💈

OlivePay 배포 메뉴얼

1. Docker Compose YAML 파일

```yaml
services:
  eureka:
    container_name: eureka
    build:
      context: .
      dockerfile: backend/cloud/Dockerfile
    ports:
      - "8301:8301"
    networks:
      - olivepay-net
    environment:
      - CLOUD_PORT=8301
      - CLOUD_HOST_NAME=[CLOUD_HOST_NAME]

  donation:
    container_name: donation
    build:
      context: .
      dockerfile: backend/donation/Dockerfile
    ports:
      - "8107:8107"
    depends_on:
      donation-db:
        condition: service_started
      eureka:
        condition: service_started
      gateway:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - BACKEND_SERVER=[BACKEND_SERVER]
      - DB_URL=[DB_URL]
      - DB_USER=[DB_USER]
      - DB_PASSWORD=[DB_PASSWORD]
      - EUREKA_PORT=8301
      - DONATION_PORT=8107
      - EUREKA_REGISTER=[EUREKA_REGISTER]
      - KAFKA_SERVER1=[KAFKA_SERVER1]
      - KAFKA_SERVER2=[KAFKA_SERVER2]
      - KAFKA_SERVER3=[KAFKA_SERVER3]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8107/ac
tuator/health || exit 1"]
      interval: 10s
      timeout: 3s
      retries: 3

  funding:
    container_name: funding
    build:
      context: .
      dockerfile: backend/funding/Dockerfile
    ports:
      - "8106:8106"
    depends_on:
      funding-db:
        condition: service_started
      eureka:
        condition: service_started
      gateway:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - BACKEND_SERVER=[BACKEND_SERVER]
      - DB_URL=[DB_URL]
      - DB_USER=[DB_USER]
      - DB_PASSWORD=[DB_PASSWORD]
      - EUREKA_PORT=8301
      - FUNDING_PORT=8106
      - EUREKA_REGISTER=[EUREKA_REGISTER]
      - FINTECH_URL=[FINTECH_URL]
      - FINTECH_APP_NO=[FINTECH_APP_NO]
      - FINTECH_API_KEY=[FINTECH_API_KEY]
      - INSTITUTION_CODE=[INSTITUTION_CODE]
      - FINTECH_MANAGER_USER_KEY=[FINTECH_MANAGER_USER_KEY]
      - KAFKA_SERVER1=[KAFKA_SERVER1]
      - KAFKA_SERVER2=[KAFKA_SERVER2]
      - KAFKA_SERVER3=[KAFKA_SERVER3]
      - ORGANIZATION_ACCOUNT_NO=[ORGANIZATION_ACCOUNT_NO]
      - DONATION_ACCOUNT_NO=[DONATION_ACCOUNT_NO]
      - CHANGE_ACCOUNT_NO=[CHANGE_ACCOUNT_NO]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8106/ac
tuator/health || exit 1"]
      interval: 10s
      timeout: 3s
      retries: 3

  card:
    container_name: card
    build:
      context: .
      dockerfile: backend/card/Dockerfile
    ports:
      - "8105:8105"
    depends_on:
      card-db:
        condition: service_started
      eureka:
        condition: service_started
      gateway:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - BACKEND_SERVER=[BACKEND_SERVER]
      - DB_URL=[DB_URL]
      - DB_USER=[DB_USER]
      - DB_PASSWORD=[DB_PASSWORD]
      - EUREKA_PORT=8301
      - CARD_PORT=8105
      - EUREKA_REGISTER=[EUREKA_REGISTER]
      - accountTypeUniqueNo=[accountTypeUniqueNo]
      - apiKey=[apiKey]
      - BCCard=[BCCard]
      - DTCard=[DTCard]
      - fintechAppNo=[fintechAppNo]
      - HDCard=[HDCard]
      - HNCard=[HNCard]
      - IBKCard=[IBKCard]
      - institutionCode=[institutionCode]
      - KBCard=[KBCard]
      - LTCard=[LTCard]
      - NHCard=[NHCard]
      - SHCard=[SHCard]
      - SSCard=[SSCard]
      - WRCard=[WRCard]
      - KAFKA_SERVER1=[KAFKA_SERVER1]
      - KAFKA_SERVER2=[KAFKA_SERVER2]
      - KAFKA_SERVER3=[KAFKA_SERVER3]
      - FINTECH_SERVER=[FINTECH_SERVER]
      - OLIVE_USER_KEY=[OLIVE_USER_KEY]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8105/ac
tuator/health || exit 1"]

  member:
    container_name: member
    build:
      context: .
      dockerfile: backend/member/Dockerfile
    ports:
      - "8101:8101"
    depends_on:
      member-db:
        condition: service_started
      member-redis-db:
        condition: service_started
      eureka:
        condition: service_started
      gateway:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - BACKEND_SERVER=[BACKEND_SERVER]
      - MYSQL_URL=[MYSQL_URL]
      - MYSQL_USERNAME=[MYSQL_USERNAME]
      - MYSQL_PASSWORD=[MYSQL_PASSWORD]
      - EUREKA_PORT=8301
      - MEMBER_PORT=8101
      - EUREKA_REGISTER=[EUREKA_REGISTER]
      - FINTECH_SERVER=[FINTECH_SERVER]
      - REDIS_HOST=[REDIS_HOST]
      - REDIS_PASSWORD=[REDIS_PASSWORD]
      - REDIS_PORT=6379
      - apiKey=[apiKey]
      - USER_KEY_DUMMY=[USER_KEY_DUMMY]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8101/ac
tuator/health || exit 1"]

  franchise:
    container_name: franchise
    build:
      context: .
      dockerfile: backend/franchise/Dockerfile
    ports:
      - "8104:8104"
    depends_on:
      franchise-db:
        condition: service_started
      eureka:
        condition: service_started
      gateway:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - BACKEND_SERVER=[BACKEND_SERVER]
      - DB_URL=[DB_URL]
      - DB_USER=[DB_USER]
      - DB_PASSWORD=[DB_PASSWORD]
      - EUREKA_PORT=8301
      - FRANCHISE_PORT=8104
      - EUREKA_REGISTER=[EUREKA_REGISTER]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8104/ac
tuator/health || exit 1"]
      interval: 10s
      timeout: 3s
      retries: 3

  payment:
    container_name: payment
    build:
      context: .
      dockerfile: backend/payment/Dockerfile
    ports:
      - "8103:8103"
    depends_on:
      payment-db:
        condition: service_started
      eureka:
        condition: service_started
      gateway:
          condition: service_started
    networks:
      - olivepay-net
    environment:
      - BACKEND_SERVER=[BACKEND_SERVER]
      - DB_URL=[DB_URL]
      - DB_USER=[DB_USER]
      - DB_PASSWORD=[DB_PASSWORD]
      - EUREKA_PORT=8301
      - PAYMENT_PORT=8103
      - EUREKA_REGISTER=[EUREKA_REGISTER]
      - FINTECH_APP_NO=[FINTECH_APP_NO]
      - INSTITUTION_CODE=[INSTITUTION_CODE]
      - FINTECH_API_KEY=[FINTECH_API_KEY]
      - FINTECH_URL=[FINTECH_URL]
      - KAFKA_GROUP_ID_CONFIG=[KAFKA_GROUP_ID_CONFIG]
      - END_POINT=[END_POINT]
      - TOPIC_PREFIX=[TOPIC_PREFIX]
      - KAFKA_SERVER1=[KAFKA_SERVER1]
      - KAFKA_SERVER2=[KAFKA_SERVER2]
      - KAFKA_SERVER3=[KAFKA_SERVER3]
      - OLIVE_USER_KEY=[OLIVE_USER_KEY]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8103/ac
tuator/health || exit 1"]
      interval: 10s
      timeout: 3s
      retries: 3

  transaction:
    container_name: transaction
    build:
      context: .
      dockerfile: backend/transaction/Dockerfile
    ports:
      - "8302:8302"
    depends_on:
      eureka:
        condition: service_started
      kafka1:
        condition: service_started
      kafka2:
        condition: service_started
      kafka3:
        condition: service_started
      zookeeper:
        condition: service_started
      gateway:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - BACKEND_SERVER=[BACKEND_SERVER]
      - EUREKA_PORT=8301
      - KAFKA_SERVER1=[KAFKA_SERVER1]
      - KAFKA_SERVER2=[KAFKA_SERVER2]
      - KAFKA_SERVER3=[KAFKA_SERVER3]
      - TRANSACTION_PORT=8302
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8302/ac
tuator/health || exit 1"]
      interval: 10s
      timeout: 3s
      retries: 3

  auth:
    container_name: auth
    build:
      context: .
      dockerfile: backend/auth/Dockerfile
    ports:
      - "8102:8102"
    depends_on:
      member-db:
        condition: service_started
      member-redis-db:
        condition: service_started
      gateway:
        condition: service_started
      eureka:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - AUTH_PORT=8102
      - BACKEND_SERVER=[BACKEND_SERVER]
      - EUREKA_PORT=8301
      - FINTECH_SERVER=[FINTECH_SERVER]
      - MYSQL_PASSWORD=[MYSQL_PASSWORD]
      - MYSQL_URL=[MYSQL_URL]
      - MYSQL_USERNAME=[MYSQL_USERNAME]
      - REDIS_HOST=[REDIS_HOST]
      - REDIS_PASSWORD=[REDIS_PASSWORD]
      - REDIS_PORT=6379
      - JWT_SECRET_KEY=[JWT_SECRET_KEY]
      - EUREKA_REGISTER=[EUREKA_REGISTER]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8102/ac
tuator/health || exit 1"]
      interval: 10s
      timeout: 3s
      retries: 3

  common:
    container_name: common
    build:
      context: .
      dockerfile: backend/common/Dockerfile
    ports:
      - "8201:8201"
      - "587:587"
    depends_on:
      common-redis-db:
        condition: service_started
      gateway:
        condition: service_started
      eureka:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - COMMON_PORT=8201
      - BACKEND_SERVER=[BACKEND_SERVER]
      - EUREKA_PORT=8301
      - REDIS_HOST=[REDIS_HOST]
      - REDIS_PASSWORD=[REDIS_PASSWORD]
      - REDIS_PORT=6379
      - EUREKA_REGISTER=[EUREKA_REGISTER]
      - SMS_API_KEY=[SMS_API_KEY]
      - SMS_API_SECRET=[SMS_API_SECRET]
      - SMS_SENDER=[SMS_SENDER]
      - SMS_PROVIDER=[SMS_PROVIDER]
      - CLOVA_SECRET=[CLOVA_SECRET]
      - CLOVA_URL=[CLOVA_URL]
      - MAIL_USER=[MAIL_USER]
      - MAIL_PASSWORD=[MAIL_PASSWORD]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8201/ac
tuator/health || exit 1"]
      interval: 10s
      timeout: 3s
      retries: 3

  gateway:
    container_name: gateway
    build:
      context: .
      dockerfile: backend/gateway/Dockerfile
    ports:
      - "8000:8000"
    depends_on:
      eureka:
        condition: service_started
    networks:
      - olivepay-net
    environment:
      - JWT_SECRET_KEY=[JWT_SECRET_KEY]
      - MEMBER_SERVER=[MEMBER_SERVER]
      - MEMBER_PATH=[MEMBER_PATH]
      - MEMBER_PORT=8101
      - MEMBER_SCHEME=http
      - REDIS_HOST=[REDIS_HOST]
      - REDIS_PASSWORD=[REDIS_PASSWORD]
      - REDIS_PORT=6379
      - BACKEND_SERVER=[BACKEND_SERVER]
      - EUREKA_PORT=8301
      - GATEWAY_PORT=8000
      - GATEWAY_USER=[GATEWAY_USER]
      - GATEWAY_PASSWORD=[GATEWAY_PASSWORD]
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8000/ac
tuator/health || exit 1"]

  zookeeper:
    image: zookeeper:latest
    container_name: zookeeper
    ports:
      - "2181:2181"
    networks:
      - olivepay-net

  kafka1:
    image: wurstmeister/kafka:2.13-2.8.1
    container_name: kafka1
    ports:
      - "9092:9092"
      - "29092:29092"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ADVERTISED_HOST_NAME=kafka1
      KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka1:9092,E
XTERNAL://localhost:29092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAIN
TEXT,EXTERNAL:PLAINTEXT
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092,EXTERNAL://
0.0.0.0:29092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 2
    networks:
      - olivepay-net

  kafka2:
    image: wurstmeister/kafka:2.13-2.8.1
    container_name: kafka2
    ports:
      - "9093:9093"
      - "29093:29093"
    environment:
      KAFKA_BROKER_ID: 2
      KAFKA_ADVERTISED_HOST_NAME=kafka2
      KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka2:9093,E
XTERNAL://localhost:29093
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAIN
TEXT,EXTERNAL:PLAINTEXT
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9093,EXTERNAL://
0.0.0.0:29093
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 2
    networks:
      - olivepay-net

  kafka3:
    image: wurstmeister/kafka:2.13-2.8.1
    container_name: kafka3
    ports:
      - "9094:9094"
      - "29094:29094"
    environment:
      KAFKA_BROKER_ID: 3
      KAFKA_ADVERTISED_HOST_NAME=kafka3
      KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka3:9094,E
XTERNAL://localhost:29094
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAIN
TEXT,EXTERNAL:PLAINTEXT
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9094,EXTERNAL://
0.0.0.0:29094
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 2
    networks:
      - olivepay-net

  donation-db:
    image: mysql:latest
    container_name: donation-db
    environment:
      MYSQL_ROOT_PASSWORD=[MYSQL_ROOT_PASSWORD]
      MYSQL_DATABASE=donation
      MYSQL_USER=donation
      MYSQL_PASSWORD=[MYSQL_PASSWORD]
      TZ: Asia/Seoul
    ports:
      - "3107:3306"
    volumes:
      - donation_data:/var/lib/mysql
    networks:
      - olivepay-net

  member-db:
    image: mysql:latest
    container_name: member-db
    environment:
      MYSQL_ROOT_PASSWORD=[MYSQL_ROOT_PASSWORD]
      MYSQL_DATABASE=member
      MYSQL_USER=member
      MYSQL_PASSWORD=[MYSQL_PASSWORD]
      TZ: Asia/Seoul
    ports:
      - "3101:3306"
    volumes:
      - member_data:/var/lib/mysql
    networks:
      - olivepay-net

  card-db:
    image: mysql:latest
    container_name: card-db
    environment:
      MYSQL_ROOT_PASSWORD=[MYSQL_ROOT_PASSWORD]
      MYSQL_DATABASE=card
      MYSQL_USER=card
      MYSQL_PASSWORD=[MYSQL_PASSWORD]
      TZ: Asia/Seoul
    ports:
      - "3105:3306"
    volumes:
      - card_data:/var/lib/mysql
    networks:
      - olivepay-net

  franchise-db:
    image: mysql:latest
    container_name: franchise-db
    environment:
      MYSQL_ROOT_PASSWORD=[MYSQL_ROOT_PASSWORD]
      MYSQL_DATABASE=franchise
      MYSQL_USER=franchise
      MYSQL_PASSWORD=[MYSQL_PASSWORD]
      TZ: Asia/Seoul
    ports:
      - "3104:3306"
    volumes:
      - franchise_data:/var/lib/mysql
    networks:
      - olivepay-net
```

```
funding-db:
  image: mysql:latest
  container_name: funding-db
  environment:
    MYSQL_ROOT_PASSWORD=[MYSQL_ROOT_PASSWORD]
    MYSQL_DATABASE=funding
    MYSQL_USER=funding
    MYSQL_PASSWORD=[MYSQL_PASSWORD]
    TZ: Asia/Seoul
  ports:
    - "3106:3306"
  volumes:
    - funding_data:/var/lib/mysql
  networks:
    - olivepay-net

payment-db:
  image: mysql:latest
  container_name: payment-db
  environment:
    MYSQL_ROOT_PASSWORD=[MYSQL_ROOT_PASSWORD]
    MYSQL_DATABASE=payment
    MYSQL_USER=payment
    MYSQL_PASSWORD=[MYSQL_PASSWORD]
    TZ: Asia/Seoul
  ports:
    - "3103:3306"
  volumes:
    - payment_data:/var/lib/mysql
  networks:
    - olivepay-net

member-redis-db:
  image: redis:latest
  command: redis-server --requirepass [REDIS_PASSWORD]
  environment:
    REDIS_PASSWORD=[REDIS_PASSWORD]
    ports:
      - "6101:6379"
    volumes:
      - member_redis_data:/data
    networks:
      - olivepay-net

  common-redis-db:
    image: redis:latest
    command: redis-server --requirepass [REDIS_PASSWORD]
    environment:
      REDIS_PASSWORD=[REDIS_PASSWORD]
    ports:
      - "6201:6379"
    volumes:
      - common_redis_data:/data
    networks:
      - olivepay-net

networks:
  olivepay-net:

volumes:
  donation_data:
  card_data:
  member_data:
  member_redis_data:
  franchise_data:
  funding_data:
  payment_data:
  common_redis_data:
```

2. 서비스 포트 목록

서비스 이름	포트 번호
`eureka`	8301

`donation`	8107
`funding`	8106
`card`	8105
`member`	8101
`franchise`	8104
`payment`	8103
`transaction`	8302
`auth`	8102
`common`	8201
`gateway`	8000
`zookeeper`	2181
`kafka1`	9092 (내부), 29092 (외부)
`kafka2`	9093 (내부), 29093 (외부)
`kafka3`	9094 (내부), 29094 (외부)

---

3. Backend 배포 및 실행 방법

1. **Docker 및 Docker Compose 설치**

   Docker와 Docker Compose가 설치되어 있는지 확인합니다. 설치되어 있지 않다면 아래 명령어로 설치할 수 있습니다:

   ```
   # Docker 설치
   sudo apt update
   sudo apt install docker.io

   # Docker Compose 설치
   sudo apt install docker-compose
   ```

2. **프로젝트 클론**
   olivepay 프로젝트를 클론하고 back/master 브랜치로 체크아웃 합니다.

   ```
   git clone https://lab.ssafy.com/s11-fintech-finance-sub
   1/S11P21A601
   git checkout back/master
   ```

3. **Docker Compose yml을 프로젝트의 최상단에 복사합니다.**
4. **Docker Compose 실행**
   모든 컨테이너를 동시에 실행하기 위해 Docker Compose 명령어를 사용합니다:

   ```
   docker-compose up --build
   ```

---

4. Frontend 배포 및 실행 방법

1. **프로젝트 클론**
   olivepay 프로젝트를 클론하고 back/master 브랜치로 체크아웃 합니다.

```
git clone https://lab.ssafy.com/s11-fintech-finance-sub1/S11P
git checkout back/master
```

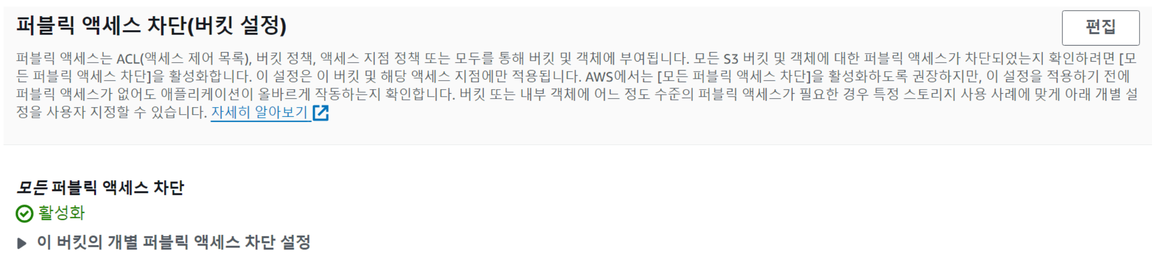
2. **Vite 빌드**

frontend 폴더로 이동하고, 빌드합니다.

```
npm install
npm run build
```

3. S3 업로드

- AWS S3 bucket에 `dist 폴더` 를 업로드 합니다.
- bucket의 경우, 퍼블릭 액세스를 차단합니다.

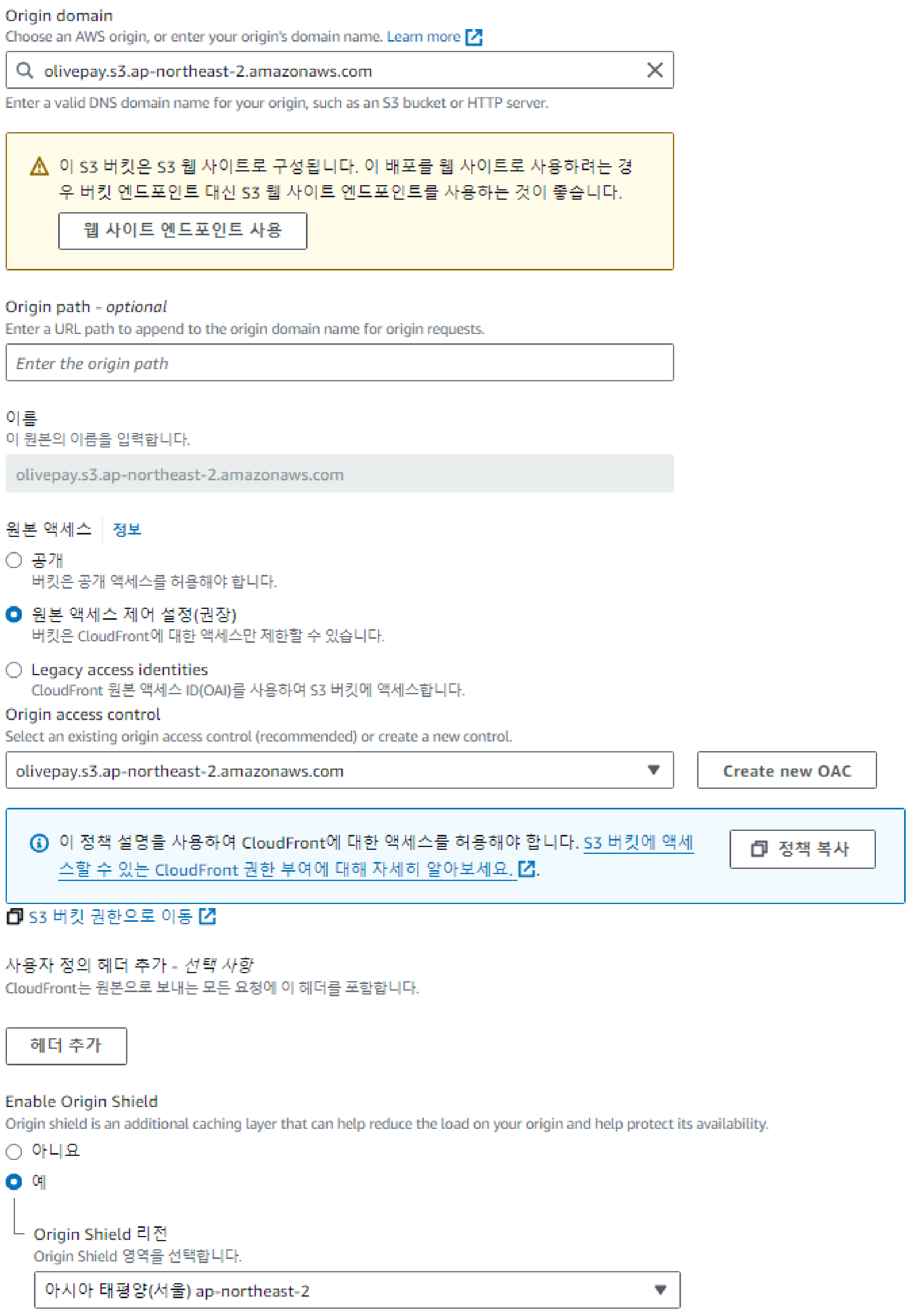


퍼블릭 액세스 차단(버킷 설정)

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 모든 S3 버킷 및 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 [모든 퍼블릭 액세스 차단]을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 [모든 퍼블릭 액세스 차단]을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 내부 객체에 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**
✓ 활성화
▶ 이 버킷의 개별 퍼블릭 액세스 차단 설정

4. **CloudFront의 CDN 활성화**


Origin domain
Choose an AWS origin, or enter your origin's domain name. Learn more

olivepay.s3.ap-northeast-2.amazonaws.com

Enter a valid DNS domain name for your origin, such as an S3 bucket or HTTP server.

⚠ 이 S3 버킷은 S3 웹 사이트로 구성됩니다. 이 배포를 웹 사이트로 사용하려는 경우 버킷 엔드포인트 대신 S3 웹 사이트 엔드포인트를 사용하는 것이 좋습니다.

웹 사이트 엔드포인트 사용

Origin path - *optional*
Enter a URL path to append to the origin domain name for origin requests.

Enter the origin path

이름
이 원본의 이름을 입력합니다.

olivepay.s3.ap-northeast-2.amazonaws.com

원본 액세스 | 정보

○ 공개
버킷은 공개 액세스를 허용해야 합니다.

● 원본 액세스 제어 설정(권장)
버킷은 CloudFront에 대한 액세스만 제한할 수 있습니다.

○ Legacy access identities
CloudFront 원본 액세스 ID(OAI)를 사용하여 S3 버킷에 액세스합니다.

Origin access control
Select an existing origin access control (recommended) or create a new control.

olivepay.s3.ap-northeast-2.amazonaws.com ▼ | Create new OAC

ⓘ 이 정책 설명을 사용하여 CloudFront에 대한 액세스를 허용해야 합니다. S3 버킷에 액세스할 수 있는 CloudFront 권한 부여에 대해 자세히 알아보세요. | 정책 복사

S3 버킷 권한으로 이동

사용자 정의 헤더 추가 - *선택 사항*
CloudFront는 원본으로 보내는 모든 요청에 이 헤더를 포함합니다.

헤더 추가

Enable Origin Shield
Origin shield is an additional caching layer that can help reduce the load on your origin and help protect its availability.

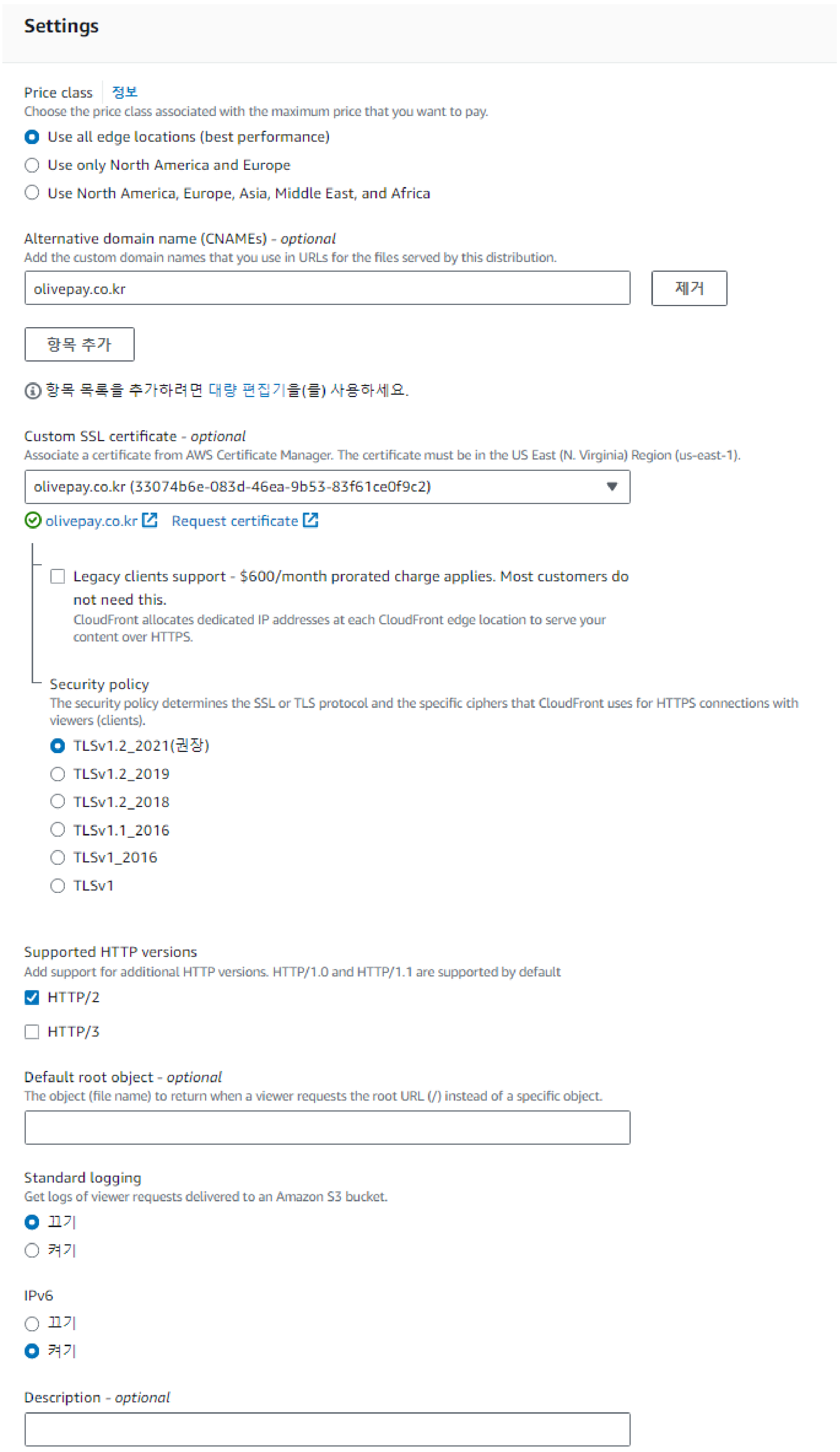
○ 아니요
● 예

Origin Shield 리전
Origin Shield 영역을 선택합니다.

아시아 태평양(서울) ap-northeast-2 ▼

5. CNAME 설정

- 구매한 도메인의 CNAME을 설정합니다.



Settings

Price class 정보

Choose the price class associated with the maximum price that you want to pay.

- [✓ Use all edge locations (best performance)]
- Use only North America and Europe
- Use North America, Europe, Asia, Middle East, and Africa

Alternative domain name (CNAMEs) - *optional*

Add the custom domain names that you use in URLs for the files served by this distribution.

olivepay.co.kr

ⓘ 항목 목록을 추가하려면 대량 편집기을(를) 사용하세요.

Custom SSL certificate - *optional*

Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

olivepay.co.kr (33074b6e-083d-46ea-9b53-83f61ce0f9c2)

✓ olivepay.co.kr Request certificate

- ☐ Legacy clients support - $600/month prorated charge applies. Most customers do not need this.
  CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS.

Security policy

The security policy determines the SSL or TLS protocol and the specific ciphers that CloudFront uses for HTTPS connections with viewers (clients).

- [✓ TLSv1.2_2021(권장)]
- TLSv1.2_2019
- TLSv1.2_2018
- TLSv1.1_2016
- TLSv1_2016
- TLSv1

Supported HTTP versions

Add support for additional HTTP versions. HTTP/1.0 and HTTP/1.1 are supported by default

- ☑ HTTP/2
- ☐ HTTP/3

Default root object - *optional*

The object (file name) to return when a viewer requests the root URL (/) instead of a specific object.

Standard logging

Get logs of viewer requests delivered to an Amazon S3 bucket.

- [✓ 끄기]
- 켜기

IPv6

- 끄기
- [✓ 켜기]

Description - *optional*