spring:

  datasource:
    url: jdbc:mysql://localhost:3306/travelus?serverTimezone=Asia/Seoul&useUniCode=y
es&characterEncoding=UTF-8
    username: root
    password: 1234

    #    url: jdbc:mysql://j11d209.p.ssafy.io:2020/travelus?serverTimezone=Asia/Seou
l&useUniCode=yes&characterEncoding=UTF-8
    #    username: travelus
    #    password: travelus209

    flyway:
      enabled: false
  # JPA 설정
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL8Dialect
        format_sql: true
        default_batch_fetch_size: 100 #fetch join 후 지연로딩하는 entity을 in 절로 미리 영
속성 컨텍스트에 저장

  data:
    redis:
      host: localhost
      port: 6379

  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest

server:
  tomcat:
    max-http-form-post-size: 10MB

  front:
    url: http://localhost:3000

  port: 8082
```

▼ [BANK] application-dev.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/tunabank?serverTimezone=Asia/Seoul&useUniCode=y
es&characterEncoding=UTF-8
    username: root
    password: 1234

  data:
    redis:
      host: localhost
      port: 6379

  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest

team:
  private:
    key: SHINHAN_HACKATHON_LSH_LJJ_HDW_GL
    baseUrl: http://localhost:8080/api/v1/bank
```

▼ [FE] .env

```
REACT_APP_JAVASCRIPT_KEY=c931b35ff8483ef03d12cf49163cc32a

# FCM 환경변수

REACT_APP_VAPID_KEY=BEepdPxxuF16h2TcorpIE2aZ0bXtCGQkoz5KzVzES8ZSmyYH86cgku7HvwTsGfNU1T\
REACT_APP_FIREBASE_API_KEY=AIzaSyBL1chfNED_KqZtVt2SNyf0ykE_Zv9wAkQ
REACT_APP_FIREBASE_AUTH_DOMAIN=fcm-test-b9231.firebaseapp.com
REACT_APP_FIREBASE_PROJECT_ID=fcm-test-b9231
REACT_APP_FIREBASE_STORAGE_BUCKET=fcm-test-b9231.appspot.com
REACT_APP_FIREBASE_MESSAGING_SENDER_ID=373701246934
REACT_APP_FIREBASE_APP_ID=1:373701246934:web:7b9f4b48dc4f4f94d96e77
REACT_APP_FIREBASE_MEASUREMENT_ID=G-1B06RZTM09
ubuntu@ip-172-26-6-117:~/env/frontend$
```

▼ MySQL 접속 정보

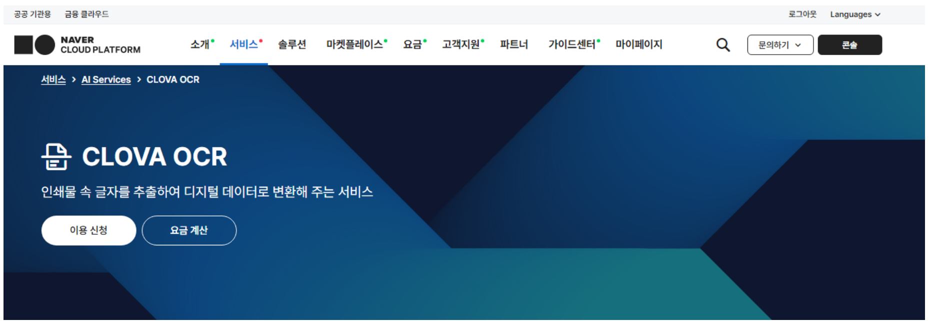
- 계정 id : travelus
- 계정 패스워드 : travelus209
- 사용 DB: travelus, tunabank

▼ 환경변수 정의 파일 리스트

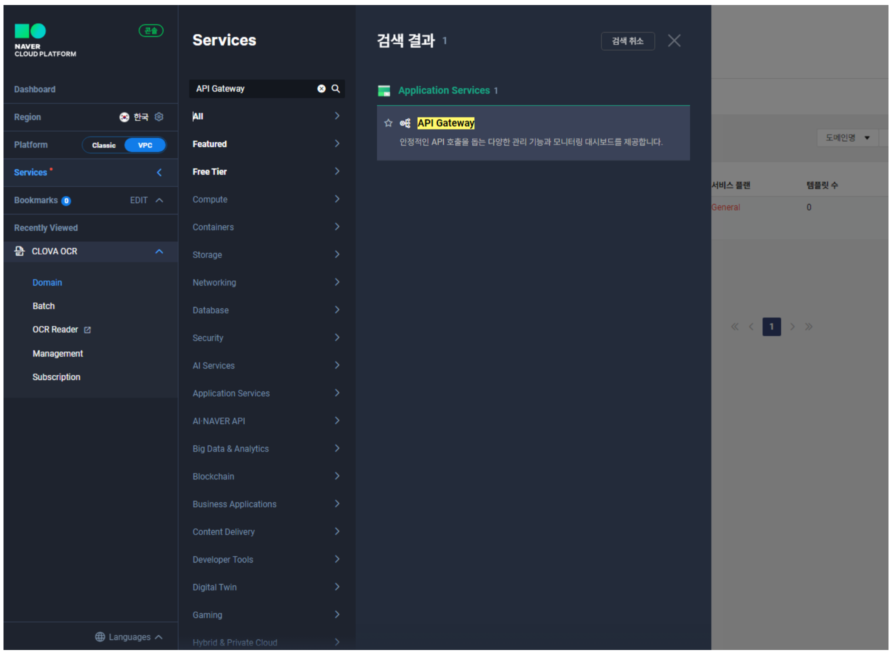
- /backend/src/main/resources/application.yml 및 yml 폴더 내 모든 yml 파일
- /backend/src/main/resources/api-key-encrypted.txt
- /frontend/.env

▼ 0.b 외부 서비스 정보

▼ 네이버 클로바 OCR API 사용 신청하기

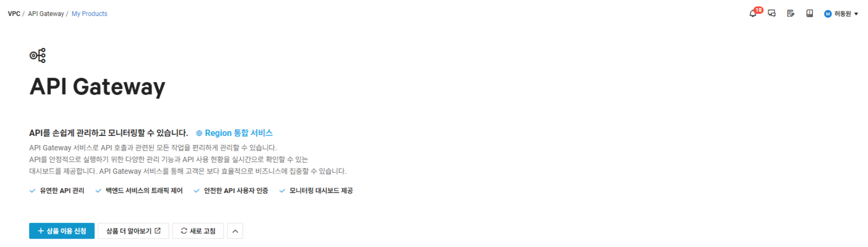


콘솔 접속



안정적인 API 호출을 돕는 다양한 관리 기능과 모니터링 대시보드를 제공합니다.

먼저 클로바 OCR을 이용하기 위해서는 Gateway라는게 있어야 합니다. Services에서 검색해줍니다.

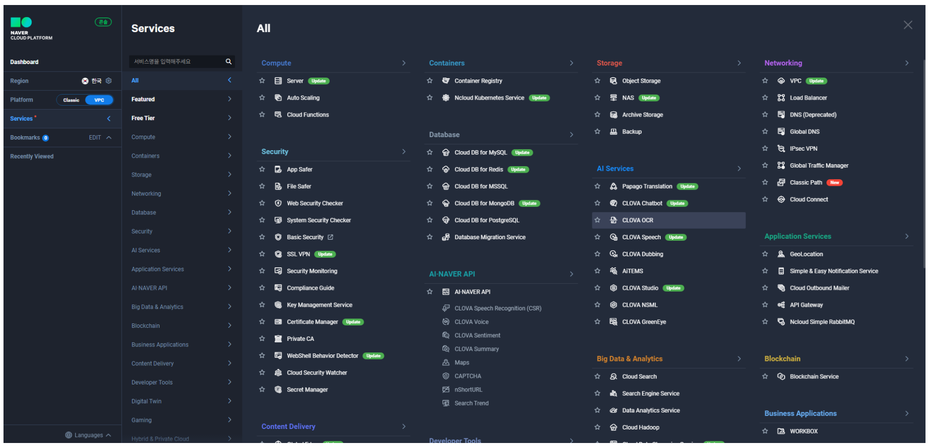


API를 손쉽게 관리하고 모니터링할 수 있습니다. Region 통합 서비스

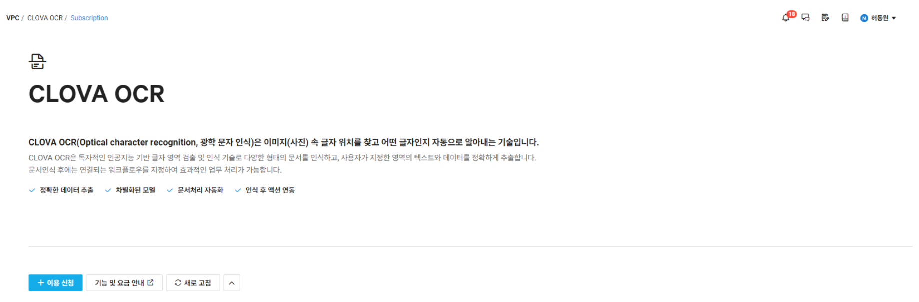
API Gateway 서비스로 API 호출과 관련된 모든 작업을 편리하게 관리할 수 있습니다.
API를 안정적으로 실행하기 위한 다양한 관리 기능과 API 사용 현황을 실시간으로 확인할 수 있는 대시보드를 제공합니다. API Gateway 서비스를 통해 고객은 보다 효율적으로 비즈니스에 집중할 수 있습니다.

✓ 유연한 API 관리 ✓ 백엔드 서비스의 트래픽 제어 ✓ 안전한 API 사용자 인증 ✓ 모니터링 대시보드 제공

`API Gateway 서비스로 API 호출과 관련된 모든 작업을 편리하게 관리할 수 있습니다.` 라고 적혀있네요. 클로바 OCR API를 쉽게 사용할 수 있도록 해주는 도구인 것 같습니다. 이용 신청 ㄱㄱ



services → clova ocr 클릭

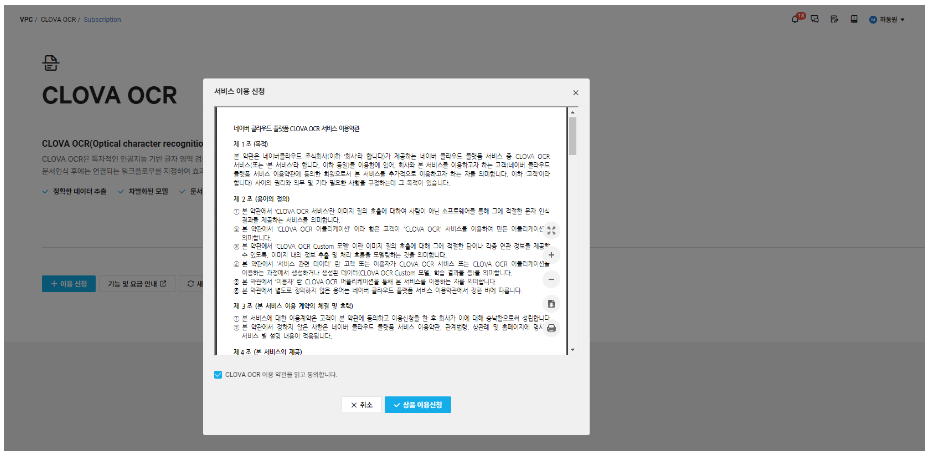


CLOVA OCR

CLOVA OCR(Optical character recognition, 광학 문자 인식)은 이미지(사진) 속 글자 위치를 찾고 어떤 글자인지 자동으로 알아내는 기술입니다.
CLOVA OCR은 독자적인 인공지능 기반 글자 영역 검출 및 인식 기술로 다양한 형태의 문서를 인식하고, 사용자가 지정한 영역의 텍스트와 데이터를 정확하게 추출합니다.
문서인식 후에는 연결되는 워크플로우를 지정하여 효과적인 업무 처리가 가능합니다.

✓ 정확한 데이터 추출 ✓ 차별화된 모델 ✓ 문서처리 자동화 ✓ 인식 후 액션 연동

이용 신청



네이버 클라우드 플랫폼 CLOVA OCR 서비스 이용약관

제 1 조 (목적)
본 약관은 네이버클라우드 주식회사(이하 '회사'라 합니다)가 제공하는 네이버 클라우드 플랫폼 서비스 중 CLOVA OCR 서비스(또는 '본 서비스'라 합니다. 이하 동일)를 이용함에 있어, 회사와 본 서비스를 이용하고자 하는 고객(네이버 클라우드 플랫폼 서비스 이용약관에 동의한 회원으로서 본 서비스를 추가적으로 이용하고자 하는 자를 의미합니다. 이하 '고객'이라 합니다) 사이의 권리와 의무 및 기타 필요한 사항을 규정하는데 그 목적이 있습니다.

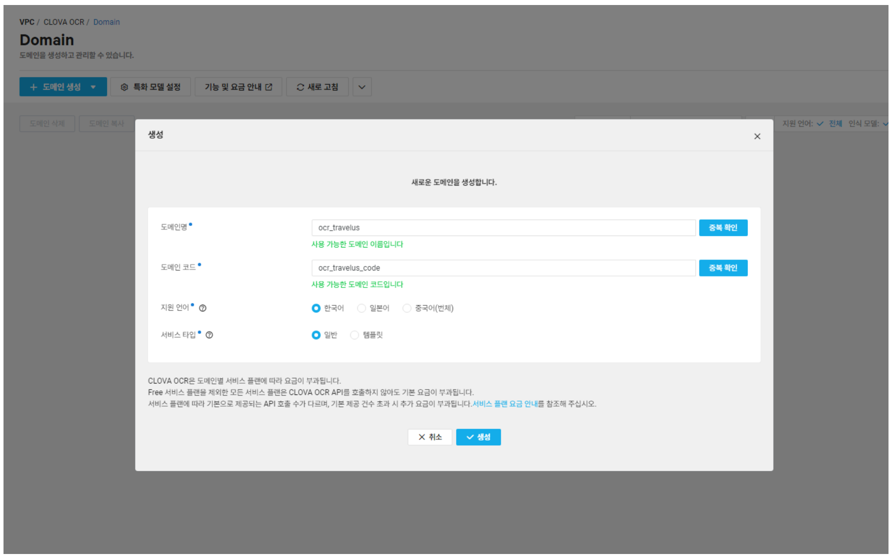
제 2 조 (용어의 정의)
① 본 약관에서 'CLOVA OCR 서비스'란 이미지 질의 호출에 대하여 사람이 아닌 소프트웨어를 통해 그에 적절한 문자 인식 결과를 제공하는 서비스를 의미합니다.
② 본 약관에서 'CLOVA OCR 어플리케이션' 이라 함은 고객이 'CLOVA OCR' 서비스를 이용하여 만든 어플리케이션 의미합니다.
③ 본 약관에서 'CLOVA OCR Custom 모델' 이란 이미지 질의 호출에 대해 그에 적절한 답이나 각종 연관 정보를 제공할 수 있도록, 이미지 내의 정보 추출 및 처리 흐름을 모델링하는 것을 의미합니다.
④ 본 약관에서 '서비스 관련 데이터' 란 고객 또는 이용자가 CLOVA OCR 서비스 또는 CLOVA OCR 어플리케이션을 이용하는 과정에서 생성하거나 생성된 데이터(CLOVA OCR Custom 모델, 학습 결과물 등)를 의미합니다.
⑤ 본 약관에서 '이용자' 란 CLOVA OCR 어플리케이션을 통해 본 서비스를 이용하는 자를 의미합니다.
⑥ 본 약관에서 별도로 정의하지 않은 용어는 네이버 클라우드 플랫폼 서비스 이용약관에서 정한 바에 따릅니다.

제 3 조 (본 서비스 이용 계약의 체결 및 효력)
① 본 서비스에 대한 이용계약은 고객이 본 약관에 동의하고 이용신청을 한 후 회사가 이에 대해 승낙함으로써 성립합니다
② 본 약관에서 정하지 않은 사항은 네이버 클라우드 플랫폼 서비스 이용약관, 관계법령, 상관례 및 홈페이지에 명시 서비스 별 설명 내용이 적용됩니다.

제 4 조 (본 서비스의 제공)

[✓] CLOVA OCR 이용 약관을 읽고 동의합니다.

약관 읽어보시고 상품 이용신청 누르면 잠시 로딩을 거치게 됩니다.



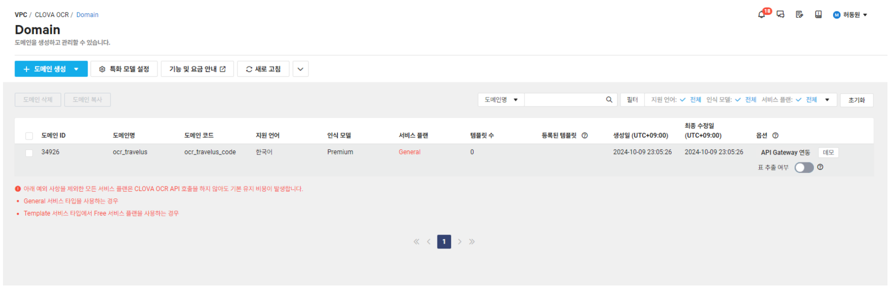
Domain
도메인을 생성하고 관리할 수 있습니다.

새로운 도메인을 생성합니다.

도메인명 •	ocr_travelus (중복 확인) 사용 가능한 도메인 이름입니다
도메인 코드 •	ocr_travelus_code (중복 확인) 사용 가능한 도메인 코드입니다
지원 언어 •	[✓ 한국어] 일본어 중국어(번체)
서비스 타입 •	[✓ 일반] 템플릿

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 안내를 참조해 주십시오.

general, free template 버전 이외 api 타입을 사용한다면 기본 요금이 빠져나가므로 주의합시다.



VPC / CLOVA OCR / Domain

Domain

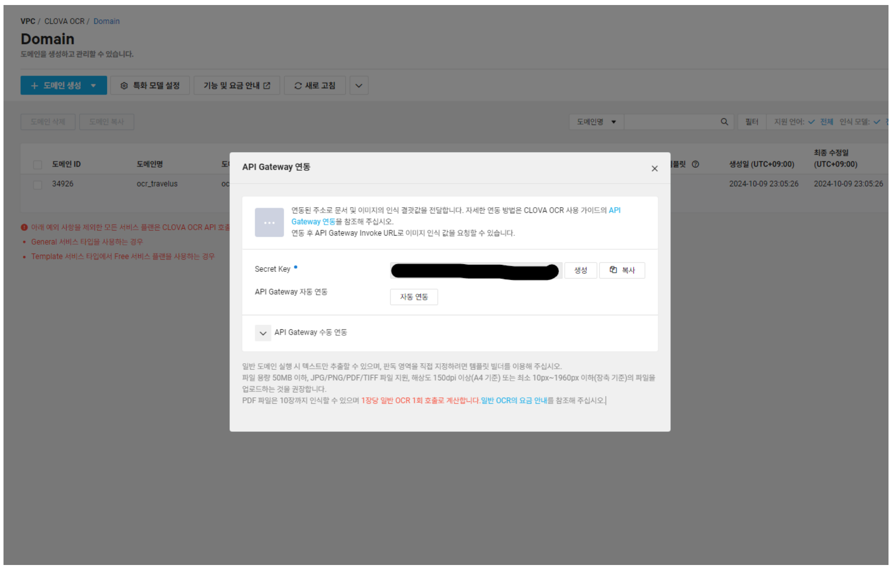
도메인을 생성하고 관리할 수 있습니다.

지원 언어: ✓ 전체 인식 모델: ✓ 전체 서비스 플랜: ✓ 전체

도메인 ID	도메인명	도메인 코드	지원 언어	인식 모델	서비스 플랜	템플릿 수	등록된 템플릿	생성일 (UTC+09:00)	최종 수정일 (UTC+09:00)	옵션
34926	ocr_travelus	ocr_travelus_code	한국어	Premium	General	0		2024-10-09 23:05:26	2024-10-09 23:05:26	API Gateway 연동 데모 표 추출 여부

아래 예외 사항을 제외한 모든 서비스 플랜은 CLOVA OCR API 호출을 하지 않아도 기본 유지 비용이 발생합니다.
- General 서비스 타입을 사용하는 경우
- Template 서비스 타입에서 Free 서비스 플랜을 사용하는 경우

API Gateway 연동 버튼 클릭



API Gateway 연동

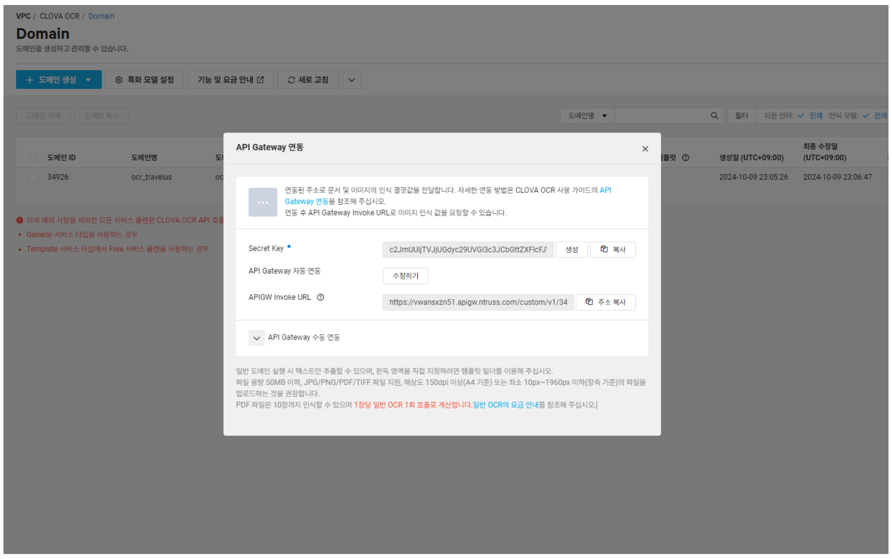
연동된 주소로 문서 및 이미지의 인식 결괏값을 전달합니다. 자세한 연동 방법은 CLOVA OCR 사용 가이드의 API Gateway 연동을 참조해 주십시오.
연동 후 API Gateway Invoke URL로 이미지 인식 값을 요청할 수 있습니다.

Secret Key	[redacted] 생성 복사
API Gateway 자동 연동	자동 연동

API Gateway 수동 연동

일반 도메인 실행 시 텍스트만 추출할 수 있으며, 판독 영역을 직접 지정하려면 템플릿 빌더를 이용해 주십시오.
파일 용량 50MB 이하, JPG/PNG/PDF/TIFF 파일 지원, 해상도 150dpi 이상(A4 기준) 또는 최소 10px~1960px 이하(장축 기준)의 파일을 업로드하는 것을 권장합니다.
PDF 파일은 10장까지 인식할 수 있으며 1장당 일반 OCR 1회 호출로 계산합니다.일반 OCR의 요금 안내를 참조해 주십시오.

오른쪽 생성 버튼을 눌러 Secret Key를 발급받아 주도록 합시다. 이후 자동 연동 버튼을 눌러줍니다.



API Gateway 연동

연동된 주소로 문서 및 이미지의 인식 결괏값을 전달합니다. 자세한 연동 방법은 CLOVA OCR 사용 가이드의 API Gateway 연동을 참조해 주십시오.
연동 후 API Gateway Invoke URL로 이미지 인식 값을 요청할 수 있습니다.

Secret Key	c2JmUUljTVJjUGdyc29UVGl3c3JCbGttZXFlcFJ' 생성 복사
API Gateway 자동 연동	수정하기
APIGW Invoke URL	https://vwansxzn51.apigw.ntruss.com/custom/v1/34 주소 복사

API Gateway 수동 연동

일반 도메인 실행 시 텍스트만 추출할 수 있으며, 판독 영역을 직접 지정하려면 템플릿 빌더를 이용해 주십시오.
파일 용량 50MB 이하, JPG/PNG/PDF/TIFF 파일 지원, 해상도 150dpi 이상(A4 기준) 또는 최소 10px~1960px 이하(장축 기준)의 파일을 업로드하는 것을 권장합니다.
PDF 파일은 10장까지 인식할 수 있으며 1장당 일반 OCR 1회 호출로 계산합니다.일반 OCR의 요금 안내를 참조해 주십시오.

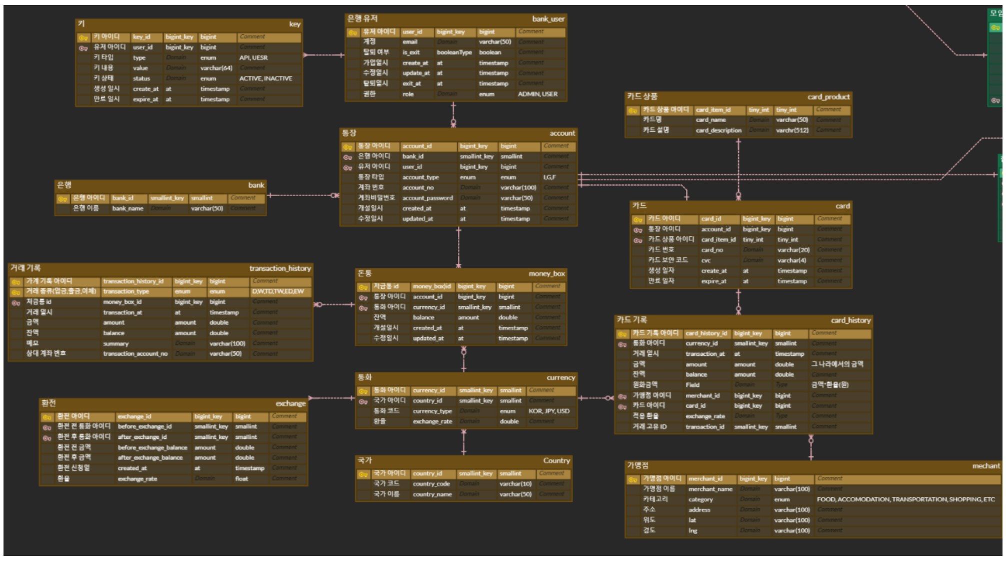
API URL까지 받으면 준비는 끝

▼ S3 버킷 만들기

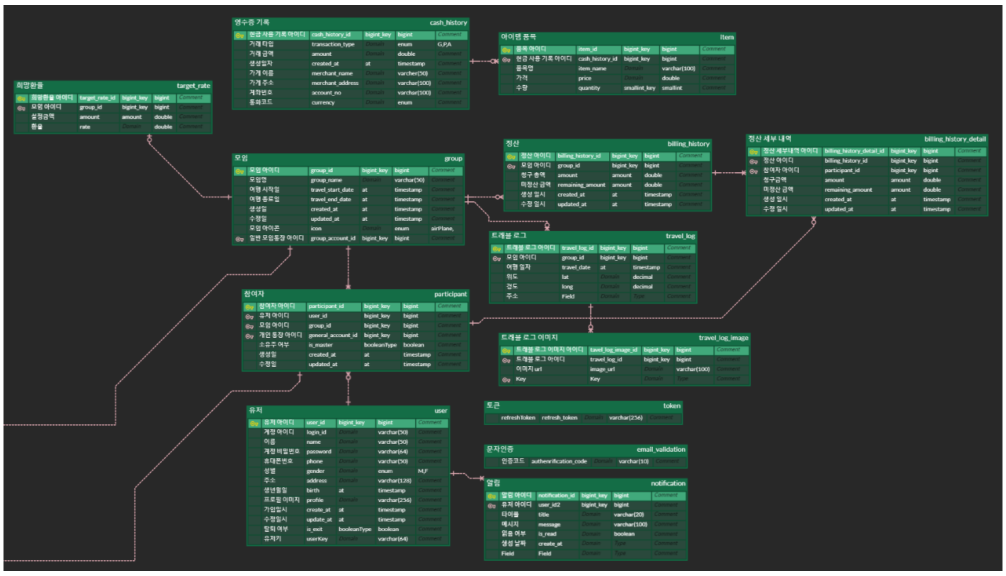
```
1. 버킷 만들기(이름, 리전 등 입력)
2. 버킷 설정(퍼블릭 액세스, 버전 관리, 암호화 등)
3. 버킷 생성
4. 버킷 정책 편집(생성)
```

▼ 1. ERD

TunaBank - 자체 은행 API



TravelUs - 서비스



▼ 2. CI / CD

▼ docker-compose.yml - mysql / redis / nginx / rabbitmq

```yaml
services:

  mysql:
    image: mysql:8.0
    container_name: mysql
    restart: always
    environment:
      MYSQL_USER: travelus
      MYSQL_ROOT_PASSWORD: travelus209
      LANG: C.utf8
      LANGUAGE: C.utf8
      LC_ALL: C.utf8
      TZ: Asia/Seoul
    ports:
      - "2020:3306"
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
      - --bind-address=0.0.0.0
    volumes:
      - /home/ubuntu/mysql-data:/var/lib/mysql
    networks:
      - travelus-network

  nginx:
    image: nginx:1.24.0
    container_name: nginx
    ports:
      - "80:80"  # Nginx의 80번 포트를 호스트의 80번 포트에 매핑
      - "443:443"
    volumes:
      - /home/ubuntu/jenkins-data/workspace/travelus-frontend/travelus/frontend/travelu
s/build:/usr/share/nginx/html
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
      - /etc/letsencrypt/live/j11d209.p.ssafy.io:/etc/letsencrypt/live/j11d209.p.ssafy.i
o:ro
      - /etc/letsencrypt/archive/j11d209.p.ssafy.io:/etc/letsencrypt/archive/j11d209.p.s
safy.io:ro
    networks:
      - travelus-network

  rabbitmq:
    container_name: rabbitmq
    image: rabbitmq:3.7-alpine
    environment:
      - RABBITMQ_USER=travelus
      - RABBITMQ_PASSWORD=travelus209
    ports:
      - "5672:5672"
      - "15672:15672"
    expose:
      - "15672"
    networks:
      - travelus-network

  redis:
    image: redis:latest
    container_name: redis
    ports:
      - "6379:6379"
    environment:
      - REDIS_PASSWORD=travelus209
    volumes:
      - /home/ubuntu/redis-data:/var/lib/redis
    networks:
      - travelus-network

networks:
  travelus-network:
    external: true
```

▼ Jenkins

Dockerfile-Jenkins

```
FROM jenkins/jenkins:jdk17

USER root

RUN apt-get update && \
    apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-pro
perties-common && \
    curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - && \
    add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_
release -cs) stable" && \
    apt-get update && \
    apt-get install -y docker-ce-cli iputils-ping netcat-openbsd && \
    apt-get clean

RUN groupadd -f docker
RUN usermod -aG docker jenkins
USER jenkins
```

jenkins-compose.yml

```
services:
  jenkins:
    build:
      context: .
      dockerfile: Dockerfile-Jenkins
    container_name: jenkins
    user: root  # root 사용자로 실행하여 권한 설정
    restart: always
    volumes:
      - /home/ubuntu/jenkins-data:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - "9090:8080"
      - "50000:50000"
    networks:
      - travelus-network
    environment:
      - TZ=Asia/Seoul
      - DOCKER_HOST=unix:///var/run/docker.sock

volumes:
  jenkins_home:
    driver: local

networks:
  travelus-network:
    external: true
```

▼ TunaBank(자체 은행 API 서버)

bank-compose.yml

무중단 배포를 위해 두개의 포트와 컨테이너를 정의 (Blue - Green)

```

services:
  tunabank-blue:
    image: tunabank-api
    container_name: tunabank-blue
    environment:
      - SPRING_PROFILES_ACTIVE=blue  # 환경 변수는 = 형식을 사용해야 함
      - TZ=Asia/Seoul
    env_file:
      - /home/ubuntu/env/backend/.env
    ports:
      - "8082:8082"  # 호스트 포트 8082, 컨테이너 내 8080
    networks:
      - travelus-network

  tunabank-green:
    image: tunabank-api
    container_name: tunabank-green
    environment:
      - SPRING_PROFILES_ACTIVE=green  # 환경 변수 수정
      - TZ=Asia/Seoul
    env_file:
      - /home/ubuntu/env/backend/.env
    ports:
      - "8083:8083"  # 호스트 포트 8083, 컨테이너 내 8080
    networks:
      - travelus-network

networks:
  travelus-network:
    external: true  # 외부에서 정의된 네트워크 사용 (이미 존재해야 함)
```

▼ TravelUs

backend-compose.yml

무중단 배포를 위해 두개의 포트와 컨테이너를 정의 (Blue - Green)

```
services:

  backend-blue:
    image: travelus-backend
    container_name: backend-blue
    environment:
      - SPRING_PROFILES_ACTIVE=v2,feature,blue
      - TZ=Asia/Seoul
    env_file:
      - /home/ubuntu/env/backend/.env
    ports:
      - "8080:8080"
    networks:
      - travelus-network
    volumes:
      - /home/ubuntu/backend/image:/root/test  # 마운트 설정

  backend-green:
    image: travelus-backend
    container_name: backend-green
    environment:
      - SPRING_PROFILES_ACTIVE=v2,feature,green
      - TZ=Asia/Seoul
    env_file:
      - /home/ubuntu/env/backend/.env
    ports:
      - "8081:8081"
    networks:
      - travelus-network
    volumes:
      - /home/ubuntu/backend/image:/root/test  # 마운트 설정

networks:
  travelus-network:
    external: true  # 외부에서 정의된 네트워크 사용
```

▼ Blue-Green 무중단 배포

TunaBank 무중단 스크립트 파일

```
#!/bin/bash
#1
echo "실행"
EXIST_BLUE=$(docker-compose -f bank-compose.yml ps | grep "tunabank-blue" | grep "runnin
g")

echo "$EXIST_BLUE ==== "

if [ -n "$EXIST_BLUE" ]; then
    echo "Blue server is running. Starting tunabank-green..."
    echo "$EXIST_BLUE"
    docker-compose -f bank-compose.yml up -d tunabank-green
    BEFORE_COLOR="blue"
    AFTER_COLOR="green"
    BEFORE_PORT=8082
    AFTER_PORT=8083
else
    echo "Blue server is not running. Proceeding with tunabank-blue..."
    docker-compose -f bank-compose.yml up -d tunabank-blue
    BEFORE_COLOR="green"
    AFTER_COLOR="blue"
    BEFORE_PORT=8083
    AFTER_PORT=8082
fi

echo "===== ${AFTER_COLOR} server up(port:${AFTER_PORT}) ====="

# 2
for cnt in {1..10}
do
    echo "===== 서버 응답 확인중(${cnt}/10) =====";
    UP=$(curl -s http://j11d209.p.ssafy.io:${AFTER_PORT}/api/v1/bank/actuator/health)
    if [ -z "${UP}" ]
        then
            sleep 10
            continue
        else
            break
    fi
done

if [ $cnt -eq 10 ]
then
    echo "===== 서버 실행 실패 ====="
    exit 1
fi

# 3
echo "===== Nginx 설정 변경 ====="
sed -i "s/${BEFORE_PORT}/${AFTER_PORT}/g" nginx/default.conf && docker exec -it nginx ng
inx -s reload

echo "$BEFORE_COLOR server down(port:${BEFORE_PORT})"
docker-compose -f bank-compose.yml stop tunabank-${BEFORE_COLOR}
docker-compose restart nginx
```

TravelUs 무중단 스크립트 파일

```

#!/bin/bash
#1
echo "실행"
EXIST_BLUE=$(docker-compose -f backend-compose.yml ps | grep "backend-blue" | grep "runn
ing")

echo "$EXIST_BLUE ==== "

if [ -n "$EXIST_BLUE" ]; then
    echo "Blue server is running. Starting backend-green..."
    echo "$EXIST_BLUE"
    docker-compose -f backend-compose.yml up -d backend-green
    BEFORE_COLOR="blue"
    AFTER_COLOR="green"
    BEFORE_PORT=8080
    AFTER_PORT=8081
else
    echo "Blue server is not running. Proceeding with backend-blue..."
    docker-compose -f backend-compose.yml up -d backend-blue
    BEFORE_COLOR="green"
    AFTER_COLOR="blue"
    BEFORE_PORT=8081
    AFTER_PORT=8080
fi

echo "===== ${AFTER_COLOR} server up(port:${AFTER_PORT}) ====="

# 2
for cnt in {1..10}
do
    echo "===== 서버 응답 확인중(${cnt}/10) =====";
    UP=$(curl -s http://j11d209.p.ssafy.io:${AFTER_PORT}/api/v2/actuator/health)
    if [ -z "${UP}" ]
        then
            sleep 10
            continue
        else
            break
    fi
done

if [ $cnt -eq 10 ]
then
    echo "===== 서버 실행 실패 ====="
    exit 1
fi

# 3
echo "===== Nginx 설정 변경 ====="
sed -i "s/${BEFORE_PORT}/${AFTER_PORT}/g" nginx/default.conf && docker exec -it nginx ng
inx -s reload

echo "$BEFORE_COLOR server down(port:${BEFORE_PORT})"
docker-compose -f backend-compose.yml stop backend-${BEFORE_COLOR}
docker-compose restart nginx
```

▼ Nginx

default.conf

```
server {
    listen       443 ssl;
    server_name  travelus.shop www.travelus.shop j11d209.p.ssafy.io;

    location /api/v1/bank/ {
        proxy_pass http://j11d209.p.ssafy.io:8083;
        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_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    location /api/v2/ {
        proxy_pass http://j11d209.p.ssafy.io:8081;
        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_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }

    location /.well-known/acme-challenge/ {
        allow all;
        root /var/www/certbot;
    }
    ssl_certificate /etc/letsencrypt/live/travelus.shop/fullchain.pem; # managed by Cert
bot
    ssl_certificate_key /etc/letsencrypt/live/travelus.shop/privkey.pem; # managed by Ce
rtbot

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}


server {
    if ($host = j11d209.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = www.travelus.shop) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = travelus.shop) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name travelus.shop www.travelus.shop j11d209.p.ssafy.io;

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

nginx.conf

```
user www-data;
worker_processes auto;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    client_max_body_size 50M;
    include /etc/nginx/conf.d/*.conf;
}
```