

클라우드 기반 백엔드 스프링부트 마이크로서비스 개발자 양성

한국IT교육원

Gopang Project

(쿠팡 클론 프로젝트)

이 태훈

Project Overview

▶ 쿠팡 API 조사

API 명세서

표 7.1 도서 대출 API

API명	도서 대출
리소스 URI	/rentals/{userid}/rentedItem/{book}
메서드	POST
요청 매개변수	사용자 일련번호, 도서 일련번호
요청 예시	http://localhost:8088/rentals/5/rentedItem/10001 정상 처리 시 도서대출정보를 반환 ▪ Rental 일련번호 ▪ 사용자 일련번호 ▪ 대출 가능 여부 ▪ 연체료
응답 결과	{ "id": 1, "userId": 5, "rentalStatus": "RENT_AVAILABLE", "lateFee": 0 }
응답 예시	

쿠팡에서 제공하는 api 사이트

API Docs – Open APIs

[2021/07/07 OPENAPI 서비스 장애 안내 / OPENAPI Service Disability Information]
 <https://developers.coupangcorp.com/hc/ko/categories/380002105414-API-Docs>

문제는 이 곳에서 로그인, 회원관리에 대한 api 부재...

그리고 또 다른 문제는 몇년전까지만 해도 쿠팡은 사이트주소에 URL노출이 되었지만 지금은 암호화되어 복호화를 하지않는 이상 정확한 주소를 알기가 어려워졌다

→ 일부 암호화가 필요한 부분만 처리되어있는걸 확인(개인정보가 돌아다니는쪽은 보안이 철저해짐)

gpt에게 필요한 api에 대해 질문 던졌고 받은 답변

▼ 펼치기

사용자 관리(User Management)와 관련된 더 자세한 REST API 엔드포인트를 다음과 같이 나눌 수 있습니다:

1. 사용자 등록 및 로그인:

- 사용자 등록:
 - POST /api/users/register: 새로운 사용자 등록
- 사용자 로그인:
 - POST /api/users/login: 사용자 로그인 및 인증
- 사용자 로그아웃:
 - POST /api/users/logout: 사용자 로그아웃

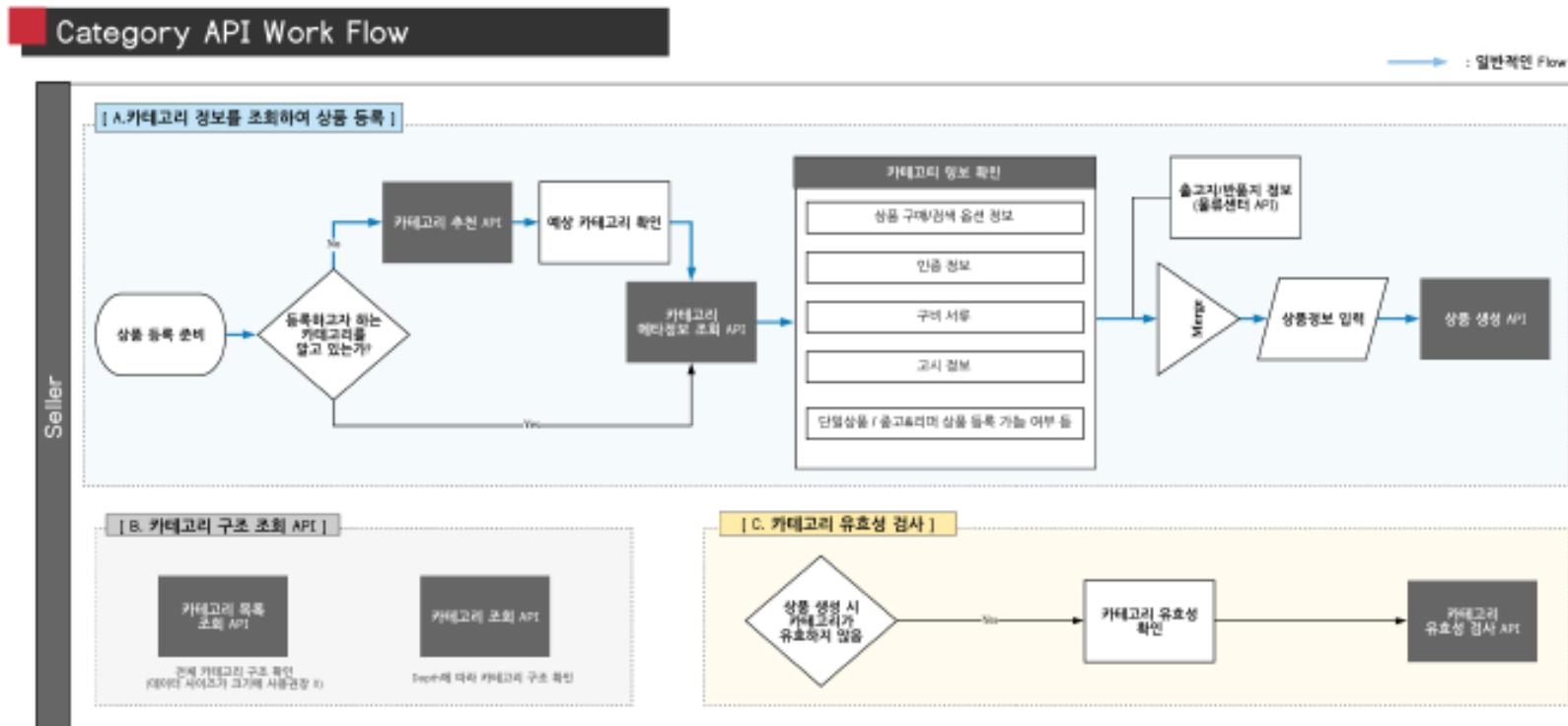
2. 사용자 프로필 조회 및 편집:

- 사용자 프로필 조회:
 - GET /api/users/{userID}/profile: 특정 사용자 프로필 조회
- 사용자 프로필 편집:
 - PUT /api/users/{userID}/profile: 특정 사용자 프로필 업데이트

3. 주문 기록 및 주문 상태 조회:

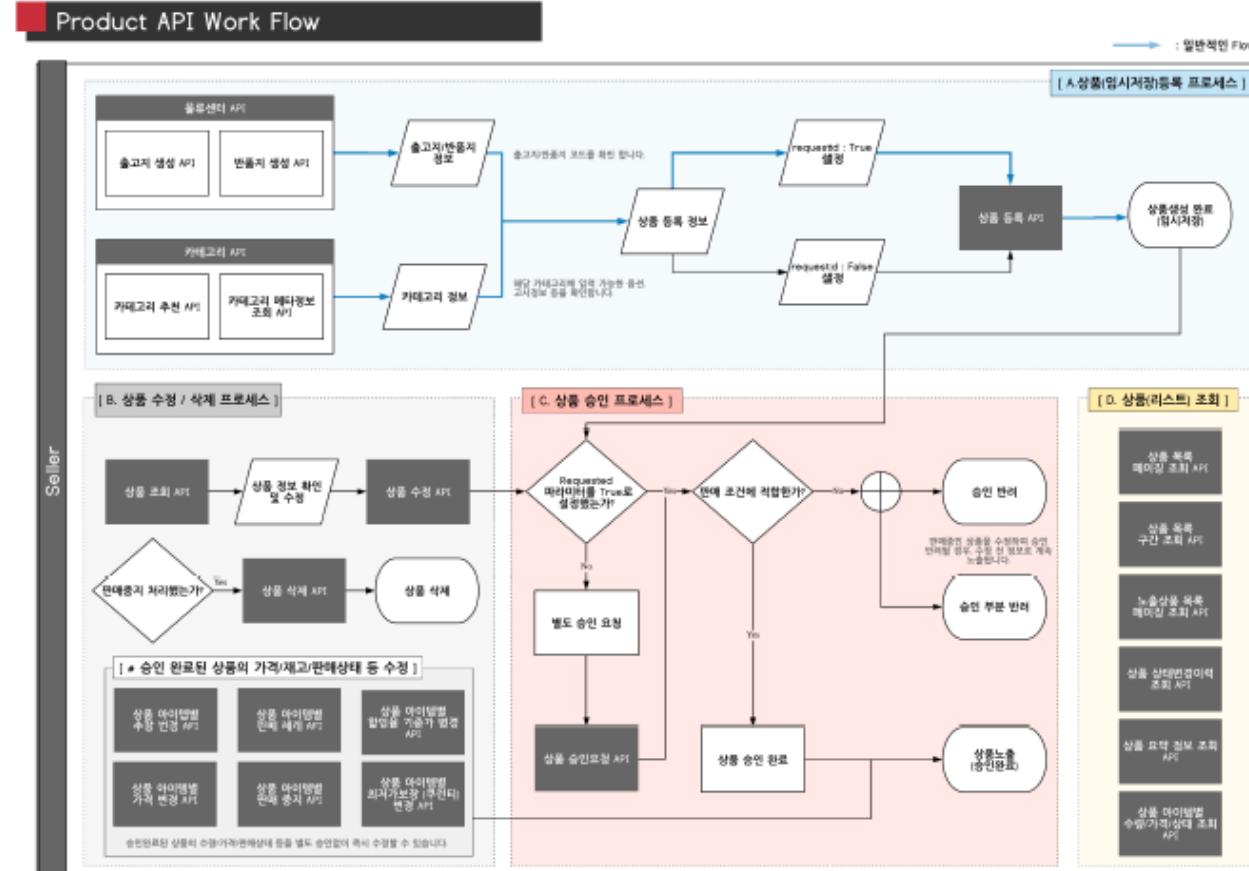
- 주문 기록 조회:
 - GET /api/users/{userID}/orders: 특정 사용자의 주문 목록 조회
- 주문 상세 정보 조회:
 - GET /api/users/{userID}/orders/{orderId}: 특정 주문의 상세 정보 조회

▶ 쿠팡 Workflow

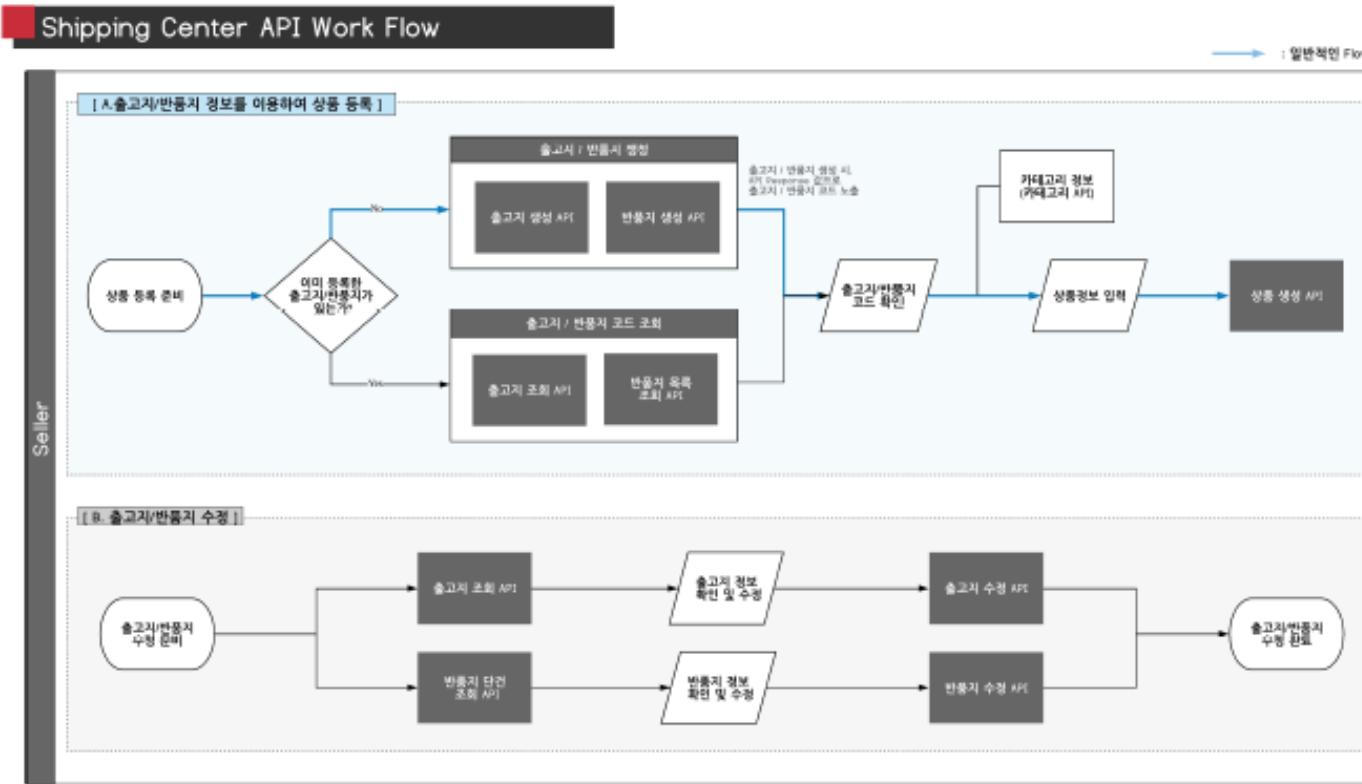


카테고리 API 워크플로우

▶ 쿠팡 Workflow



▶ 쿠팡 Workflow



상품등록 API 워크플로우

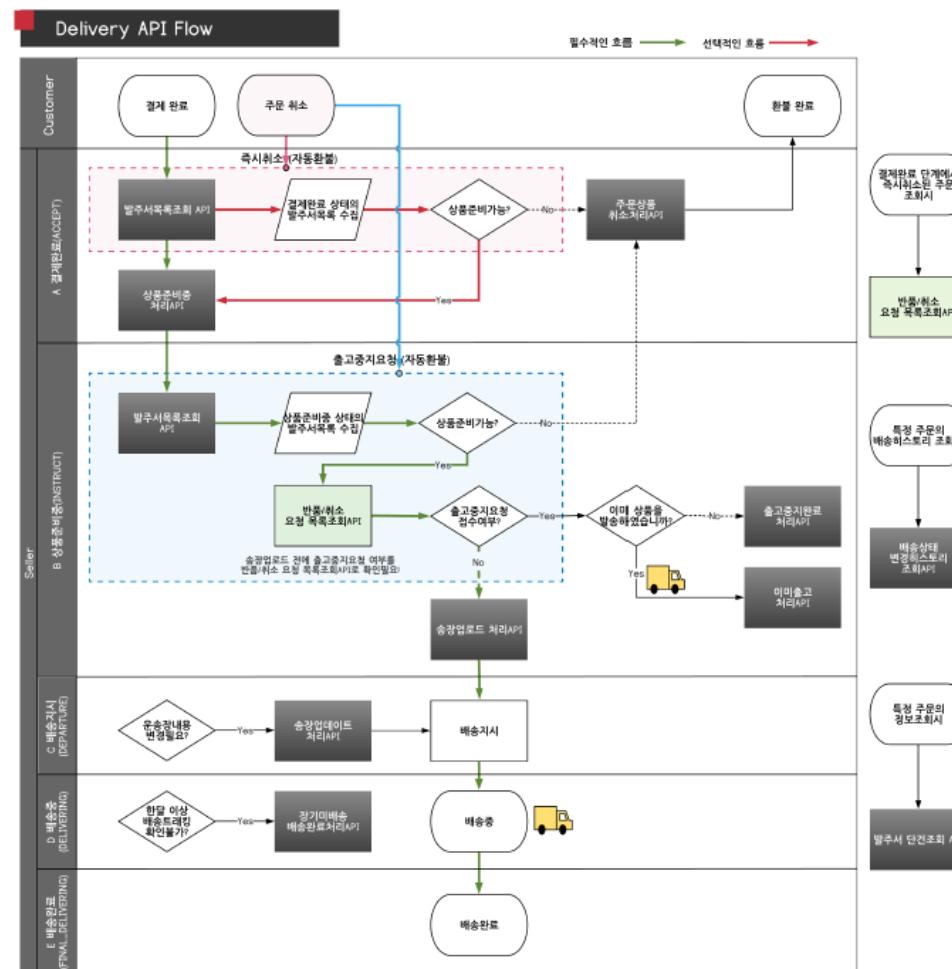
01

프로젝트 개요

기획단계

▶ 쿠팡 Workflow

Planning and Jira



배송 API 워크플로우

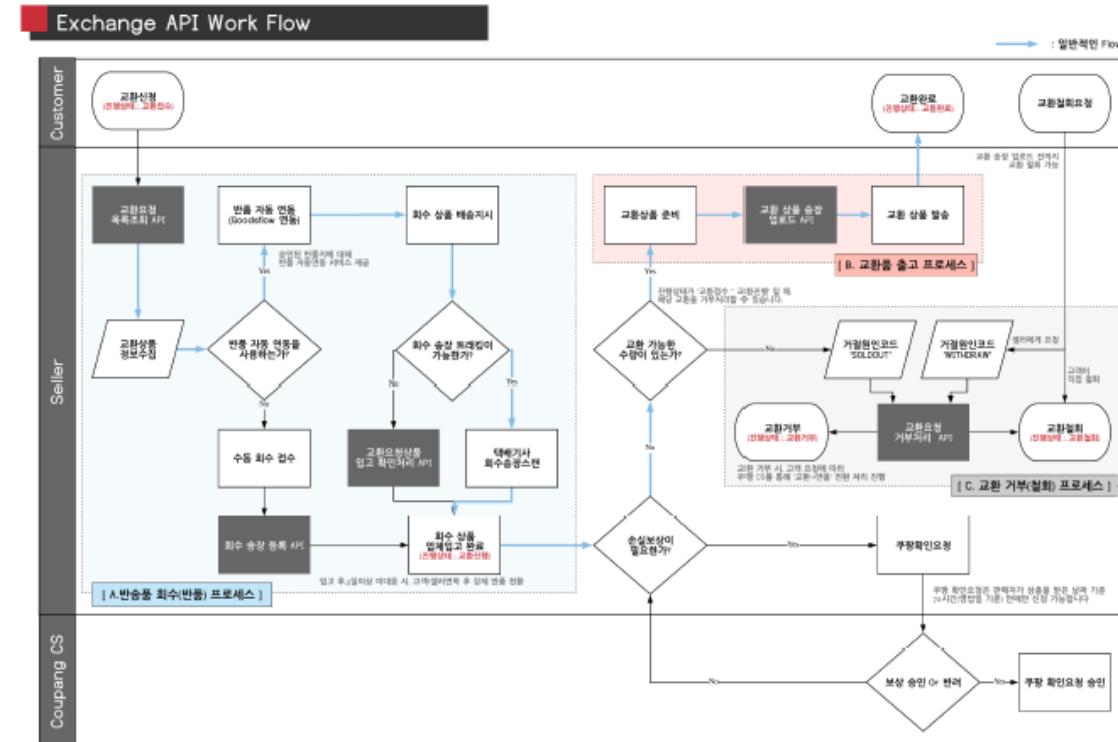
01

프로젝트 개요

기획단계

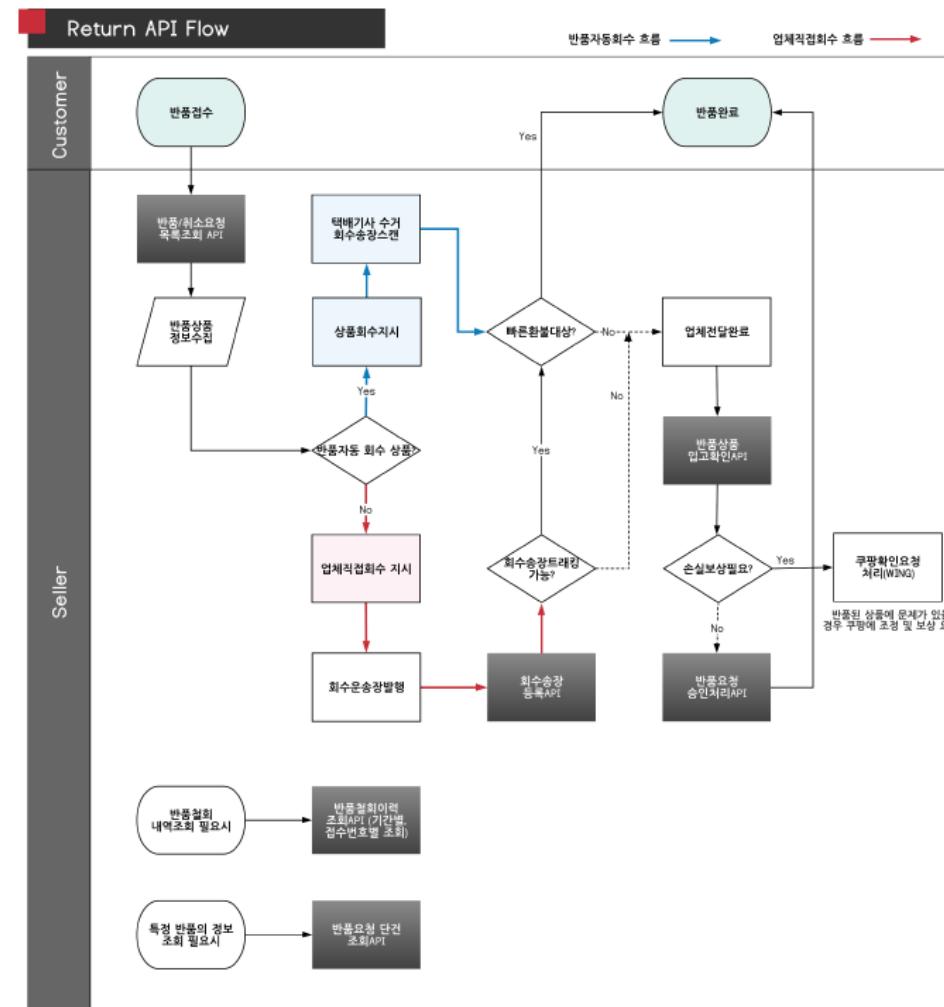
Planning and Jira

▶ 쿠팡 Workflow

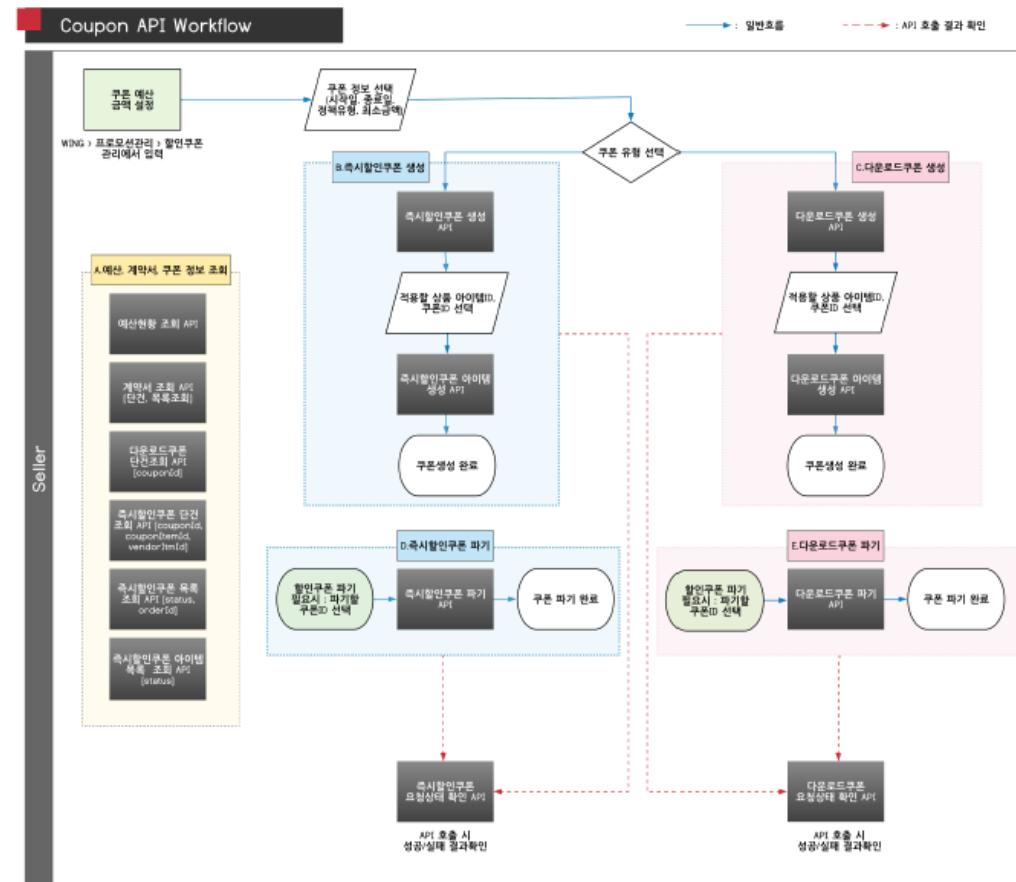


교환 API 워크플로우

▶ 쿠팡 Workflow

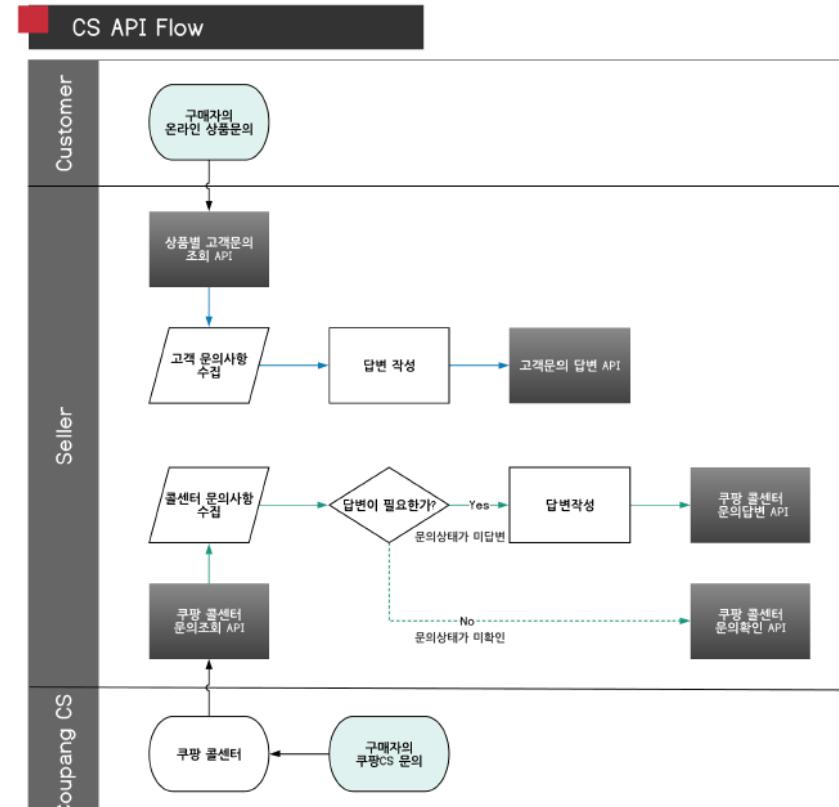


▶ 쿠팡 Workflow



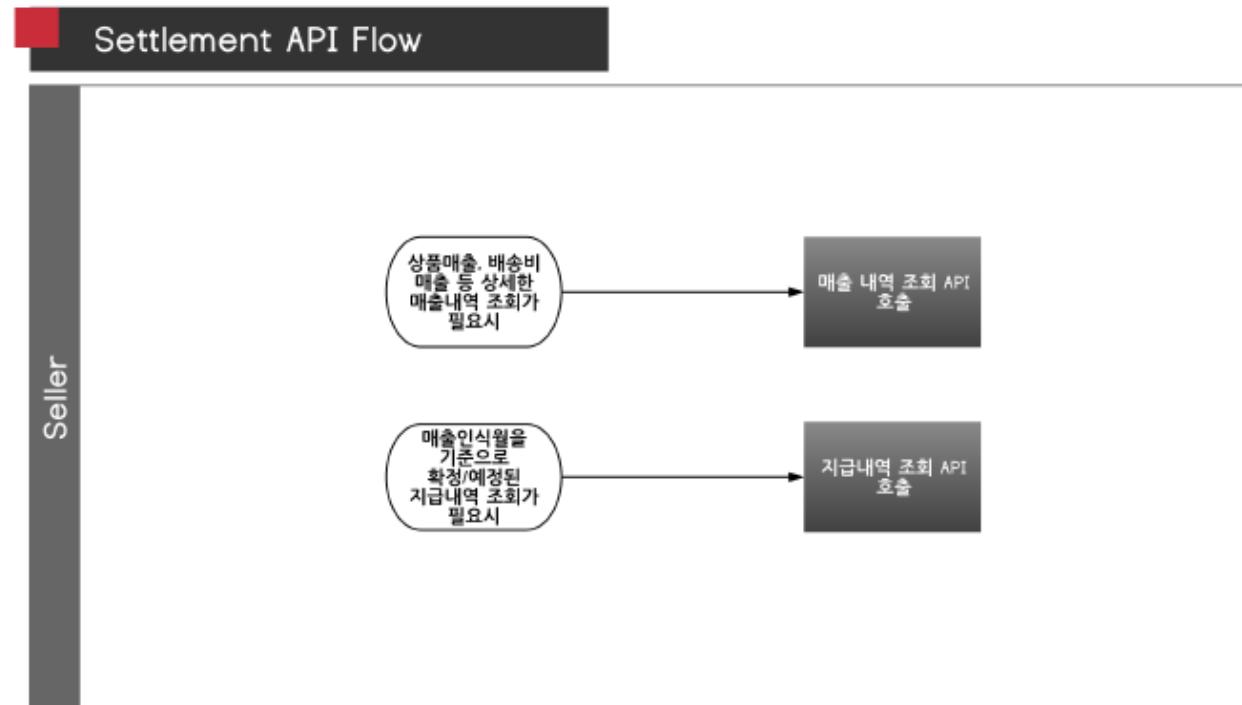
쿠폰 API 워크플로우

▶ 쿠팡 Workflow



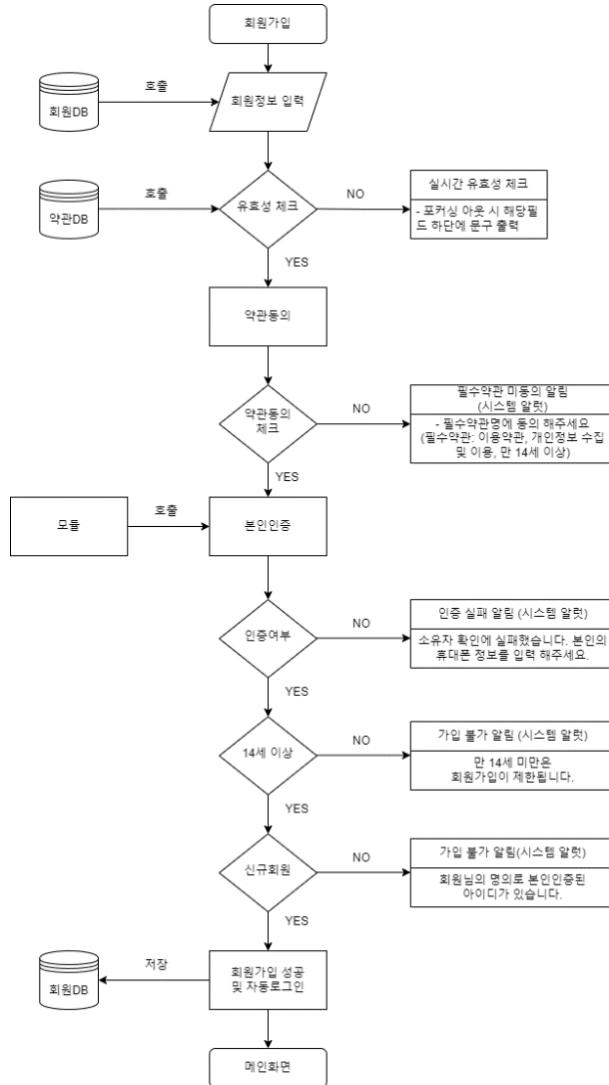
CS API 워크플로우

▶ 쿠팡 Workflow

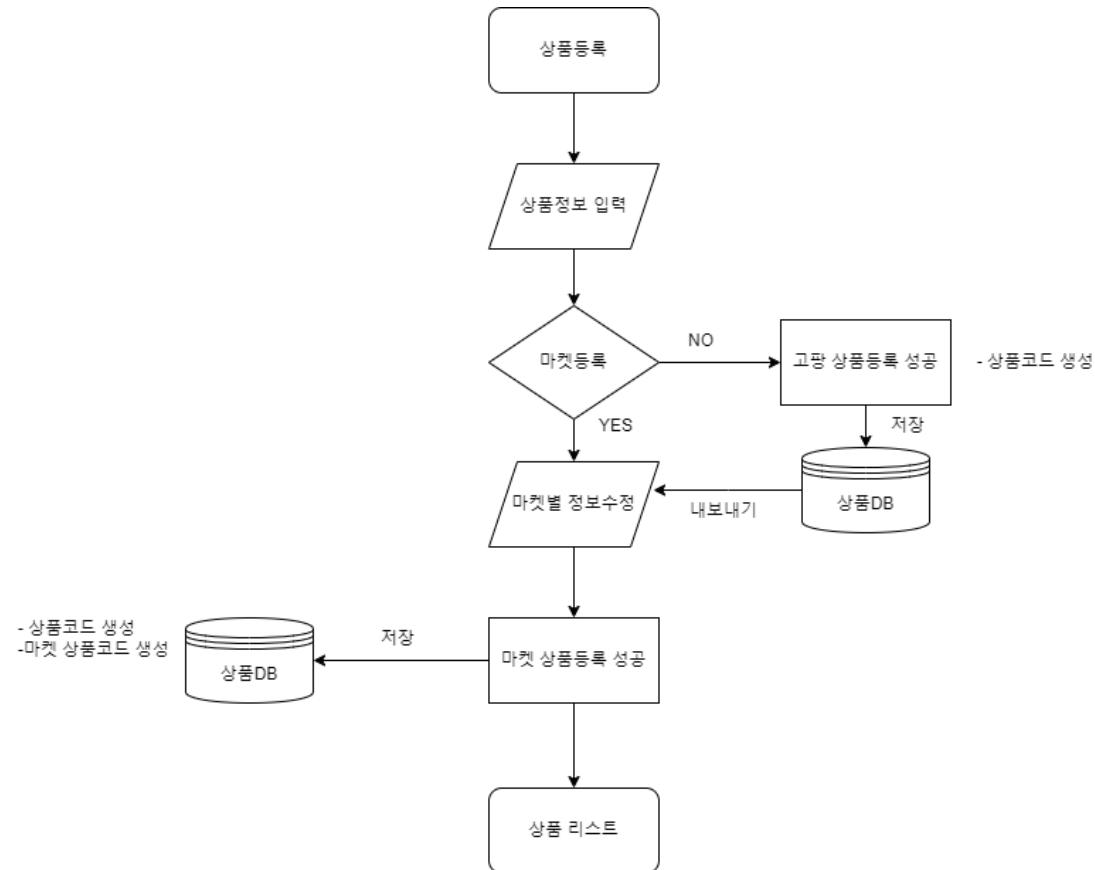


정산 API 워크플로우

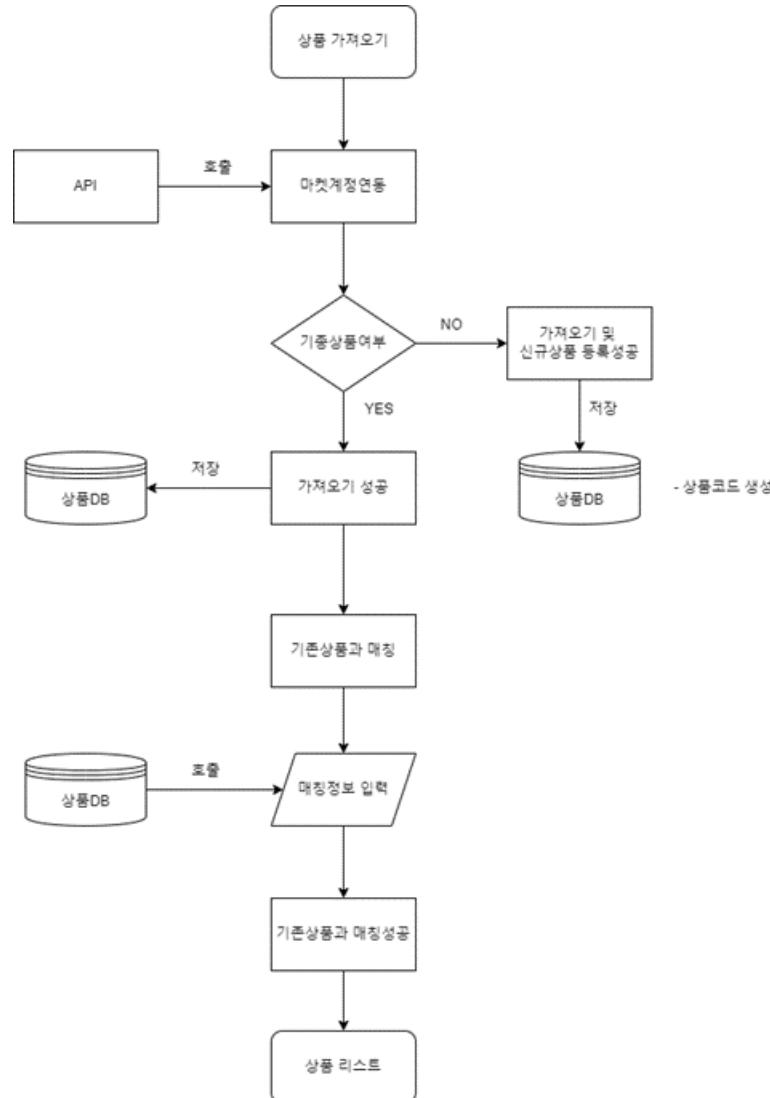
▶ Gopang Workflow(회원가입)



▶ Gopang Workflow(상품등록)

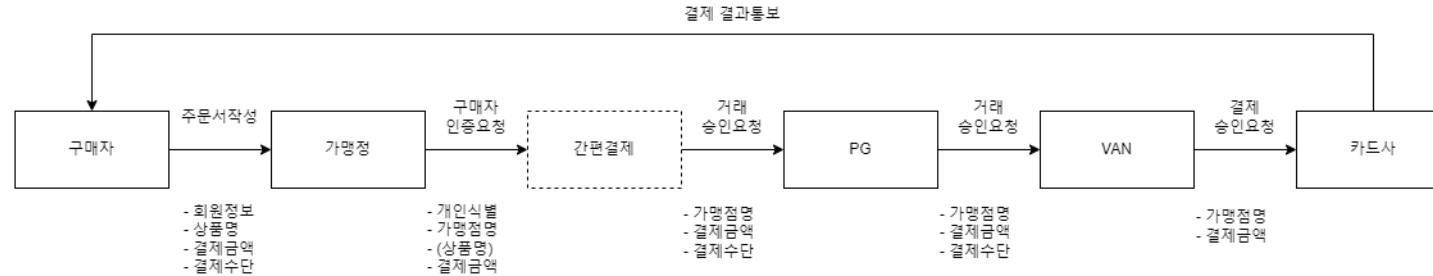


▶ Gopang Workflow
(상품 가져오기)

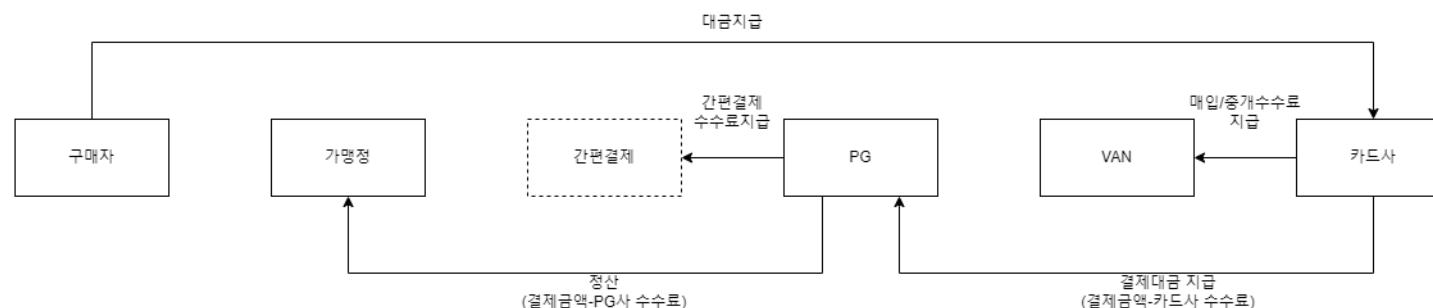


▶ Gopang Workflow(결제)

지급결제 사업자 간 결제 데이터 이동

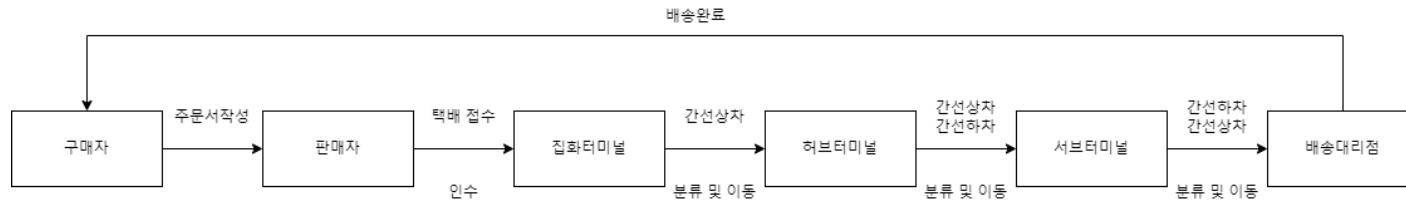


대표 가맹점 서비스(온라인결제)의 대금정산

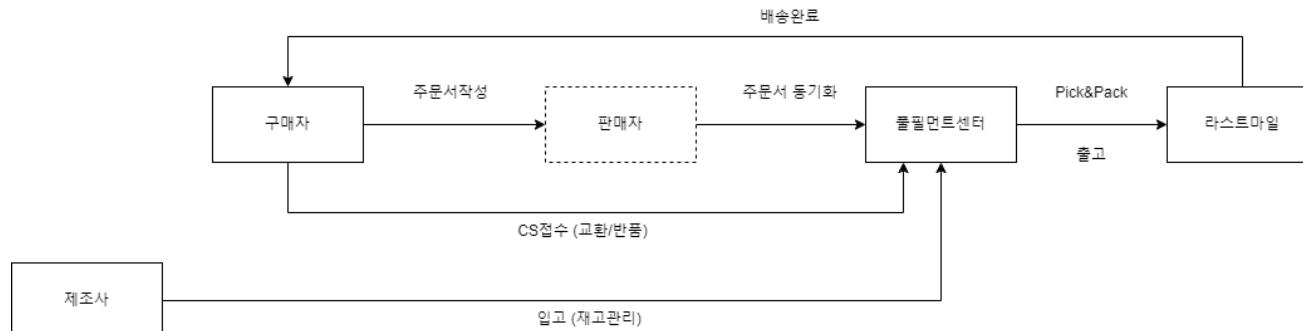


▶ Gopang Workflow(정산)

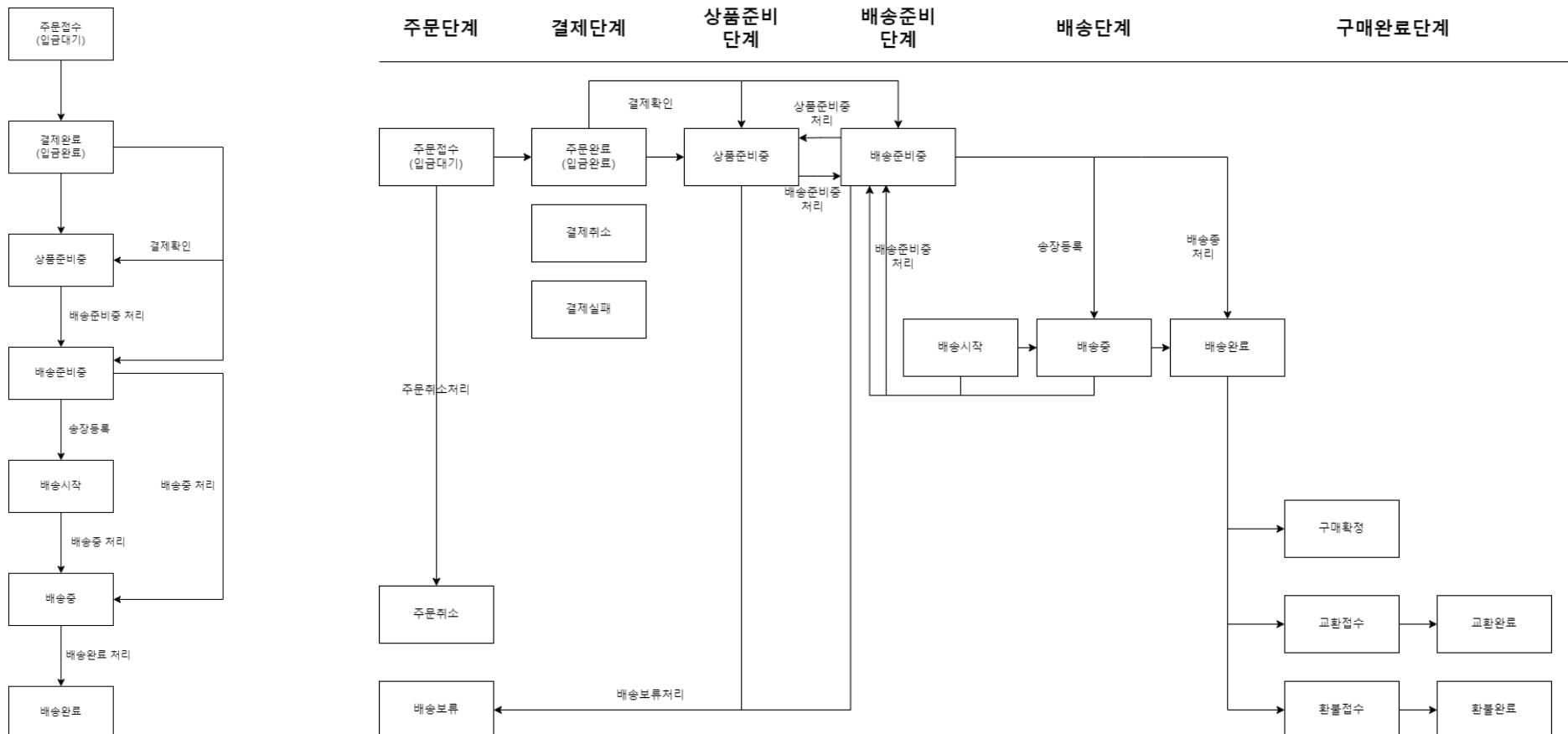
크로스도킹 방식



풀필먼트 방식



▶ Gopang Workflow(주문, 배송)



▶ Gopang API (회원, 상품)

사용자

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	POST	user	api/v1/user/join	사용자 회원가입			
2	POST	user	api/v1/user/login	사용자 로그인			
3	PUT	user	api/v1/user/logout	사용자 로그아웃			
4	PUT	user	api/v1/user/usermodify	사용자 정보 수정			
5	DELETE	user	api/v1/user/secession	사용자 탈퇴			

판매자

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	POST	seller	api/v1/seller/join	판매자 회원가입			
2	POST	seller	api/v1/seller/login	판매자 로그인			
3	PUT	seller	api/v1/seller/logout	판매자 로그아웃			
4	PUT	seller	api/v1/seller/sellermodify	판매자 정보 수정			
5	DELETE	seller	api/v1/seller/secession	판매자 탈퇴			

약관

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	GET	admin	api/v1/policies/terms	고팡 이용약관			
2	GET	admin	api/v1/policies/product	상품평 및 상품문의 운영원칙			
3	GET	admin	api/v1/policies/youth	청소년보호정책			
4	GET	admin	api/v1/policies/seller	판매이용약관			
5	GET	admin	api/v1/policies/service	쿠팡서비스이용정책			
6	GET	admin	api/v1/policies/vdp	취약점공개정책			

▶ Gopang API (판매자, 카테고리)

판매자							
IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	POST	seller	/api/v1/seller/item	판매자가 상품을 등록한다			
2	PUT	seller	/api/v1/seller/item/{itemId}	판매자가 상품을 수정한다			
3	DELETE	seller	/api/v1/seller/item/{itemId}	판매자가 상품을 삭제한다			
4	GET	seller	/api/v1/seller/items/{itemId}	판매자가 상품 상세를 조회한다			
5	GET	seller	/api/v1/seller/items	판매자가 상품 목록을 조회한다			

사용자							
IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	GET	all	/api/v1/main/items	비회원이 메인페이지 데이터를 요청한다			
2	GET	user	/api/v1/main/items	회원이 메인페이지 데이터를 요청한다			
3	GET	user	/api/v1/items/{itemId}	회원이 상품상세를 조회한다			
4	GET	user	/api/v1/items?category={string}&sort={string}	회원이 상품목록을 조회한다			

카테고리							
IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	GET	all	/api/v1/categories	전체 카테고리를 조회한다			
2	POST	admin	/api/v1/category	관리자가 카테고리를 등록한다	{ "name": , "parentId": , "depth": }		
3	POST	admin	/api/v1/category/{categoryId}	관리자가 카테고리를 수정한다	{ "name": , "parentId": , "depth": }		
4	DELETE	admin	/api/v1/category/{categoryId}	관리자가 카테고리를 삭제한다			

▶ Gopang API (주문, 주문내역, 교환, 반품)

주문

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	POST	user	/api/v1/order/	회원이 주문을 요청한다	회원정보, 상품정보, 배송정보, 결제정보		
2	DELETE	user	/api/v1/order/{order_id}	회원이 주문을 취소한다	주문번호		

주문내역

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	DELETE	user	/api/v1/history/{history_id}	회원이 주문내역을 삭제한다	주문내역번호		
2	GET	user	/api/v1/history/{user_id}/	회원이 주문 상세를 조회한다	주문내역번호		
3	GET	user	/api/v1/history/{history_id}/receipt/	회원이 영수증을 조회한다	주문내역번호		
4	GET	user	/api/v1/history/{user_id}	회원이 전체 주문내역을 조회한다	주문내역번호		

교환

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	POST	user	/api/v1/exchange/{order_id}	회원이 교환을 요청한다	교환할 상품번호		
2	DELETE	user	/api/v1/exchange/{change_id}	회원이 교환을 취소한다	교환 번호		
3	GET	user	/api/v1/exchange/	회원이 교환내역을 조회한다			
4	GET	user	/api/v1/exchange/{change_id}	회원이 교환 상세내역을 조회한다	교환 번호		

반품

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	GET	user	/api/v1/return/	회원이 반품내역을 조회한다			
2	GET	user	/api/v1/return/{return_id}	회원이 반품상세내역을 조회한다			
3	DELETE	user	/api/v1/return/{return_id}	회원이 반품을 취소한다			
4	POST	user	/api/v1/return/{order_id}	회원이 반품을 요청한다	반품 상품번호		

▶ Gopang API (결제, 포트원)

결제								
IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)	
1	POST	user	/api/v1/payment/pay	결제실행	주문번호, 결제금액, 상태			
2	GET	user	/api/v1/payment/detail	결제내역 불러오기	주문번호, 환불금액, 결제금액, 카드번호, 상태			
3	POST	user	/api/v1/payment/cancel	결제취소	주문 번호, 결제금액, 환불금액, 상태			

포트원 API								
IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)	
1	POST	user	/users/getToken	Access Token 발급	{ "code": 0, "message": "string", "response": { "access_token": "string", "expired_at": 0, "now": 0 } }			
2	GET	user	/payments/{imp_uid}	결제 단건 조회				
3	GET	user	/payments	결제 복수 조회				
4	POST	user	/payments/cancel	결제취소				
5	GET	user	/payments/status/{payment_status}	결제상태기준 복수조회				

▶ Gopang API (장바구니, 리뷰)

장바구니

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	GET	user	/api/v1/cart	회원이 장바구니 목록을 조회한다	식별자(토큰)		
2	POST	user	/api/v1/cart/items	회원이 상품을 장바구니에 추가한다	식별자(토큰), 상품정보, 수량		
3	DELETE	user	/api/v1/cart/items/{item_id}	회원이 장바구니에 특정 상품을 삭제한다	식별자(토큰), 장바구니 고유번호		
4	DELETE	user	/api/v1/cart/items	회원이 장바구니 전체삭제를 한다	식별자(토큰)		
5	PUT	user	/api/v1/cart/items/{item_id}	구매자가 장바구니에 상품수량을 변경한다	식별자(토큰), 장바구니 고유번호		

리뷰

IDX	Method	Purview	URI	Action	request(param)	response(성공)	response(실패)
1	GET	user	/api/v1/reviews	회원이 리뷰 목록을 조회한다 (마이 고팡페이지)			
2	GET	user	/api/v1/reviews/item	회원이 상품목록을 조회한다(상품 상세페이지)			
3	POST	user	/api/v1/reviews/{reviewsId}	회원이 리뷰를 등록한다			
4	POST	user	/api/v1/reviews/{reviewsId}	회원이 리뷰를 작성한다			
5	PUT	user	/api/v1/reviews/{reviewsId}	회원이 리뷰를 수정한다			
6	DELETE	user	/api/v1/reviews/{reviewsId}	회원이 리뷰를 삭제한다			

기획단계

 ERD 명세서 (회원, 판매자)

데이터 베이스명	Maria DB	테이블 명	회원
요구사항 ID		테이블 설명	회원가입 및 로그인 대상관리
칼럼명	칼럼ID	타입 및 길이	Not Null
회원번호	user_id	INT	Y
이메일	user_email	VARCHAR(30)	Y
비밀번호	user_password	VARCHAR(20)	Y
이름	user_name	VARCHAR(30)	Y
전화번호	user_phonenumber	VARCHAR(20)	Y
우편번호	user_zipcode	VARCHAR(5)	Y
주소	user_address	VARCHAR(50)	Y
상세주소	user_detail_address	VARCHAR(50)	Y
데이터 베이스명	Maria DB	테이블 명	판매자
요구사항 ID		테이블 설명	판매자 가입 및 로그인 대상관리
칼럼명	칼럼ID	타입 및 길이	Not Null
판매자번호	seller_id	INT	Y
이메일	seller_email	VARCHAR(30)	Y
비밀번호	password	VARCHAR(20)	Y
이름	name	VARCHAR(30)	Y
전화번호	phone_number	VARCHAR(20)	Y
비즈니스형태	business_type	ENUM	N

▶ ERD 명세서 (약관동의이력, 약관)

데이터 베이스명	Maria DB	테이블 명	약관 동의 이력						
요구사항 ID		테이블 설명	회원과 판매자의 약관내용 동의여부확인						
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX		기본값	제약조건
약관코드	tos_code	VARCHAR(2)	Y	Y					
회원번호	user_id	INT	Y		Y		식별자		AUTO_INCREMENT 사용
판매자번호	seller_id	INT	Y		Y				AUTO_INCREMENT 사용
동의여부	agree_whether	VARCHAR(1)	Y						
동의일자	agree_date	DATETIME	Y						

데이터 베이스명	Maria DB	테이블 명	약관						
요구사항 ID		테이블 설명	회원과 판매자에게 플랫폼의 약관내용 안내						
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX		기본값	제약조건
약관코드	tos_code	VARCHAR(2)	Y		Y				
약관명	tos_name	VARCHAR(20)	N						
약관내용	tos_content	VARCHAR(1000)	Y						
필수여부	tos_essential	VARCHAR(1)	Y						Y/N

기획단계

▶ ERD 명세서 (장바구니(회원), 장바구니(비회원), 장바구니 상품)

데이터 베이스명	Maria DB	테이블 명	장바구니(회원)					
요구사항 ID		테이블 설명	회원은 자신이 고른 상품을 볼 수 있다					
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
장바구니ID	cart_id	INT	Y	Y				
회원번호	member_id	INT	Y		Y			AUTO_INCREMENT 사용
이메일	member_email	VARCHAR(30)	Y		Y			이메일 형식에 맞게 작성
데이터 베이스명	Maria DB	테이블 명	장바구니(비회원)					
요구사항 ID		테이블 설명	비회원은 자신이 고른 상품을 볼 수 있다					
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
비회원장바구니ID	none_cart_id	INT	Y	Y				
이메일	token	VARCHAR(255)	Y					
데이터 베이스명	Maria DB	테이블 명	장바구니 상품					
요구사항 ID		테이블 설명	(비)회원이 고른 상품을 한눈에 볼 수 있게하는 목록					
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
장바구니 상품 번호	cartitem_id	INT	Y	Y				
장바구니ID	cart_id	INT	Y		Y			
장바구니 상품 수량	cartitem_count	INT	Y					
비회원장바구니ID	none_cart_id	INT	Y		Y			

▶ ERD 명세서(판매자 배송정보)

데이터 베이스명	Maria DB			테이블 명	판매자 배송 정보				
요구사항 ID				테이블 설명	판매자가 (비)회원에게 상품배송을 등록하는 목록				
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX		기본값	제약조건
판매자 배송정보ID	seller_deilvery_id	INT	Y	Y			판매자 배송정보 PK		
출고지	forwarding_place	VARCHAR(255)	N						
제주/도서산간	delivery_jeju_mountain	TINYINT	N						제주 도서산간 배송여부
택배사	courier_company	VARCHAR(255)	N						
배송방법	delivery_way	VARCHAR(255)	N						[일반배송: 신선배송:주문제작: 구매대행:설치배송 또는 판매자 직접전달]
배송비	delivery_cost	INT	N						
배송기간	delivery_period	VARCHAR(255)	N						단위: 일
이메일	seller_eamil	VARCHAR(30)	Y		Y		로그인시 아이디		

▶ ERD 명세서(상품 리뷰정보, 리뷰 이미지 파일)

데이터 베이스명	Maria DB	테이블 명	상품 리뷰 정보					
요구사항 ID		테이블 설명	사용자가 상품을 배송받고 이 상품에 대한 평가를 한다					
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
리뷰번호	review_id	INT	Y	Y			리뷰식별자	
제목	title	VARCHAR(100)	Y					
내용	content	TEXT	N					
별점	star	INT	Y					
좋아요	liked		N					
생성일	created_date	DATE	Y					
수정일	modified_date	DATE	Y					
상품ID	item_id	INT	Y		Y		상품기본정보PK	
회원번호	member_id	INT	Y		Y		식별자	AUTO_INCREMENT 사용
이메일	momber_email	VARCHAR(30)	Y		Y		로그인시 아이디	이메일 형식에 맞게 작성
데이터 베이스명	Maria DB	테이블 명	리뷰 이미지 파일					
요구사항 ID		테이블 설명	사용자가 상품을 배송받고 이 상품에 대한 이미지를 올린다					
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
이미지파일ID	review_image_file_id	INT	Y	Y				AUTO_INCREMENT 사용
이미지 파일 용도	image_type	VARCHAR(255)	Y					{ main : sub }
이미지 파일 원본 이름	original_image_name	VARCHAR(255)	Y					
이미지 파일 저장 이름	save_image_name	VARCHAR(255)	Y					중복을 막기 위해 랜덤으로 생성
이미지 파일 크기	image_size	VARCHAR(255)	Y					
S3에서관리할 key	image_key	VARCHAR(255)	Y					AWS S3에서 관리할 KEY
S3에 저장된 경로	image_path	VARCHAR(255)	Y					AWS S3 저장소의 파일에 접근하는 URL
리뷰번호	review_id	INT	Y		Y		리뷰식별자	이메일 형식에 맞게 작성

▶ ERD 명세서(상품, 상품 상세정보)

데이터 베이스명	Maria DB	테이블 명	상품					
요구사항 ID		테이블 설명	판매자가 판매 할 물품					
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
상품ID	item_id	INT	Y	Y			상품기본정보	
노출상품명	title_name	VARCHAR(100)	Y					고객이 보는 상품명
등록상품명	seller_name	VARCHAR(100)	Y					판매자 관리용 상품명
상품번호	number	VARCHAR(30)	Y					상품을 구분하기 위한 20자리 번호(우량이 부여)
판매상태	state	TINYINT	Y					
최초 등록일	create_date	DATE	Y					
최종 수정일	modified_date	DATE	Y					
판매 시작일	seller_start_date	DATE	Y					
판매 종료일	seller_end_date	DATE	Y					
판매자 번호	seller_id	INT	Y					AUTO_INCREMENT 사용
이메일	seller_email	VARCHAR(30)	Y		Y		로그인시 아이디	

데이터 베이스명	Maria DB	테이블 명	상품 상세 정보					
요구사항 ID		테이블 설명	판매자가 판매 할 물품 상세					
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
상품ID	item_detail_id	INT	N	Y			상품기본정보	
상세설명 파일명	original_file_name	VARCHAR(255)	N					HTML,PDF로 저장
상세설명 파일저장이름	save_file_name	VARCHAR(255)	N					중복방지를 위해 맨뒤생성
상세설명 파일크기	file_size	VARCHAR(255)	N					
S3에서 관리할 key	file_key	VARCHAR(255)	N					
S3에 저장된 경로	file_path	VARCHAR(255)	N					
병행수입여부	parallel_import	TINYINT	N				0:병행수입, 1:병행수입아님	
미성년자 구매여부	minor	TINYINT	N				0:가능, 1:불가능	
최대 구매수량	max_order	VARCHAR(255)	N				1인당 동일물품, 최대구매수량	
부가세 여부	tax	TINYINT	N				0:과세, 1:면세	
소재	material	VARCHAR(255)	N					
색상	color	VARCHAR(255)	N					
지수	size	VARCHAR(255)	N					
제조사(수입자)	maker	VARCHAR(255)	N				제조사가 부여한 이름	
제품명	manufacturer_item_name	VARCHAR(255)	N					
취급시 주의사항	warning	VARCHAR(255)	N					
품질보증기증	quality-assurance	VARCHAR(255)	N					
AS	as_info	VARCHAR(255)	Y		Y		로그인시 아이디	
상품ID	item_id	INT	Y		Y			

▶ ERD 명세서(반품/교환정보, 카테고리, 카테고리 타입)

데이터 베이스명	Maria DB	테이블 명	반품/교환정보				
요구사항 ID		테이블 설명	사용자가 반품/ 교환을 한 상품을 확인한다				

컬럼명	컬럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
반품/교환 ID	return_exchange_id	INT	Y	Y			반품/교환PK	반품/교환정보식별자
반품/교환지	address	VARCHAR(255)	N					반품/교환주소
반품배송비	delivery_cost	INT	N					
상품ID	item_id	INT	Y		Y		상품기본정보PK	

데이터 베이스명	Maria DB	테이블 명	카테고리				
요구사항 ID		테이블 설명	상품 분류				

컬럼명	컬럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
카테고리ID	category_id	INT	Y	Y			카테고리PK	
카테고리명	name	VARCHAR(255)	Y					
깊이	depth	VARCHAR(255)	Y					운영을 위한 값, 카테고리를 좀 더 쉽게 찾기위함
부모카테고리ID	parent_id	INT	Y		Y		부모카테고리ID PK	

데이터 베이스명	Maria DB	테이블 명	카테고리 타입				
요구사항 ID		테이블 설명	상품 분류 타입				

컬럼명	컬럼ID	타입 및 길이	Not Null	PK	FK	IDX	기본값	제약조건
카테고리ID	category_id	INT	Y		Y		카테고리PK	
상품ID	item_id	INT	Y		Y		상품기본정보	

▶ ERD 명세서(배송)

데이터 베이스명	Maria DB			테이블 명	배송				
요구사항 ID				테이블 설명	사용자가 물품을 구매하여 판매자가 물건을 보내준다				
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX		기본값	제약조건
배송내역 번호	delivery_id	INT	Y	Y					AUTO_INCREMENT 사용
주문내역 번호	history_id	INT	Y		Y				AUTO_INCREMENT 사용
회원번호	user_id	INT	Y						
배송 상태	state	ENUM	Y						
받는 사람 이름	reciever_name	VARCHAR	Y						
받는 사람 연락처	reciever_phone	VARCHAR	Y						
받는 사람 주소	reciever_addr	VARCHAR	Y						
배송 요청 사항	reciever_message	VARCHAR	N						
배송 요청 시간	time	DATETIME	Y		Y				
주문번호	order_id	INT	Y						AUTO_INCREMENT 사용

▶ ERD 명세서(주문, 주문내역)

데이터 베이스명	Maria DB		테이블 명	주문				
요구사항 ID			테이블 설명	사용자가 장바구니속 상품을 주문한다				
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX		제약조건
주문번호	order_id	INT	Y	Y				AUTO_INCREMENT 사용
회원번호	user_id	INT	Y					
장바구니 상품번호	cartitem_id	INT	Y					
총 주문 금액	amount	INT	Y					결제요청금액
받는 사람 이름	reciever_name	VARCHAR	Y					
받는 사람 연락처	reciever_phone	VARCHAR	Y					
받는 사람 주소	reciever_addr	VARCHAR	Y					
주문시간	time	DATETIME	Y					
장바구니ID	cart_id	INT	Y		Y			
데이터 베이스명	Maria DB		테이블 명	주문내역				
요구사항 ID			테이블 설명	사용자가 장바구니속 상품을 주문한 상세내역을 조회한다				
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX		제약조건
주문내역번호	history_id	INT	Y	Y			식별자	AUTO_INCREMENT 사용
회원번호	user_id	INT	Y					
상품번호	item_id	INT	Y					
총주문금액	order_amount	INT	Y					
주문상태	order_state	ENUM	Y				상품준비중	주문환호', '주문취소', '결제완료', '결제취소', '상품준비중', '배송준비중', '배송완료', '배송종', '판매자 취소'
받는사람이름	reciever_name	VARCHAR	Y					
받는사람연락처	reciever_phone	VARCHAR	Y					
받는사람주소	reciever_addr	VARCHAR	Y					
배송요정사항	reciever_message	VARCHAR	N					
주문시간	time	DATETIME	Y					
주문완료시간	order_time	DATETIME	Y					
주문번호	order_id	INT	Y		Y		식별자	AUTO_INCREMENT 사용
결제번호	payment_key	INT	Y		Y		결제정보식별자	

▶ ERD 명세서(교환내역, 반품/취소내역)

데이터 베이스명	Maria DB	테이블 명	교환내역						
요구사항 ID		테이블 설명	사용자가 상품을 교환한 내역을 확인한다						
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX		기본값	제약조건
교환내역 번호	change_id	INT	Y	Y			식별자		AUTO_INCREMENT 사용
장바구니 상품 번호	cartitem_id	INT	Y						
결제번호	payment_id	INT	Y						
총 주문금액	order_amount	INT	Y						결제 요청금액
받는사람 연락처	reciever_phone	VARCHAR	Y						
받는사람 주소	reciever_addr	VARCHAR	Y						
배송요청 사항	reciever_message	VARCHAR	N						
교환 연락처	phone	VARCHAR	N						교환과정에서 배송 정보 변경시 기입
교환 주소	addr	VARCHAR	N						교환과정에서 배송 정보 변경시 기입
교환 이름	name	VARCHAR	N						교환과정에서 배송 정보 변경시 기입
교환요청시간	time	DATETIME	Y						
교환완료시간	change_time	DATETIME	Y						
주문내역번호	history_id	INT	Y		Y		식별자		AUTO_INCREMENT 사용
배송내역번호	delivery_id	INT	Y		Y		식별자		AUTO_INCREMENT 사용

데이터 베이스명	Maria DB	테이블 명	반품/취소내역						
요구사항 ID		테이블 설명	사용자가 상품을 반품, 취소한 내역을 확인한다						
칼럼명	칼럼ID	타입 및 길이	Not Null	PK	FK	IDX		기본값	제약조건
반품/취소내역번호	cancel_id	INT	Y	Y			식별자		AUTO_INCREMENT 사용
장바구니 상품번호	cartitem_id	INT	Y						
결제번호	payment_id	INT	Y						
취소금액	amout	INT	N						반품/취소 요청된 상품들의 총 금액
총주문금액	order_amount	INT	Y						결제 요청금액
반품사유	comment	VARCHAR	Y						
취소요청시간	time	DATETIME	Y						
취소완료시간	cancel_time	DATETIME	Y						
주문내역번호	history_id	INT	Y		Y		식별자		AUTO_INCREMENT 사용
배송내역번호	delivery_id	INT	Y		Y		식별자		AUTO_INCREMENT 사용

기획단계

▶ ERD 명세서(결제, 결제취소, 카드정보)

데이터베이스명	Maria DB	테이블명	결제
요구사항 ID		테이블 설명	사용자가 장바구니에 담은 상품을 결제한다
칼럼명	칼럼ID	타입 및 길이	Not Null
결제번호	payment_key	INT	Y
주문번호	merchant_uid	VARCHAR	N
회원번호	user_id	VARCHAR	Y
결제요청금액	amount	INT	Y
결제상태	status	VARCHAR	N
결제시간	time	DATETIME	N
주문번호	order_id	INT	Y
			Y
			주문번호 식별자
			AUTO_INCREMENT 사용
데이터베이스명	Maria DB	테이블명	결제취소
요구사항 ID		테이블 설명	사용자가 결제를 취소한다
칼럼명	칼럼ID	타입 및 길이	Not Null
결제취소번호	payment_cancel_id	INT	Y
주문번호	merchant_uid	VARCHAR	N
취소요청금액	cancel_amount	INT	Y
총금액	amount	INT	Y
결제상태	status	VARCHAR	Y
취소시간	time	DATETIME	N
주문번호	order_id	INT	Y
			Y
			주문번호 식별자
			AUTO_INCREMENT 사용
데이터베이스명	Maria DB	테이블명	카드정보
요구사항 ID		테이블 설명	사용자가 결제수단인 카드를 등록한다
칼럼명	칼럼ID	타입 및 길이	Not Null
주문번호	merchant_uid	VARCHAR	N
카드번호	card_number	VARCHAR	N
카드만료일	expiry	VARCHAR	N
카드소지자생년월일	birth	VARCHAR	N
카드비밀번호2자리	pwd_2digit	VARCHAR	N
카드인증번호	cvc	VARCHAR	N

Payment

▶ PORTONE의 장점

1. 쉬운 개발 연동

- 포트원의 API 만으로 여러가지 PG사와 결제수단을 쉽게 도입 할 수 있습니다.

2. 운영 편의성

- 포트원 관리자콘솔이 제공됩니다.
- 결제내역관리(조회, 환불), 연동설정, 관리자계정 추가 등을 하나의 페이지에서 관리 할 수 있습니다.

3. 통합 관리

- 여러 PG사의 계약을 위한 신청은 포트원 관리자콘솔 한 곳에서 한꺼번에 진행할 수 있습니다.

포트원은 복잡한 PG결제 연동규격을 누구나 쉽게 연동할 수 있도록 REST API를 지원합니다.

▶ PORTONE을 이용하는 기업

크림(Kream)

번개장터

오늘의집

젠틀몬스터

29CM(무신사)

패스트캠퍼스

노스페이스

펫프렌즈

뉴퍼마켓(구. AJ전시몰)

랜덤투유

원더월(wonderwall)

월급쟁이부자들

여신티켓

일반결제 정기결제 카카오페이지

일반결제 카카오페이지 토스페이 네이버페이

일반결제 카카오페이지 네이버페이 페이코

일반결제 카카오페이지 네이버페이 페이코

일반결제 정기결제 카카오페이지 네이버페이 페이코

일반결제 네이버페이

일반결제 정기결제 카카오페이지 네이버페이

일반결제 카카오페이지 네이버페이 페이코

카카오페이지 네이버페이 페이코 일반결제

일반결제 카카오페이지 네이버페이 해외결제

일반결제 카카오페이지 토스페이 네이버페이

일반결제 정기결제

혼다 코리아

월스아웃

화해

오픈갤러리

숨고

호갱노노

자비스

오누이

밀리의 서재

헤이딜러

마켓디자이너스(튜터링)

리디 주식회사(라프텔)

일반결제

일반결제 카카오페이지 네이버페이 해외결제

일반결제 카카오페이지 네이버페이 페이코

일반결제 정기결제 네이버페이

일반결제 정기결제 네이버페이

정기결제

일반결제 정기결제

일반결제 정기결제

일반결제 정기결제 카카오페이지 네이버페이

일반결제

일반결제 정기결제

문화상품권 카카오페이지 해외결제

▶ 1. PORTONE 회원가입 하기

간편하게 **로그인하고**
내 상점을 확인해보세요



이메일 주소(ID)를 입력해주세요

비밀번호를 입력해주세요

비밀번호 재설정 | 가입하기

로그인

결제 그 이상의 경험,
포트원 가입하기



결제대행사 가입비용 100% 지원 이벤트 진행중

수신 가능한 이메일 주소(ID)를 입력해 주세요

비밀번호를 입력해 주세요

비밀번호를 확인해 주세요

포트원 필수/선택 약관에 모두 동의합니다 ▲

[필수] 포트원 이용약관에 동의

[필수] 개인정보 수집 및 이용에 동의

[선택] 마케팅 활용 및 광고성 정보 수신에 동의

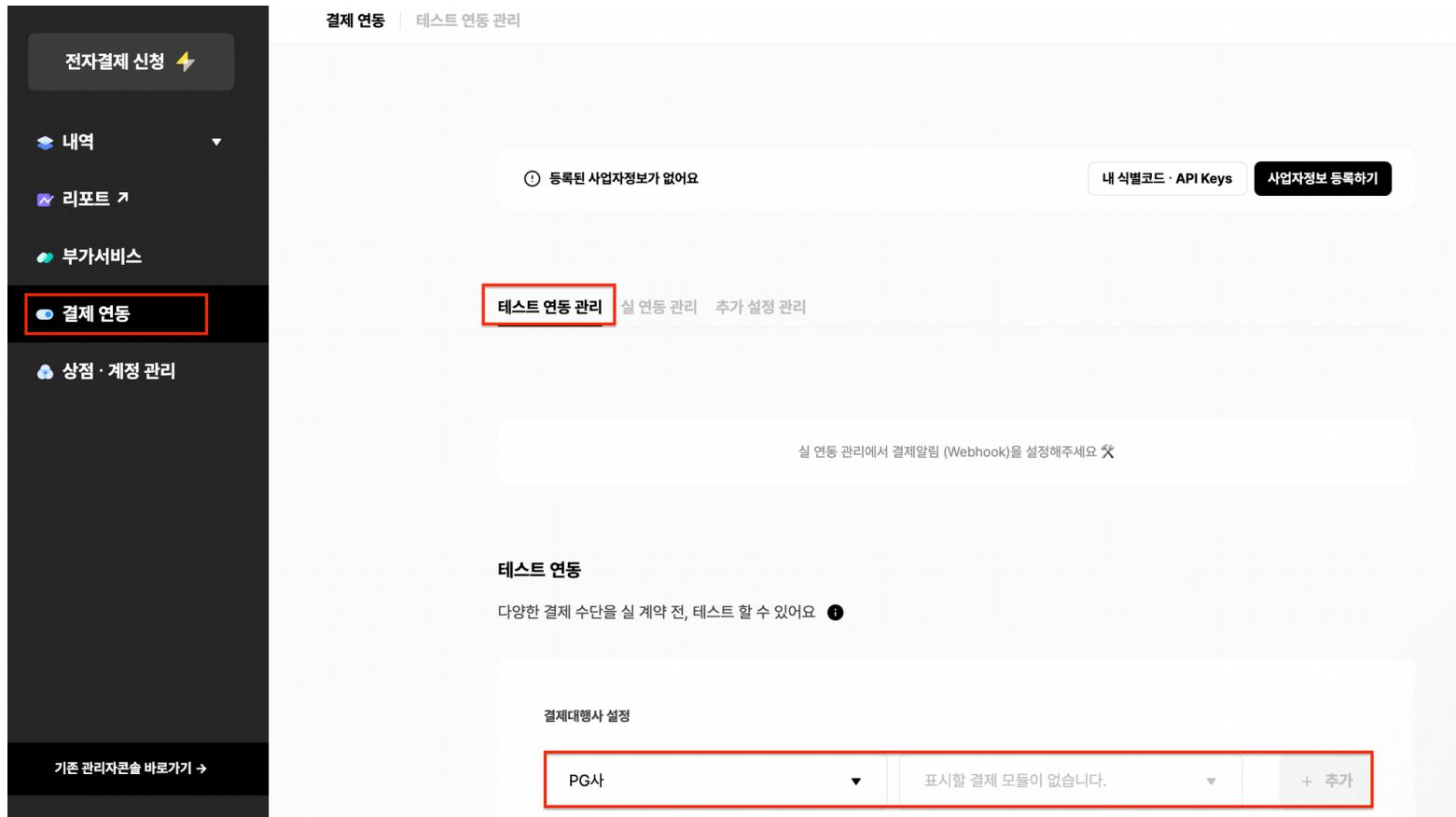
1:1 채팅문의 | 로그인하기

가입

연동을 진행하기 위해서는 포트원 관리자페이지 회원가입이 선행되어야 합니다.

포트원 관리자 페이지는
별도 계약없이 무료 회원가입이 가능합니다.

2. PG 정보 설정하기



포트원 회원가입 이후 관리자 페
이지 내 결제 연동 -> 테스트 연
동 관리 탭에서 연동하고자 하
는 PG사를 선택합니다.

테스트 연동 관리 메뉴에서는 포
트원에서 지원하는 각 PG사별
테스트 설정을 진행할 수 있습니
다.

3. 연동정보확인 하기

내 식별코드 · API Keys

입력 칸을 클릭하시면 복사가 가능합니다

REST API Secret 재발급은 신중하게 진행해 주세요

가맹점 식별코드	<input type="text"/>
REST API Key	<input type="text"/> 02-77-4000-10-0000
REST API Secret	<p>ⓘ key 값이 유출되지 않도록 유의해 주세요</p> <pre>dh40237da9h009hhhe12chdfhe1897a8a47h u9u3a5eb</pre>

결제창 연동 및 API 호출 시 필요한 연동정보를 확인할 수 있습니다.

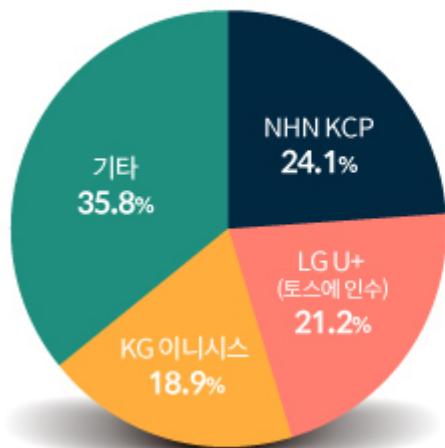
1. 포트원 관리자콘솔 로그인 -> 상점 · 계정 관리 -> 내 식별코드 · API Keys 선택
2. 가맹점 식별코드 / REST API Key / REST API Secret 확인

재발급

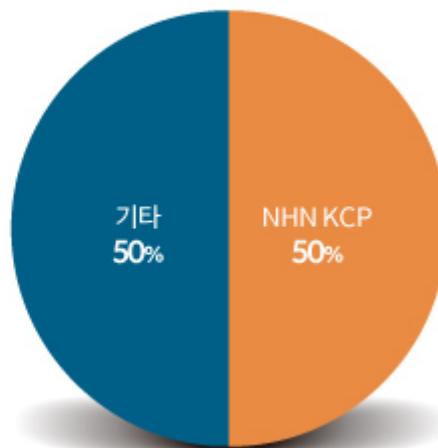
반드시 기존 연동 확인 후 진행해 주세요

▶ 국내 PG사 점유율

국내 PG 시장점유율



국내 OnlineVAN 시장점유율



NHN KCP 결제 테스트

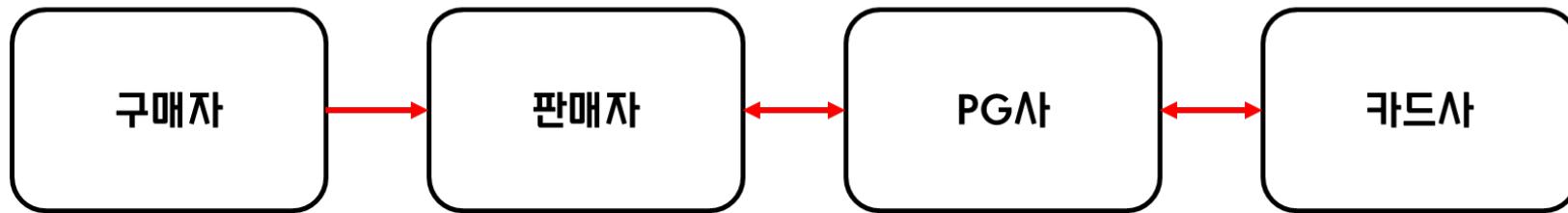
정보 수정 삭제

결제채널 이름	<u>NHN KCP Payco 허브형 결제창</u>
채널 키 (Channel Key)	<u>channel-key-</u> [REDACTED]
PG상점아이디 (사이트코드)	<u>A5</u> [REDACTED]
사이트키	<u>1Vzgj</u> [REDACTED]
배치결제그룹아이디	[REDACTED]

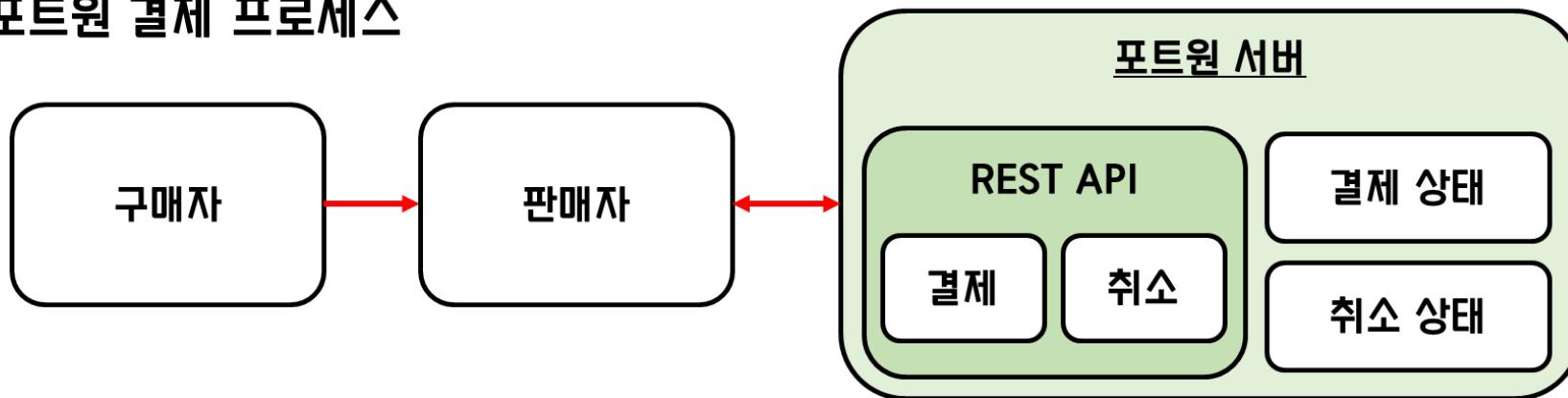
국내 시장 점유율 1위 PG사가 NHN KCP 이므로 결제대행사에 NHN KCP를 사용함.

▶ 결제 프로세스 비교

일반 결제 프로세스



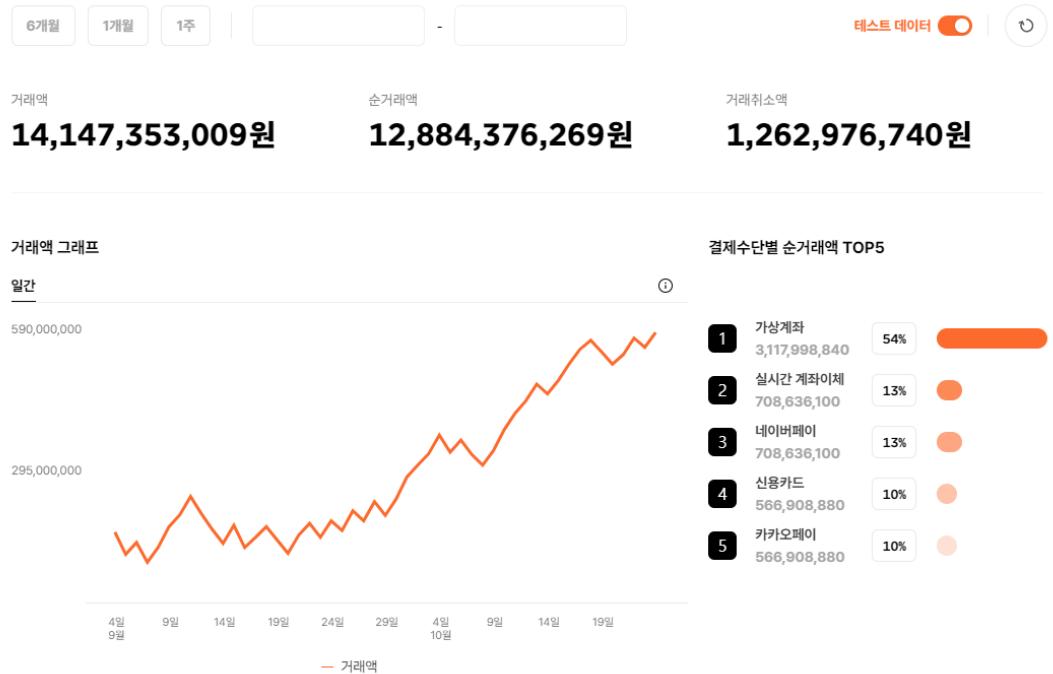
포트원 결제 프로세스



포트원과 PG 직접계약의 큰 차이점은
결제에 사용하는 **인증키와 PG사 연동**을 포트원에서 처리해줘 **결제 FLOW가 줄어든다**는 장점이
있다.

▶ 포트원 관리자 콘솔

대시보드



기준 시간: 상태 승인 시점 | 2023/05/25 00:00:00 ~ 2023/11/25 17:04:21 | 내역 검색 | 전체 | 검색 | 필터 | 순 거래액 | 역설 다운로드

전체 601 | 결제완료 300 | 결제예정 3 | 결제실패 26 | 결제취소 172 | 승인대기 0 | 상세 승인 시각 2023.11.22

결제상태	UID	MID	주문명	고객정보	결제수단	결제대행사	결제금액	구분	상세 승인 시각
결제완료	imps_625740308258	qwerty5829	포트원 거래번호	-	신용카드	NHN KCP	1,000 KRW	테스트 일반	22:29:00 취소
결제완료	imps_695044706557	507	주문명	-	신용카드	NHN KCP	100 KRW	테스트 일반	12:34:04 취소
결제완료	imps_723005278965	506		-	신용카드	NHN KCP	100 KRW	테스트 일반	12:33:25 취소
결제완료	imps_947001254071	505		-	신용카드	NHN KCP	100 KRW	테스트 일반	12:33:21 취소
결제완료	imps_845248396927	402		-	신용카드	NHN KCP	100 KRW	테스트 일반	12:04:08 취소
결제완료	imps_496040747350	qwerty58293		-	신용카드	NHN KCP	1,000 KRW	테스트 일반	11:44:00 취소
결제완료	imps_186830490244	qwerty58292		-	신용카드	NHN KCP	1,000 KRW	테스트 일반	11:40:30 취소
결제완료	imps_789626980882	qwerty58291		-	신용카드	NHN KCP	1,000 KRW	테스트 일반	11:37:07 취소
결제완료	imps_485612873014	qwerty58290		-	신용카드	NHN KCP	1,000 KRW	테스트 일반	11:36:52 취소
결제완료	imps_184158491294	qwerty58289		-	신용카드	NHN KCP	1,000 KRW	테스트 일반	11:12:38 취소

Access Token 발급요청

내 식별코드 · API Keys

입력 칸을 클릭하시면 복사가 가능합니다
REST API Secret 재발급은 신중하게 진행해 주세요

가맹점 식별코드	<input type="text"/>
REST API Key	<input type="text"/> 02777700000000000000000000000000
REST API Secret	<input type="text"/> ! key 값이 유출되지 않도록 유의해 주세요 dh40237da9b009bbbe12chdfbe1897a8a47h uuu3asaeo

재발급 반드시 기준 연동 확인 후 진행해 주세요

관리자 콘솔에서 확인한 REST API 키와 REST API Secret로
(POST) <https://api.iamport.kr/users/getToken> 를 호출하여
access token 발급을 요청할 수 있습니다.

API 호출 시 access_token을 Authorization 또는 X-ImpTokenHeader으로
HTTP header를 통해 전달합니다.

기준 NTP(Network Time Protocol)

포트원 REST API 서버는 Google Public NTP를 이용하여 기준시간과
동기화하고 있습니다.

Access Token 사용하기

발급받은 access token을 사용하여 포트원 REST API 요청을 할 수 있습니다.

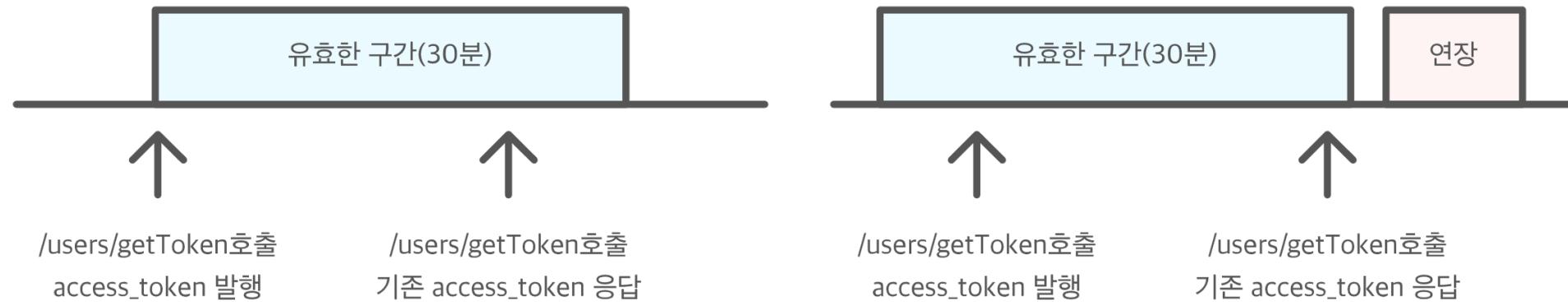
포트원 REST API는 **Bearer** 인증 방식을 사용하기 때문에 HTTP 요청 헤더에 access token을 다음과 같은 형식으로 포함합니다.

Authorization : Bearer a9ace025c90c0da2161075da6ddd3492a2fca776

Access Token 재발행과 재사용

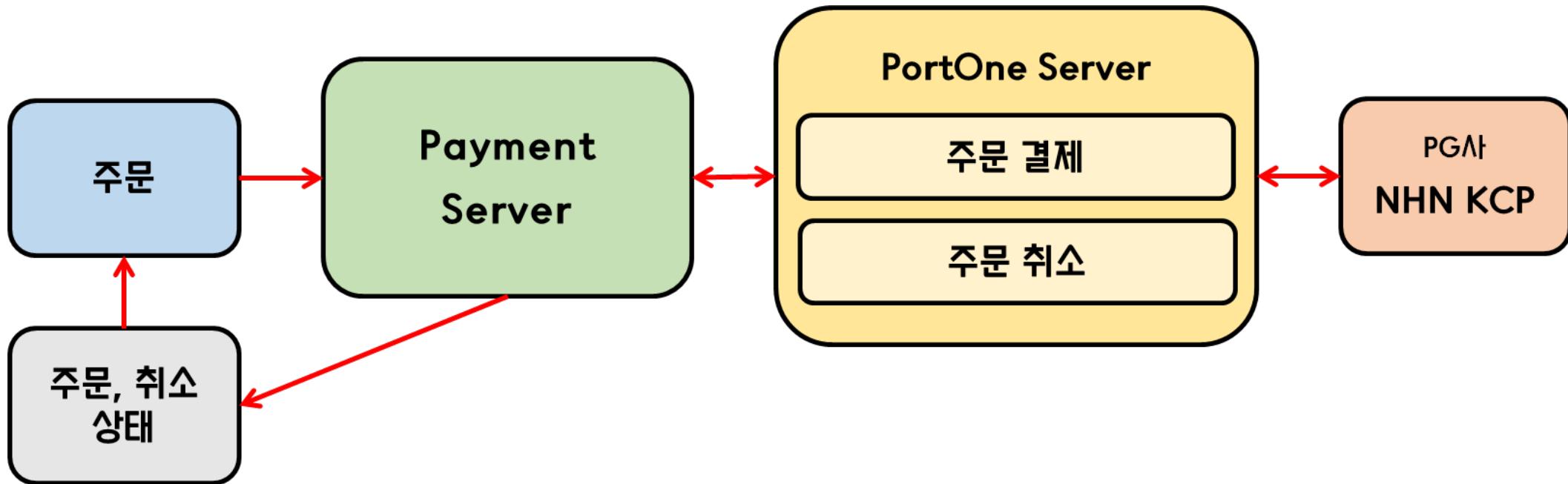
Access token의 만료기한은 발행 시간부터 30분입니다.

토큰은 만료기한이 지나면 사용할 수 없습니다. 만료된 토큰으로 API 요청을 하면 401 Unauthorized 응답을 받습니다.



- **토큰의 만료 전 발급 :** 기존 access token을 다시 발급한다. (만료기한은 기존 토큰과 동일, 단 기존 만료시간 전 1분이내 요청 시 5분 연장됨)
Access token의 재사용과 만료기한 5분 연장 동작방식은 한 가맹점에서 여러 대의 웹서버가 동시에 경쟁적으로 REST API(/users/getToken)를 호출하는 상황 한 가맹점에서 여러 대의 웹서버가 시간 동기화 되어있지 않은 상황을 고려해서 설계되었습니다.
- **토큰의 만료 후 발급 :** 새로운 access token을 발급한다. (만료기한은 발행시간 후 30분)

▶ Payment Service 구성



▶ Payment ERD

결제							Card
결제번호	payment_key	결제정보 식별자	INT	NOT NULL	Default value	Comment	
주문번호	merchant_uid	주문번호 식별자	VARCHAR	NOT NULL	Default value	AUTO_INCREMENT 사용	
결제요청금액	amount	Domain	INT	NOT NULL	Default value	Comment	
카드번호	card_number	Domain	VARCHAR	NULL	Default value	Comment	
카드만료일	expiry	Domain	VARCHAR	NULL	Default value	Comment	
카드소지자 생년월일	birth	Domain	VARCHAR	NULL	Default value	Comment	
카드 비밀번호 앞 2자리	pwd_2digit	Domain	VARCHAR	NULL	Default value	Comment	
카드 인증번호	cvc	Domain	VARCHAR	NULL	Default value	Comment	
결제 상태	successStatus	Domain	VARCHAR	NULL	Default value	Comment	
결제시간	time	Domain	DATETIME	NULL	Default value	Comment	

취소							Cancel
결제취소번호	payment_cancel_id	결제취소정보 식별자	INT	NOT NULL	Default value	Comment	
주문번호	merchant_uid	Domain	VARCHAR	NULL	Default value	Comment	
취소요청금액	cancel_amount	Domain	INT	NOT NULL	Default value	Comment	
총 금액	amount	Domain	INT	NOT NULL	Default value	Comment	
결제 상태	status	Domain	VARCHAR	NOT NULL	Default value	Comment	
취소시간	time	Domain	DATETIME	NULL	Default value	Comment	
주문번호	order_id	주문번호 식별자	INT	NOT NULL	Default value	AUTO_INCREMENT 사용	

▶ Payment REST API

설명	메소드	API
BaseUrl	POST	http://api.iimport.kr

설명	메소드	API
токن 생성	POST	/users/getToken

Parameter	설명	데이터 유형
1. imp_key	REST API 키	string
2. imp_secret	REST API Secret	string

Response Class (Status 200)

```
{ "code": 0, "message": null, "response": {
  "access_token": "dfa54492f2142dbf63d8c954f9de1b4f73ddcc23",
  "now": 1700699981, "expired_at": 1700701781 } }
```

필요하고 사용하는 값 : access_token

▶ Payment REST API

설명	메소드	API
결제	POST	/subscribe/payments/onetime

Parameter	설명	데이터 유형
1. merchant_uid	거래 고유번호	string(40)
2. amount	결제금액	int
3. card_number	카드번호 (dddd-dddd-dddd-dddd)	string
4. expiry	카드 유효기간 (YYYY-MM)	string
5. birth	생년월일6자리 (YYMMDD)	string
6. pwd_2digit	카드비밀번호 앞 2자리	string
7. cvc	카드 인증번호 (카드 뒷면 3자리)	string

필요한 값 : merchant_uid(주문번호) , status(결제상태)

주문으로 보내는 값 : merchant_uid(주문번호) , status(결제상태)

▶ Payment REST API

설명	메소드	API
결제취소	POST	/payments/cancel

Parameter	설명	데이터 유형
1. merchant_uid	거래 고유번호	string(40)
2. amount	결제금액	int

필요한 값 : merchant_uid(주문번호), amount(결제금액) , status(결제상태)

주문으로 보내는 값 : merchant_uid(주문번호) ,cancel_amount(취소금액), amount(결제금액) , remainingBalance (남은금액),
status(결제상태)

▶ Payment 받아야할 결제 정보 데이터

결제

- merchant_uid : 주문번호.
- amount : 결제요청금액.
- card_number : 카드번호(4100-1234-1234-1234).
- expiry : 카드 만료일(YYYY-MM)
- birth : 카드 소지자 생년월일6자리 (YYMMDD)
- pwd_2digit : 카드 비밀번호 앞 2자리
- cvc : 카드 인증번호

결제취소/환불

- merchant_uid : 주문번호
- amount : (부분)취소 요청금액 [전액 환불이면 주문번호만 있으면 됨.]

Order

01 프로젝트 개요

Order

▶ Order ERD

주문								Order		
주문번호	order_id	식별자	INT	NOT NULL	Default value	AUTO_INCREMENT 사용				
회원번호	user_id	Domain	INT	NOT NULL	Default value	Comment				
장바구니 상품 번호	cartItem_id	Domain	INT	NOT NULL	Default value	Comment				
총 주문금액	amount	Domain	INT	NOT NULL	Default value	결제 요청 금액				
받는사람 이름	reciever_name	Domain	VARCHAR	NOT NULL	Default value	Comment				
받는사람 연락처	reciever_phone	Domain	VARCHAR	NOT NULL	Default value	Comment				
받는사람 주소	reciever_addr	Domain	VARCHAR	NOT NULL	Default value	Comment				
주문시간	time	Domain	DATETIME	NOT NULL	Default value	Comment				
장바구니ID	cart_id	Domain	INT	NOT NULL	Default value	Comment				

주문내역								Order_History		
주문내역 번호	history_id	식별자	INT	NOT NULL	Default value	AUTO_INCREMENT 사용				
회원번호	user_id	Domain	INT	NOT NULL	Default value	Comment				
상품번호	item_id	Domain	INT	NOT NULL	Default value	Comment				
총 주문금액	order_amount	Domain	INT	NOT NULL	Default value	결제 요청 금액				
주문상태	order_state	Domain	ENUM	NOT NULL	'상품준비중'	'주문완료', '주문취소', '결제완료', '결제취소', '상품준비중', '배송준비중', '배송완료', '배송중', '판매자 취소'				
받는사람 이름	reciever_name	Domain	VARCHAR	NOT NULL	Default value	Comment				
받는사람 연락처	reciever_phone	Domain	VARCHAR	NOT NULL	Default value	Comment				
받는사람 주소	reciever_addr	Domain	VARCHAR	NOT NULL	Default value	Comment				
배송 요청 사항	reciever_message	Domain	VARCHAR	NULL	Default value	Comment				
주문시간	time	Domain	DATETIME	NOT NULL	Default value	Comment				
주문완료시간	order_time	Domain	DATETIME	NOT NULL	Default value	Comment				
주문번호	order_id	식별자	INT	NOT NULL	Default value	AUTO_INCREMENT 사용				
결제번호	payment_key	결제정보 식별자	INT	NOT NULL	Default value	Comment				

▶ 요구사항 확인

1. 사용자는 상품과 수량을 선택 후 주문할 수 있다.
2. 사용자는 주문을 취소할 수 있다.
3. 여러 서버간 통신을 통해 트랜잭션이 일어나므로 분산 추적이 가능해야된다.

▶ 주문 시나리오 1

1. 장바구니 측에서 주문 요청 이벤트 발생
2. 주문 요청 이벤트 수신
3. 회원 정보 요청 이벤트 발생
4. 재고 관련 이벤트(-), 결제 요청 이벤트 발생
5. 결제에 관한 이벤트 응답 결과 수신 후 배송 상태 이벤트 발생

▶ 주문 시나리오 2

6. 주문내역 이벤트 발행
7. 환불 시 결제 시스템에 취소 이벤트 발생
8. 환불과 주문 취소 등 비정상으로 처리 될 시 결제 측에서 이벤트 발생
9. 해당 이벤트 수신 후 재고 변화 이벤트(+) 발생
10. 상품 측에서 재고 변화 이벤트 수신

▶ 연관서비스 및 필요데이터

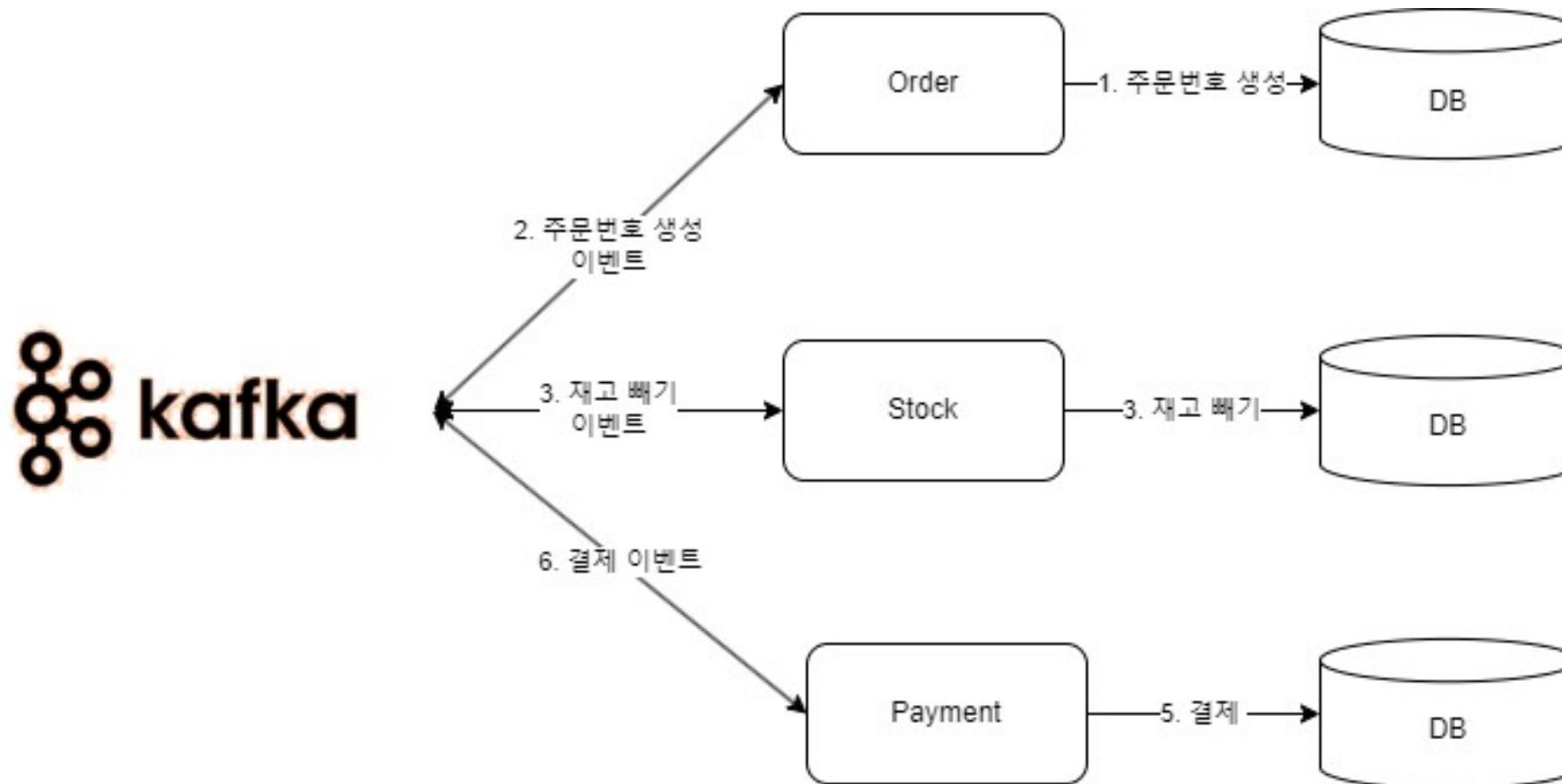
받을 데이터

- 회원 서비스
- 회원 아이디
- 회원 이름
- 회원 전화번호
- 결제 서비스
- 결제 상태
- 상품 서비스 / 장바구니
- 상품 정보
- 주문 수량

보낼 데이터

- 회원 서비스
회원 인덱스
- 결제 서비스
결제 총 금액
가맹점 주문 번호

▶ 분산 트랜잭션 프로세스 (Choreography 기준)



▶ 주문 요청

The screenshot shows the Postman application interface for making a POST request to the 'Order' API.

Request Details:

- Method: POST
- URL: <http://localhost:8072/gopang-order/api/v1/order>
- Headers: (8)
- Body (JSON):

JSON Body Content:

```
1 {  
2   "user_id": "3",  
3   "ReqItemOrder": [  
4     [  
5       {"  
6         "itemId": 1,  
7         "amount": 2  
8       },  
9       {"  
10        "itemId": 2,  
11        "amount": 2  
12      }  
13    ],  
14    "reciever_name": "김민규2",  
15    "reciever_phone": "01075595808",  
16    "reciever_addr": "대구광역시 중구 남산1동"  
17 }
```

Response Summary:

- Status: 200 OK
- Time: 1053 ms
- Size: 133 B

▶ DB 저장 데이터

```
MariaDB [gopang]> select * from orders;
+-----+-----+-----+-----+-----+-----+-----+
| amount | order_id | time | user_id | delete_yn | order_state | reciever_addr
+-----+-----+-----+-----+-----+-----+-----+
| 36000 | 1 | NULL | 3 | N | CANCEL | 대구광역시 중구 남산1동
+-----+-----+-----+-----+-----+-----+
1 row in set (0.001 sec)

MariaDB [gopang]> select * from order_details;
+-----+-----+-----+-----+-----+-----+-----+
| amount | detail_id | item_id | order_id | price | total_price | user_id | item_name
+-----+-----+-----+-----+-----+-----+-----+
| 2 | 1 | 1 | 1 | 9000 | 18000 | 3 | Sample Item1 |
| 2 | 2 | 2 | 1 | 9000 | 18000 | 3 | Sample Item2 |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.001 sec)
```

▶ 애플리케이션 테스트

주문 - 아이템 서버 간 RestTemplate 통신

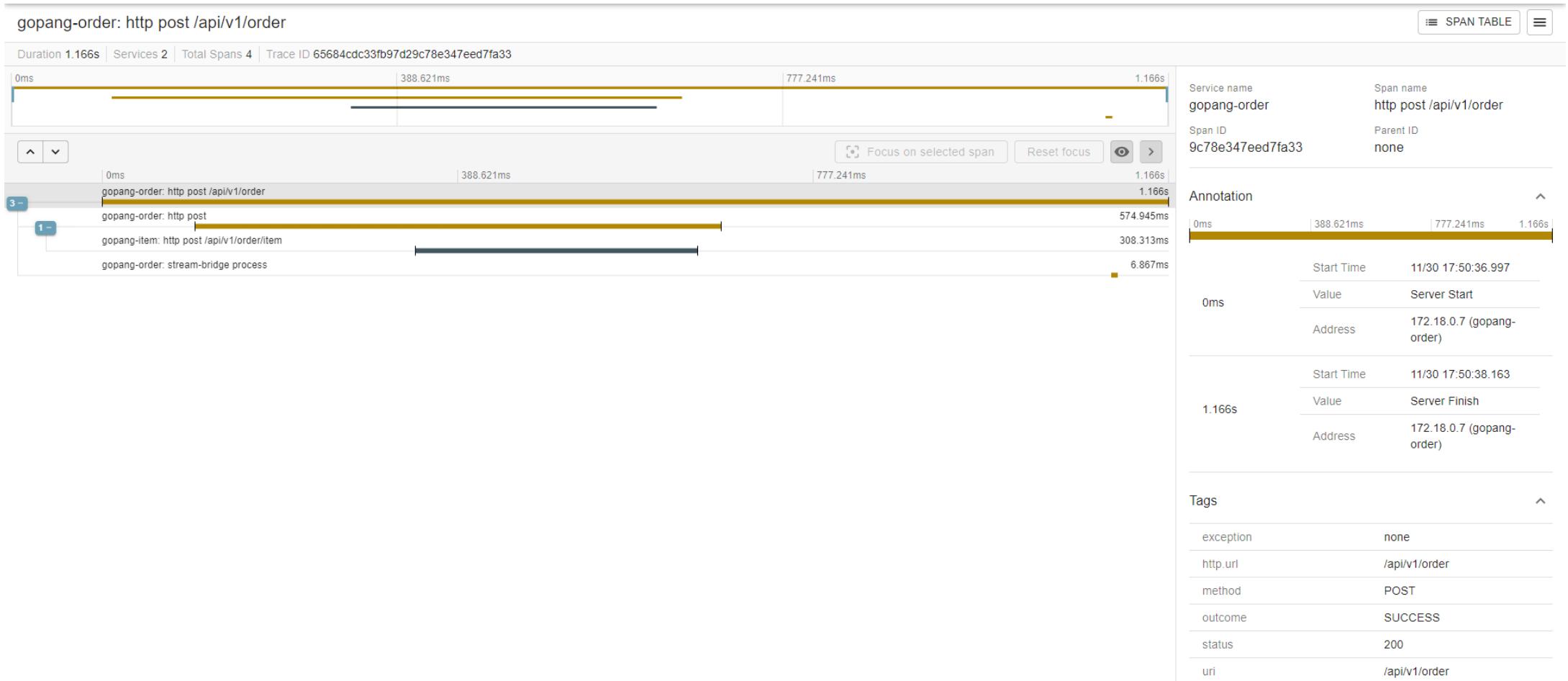
```
docker-compose-gatewayserver-1 | 2023-11-30T08:39:49.990Z TRACE 1 --- [or-http-epoll-3] o.s.w.s.adapter.HttpWebHandlerAdapter : [b941973a-3] HTTP POST "/gopang-order/api/v1/order"  
, headers={masked}  
docker-compose-gatewayserver-1 | The authentication name from the token is :  
docker-compose-gopangitems-1 | 2023-11-30T08:39:50.016Z " INFO [gopang-item,65684a561c8e17b20cf81658d574b23c,701fe2728be5049a]" 1 --- [nio-8081-exec-3] c.gopang.itemserver.service.  
ItemService : 상품 조회 요청 들어옴
```

▶ 애플리케이션 테스트

주문 - 아이템 서버 간 RestTemplate 통신

```
docker-compose-orderservice-1 | 2023-11-30T08:39:50.077Z "INFO [gopang-order,65684a561c8e17b20cf81658d574b23c,0cf81658d574b23c]" 1 --- [nio-8082-exec-3] c.g.o.event.OrderEventListene
ner : Recieved message to paymentRequest-topic : 2번 주문 결제 요청됨.
docker-compose-gatewayserver-1 | 2023-11-30T08:39:50.085Z TRACE 1 --- [or-http-epoll-3] o.s.w.s.adapter.HttpWebHandlerAdapter : [b941973a-3] Completed 200 OK, headers={masked}
docker-compose-gatewayserver-1 | 2023-11-30T08:39:50.085Z TRACE 1 --- [or-http-epoll-3] org.springframework.web.HttpLogging : [b941973a-3, L:/172.18.0.11:8072 - R:/172.18.0.1:34
686] Handling completed
docker-compose-gopangpayment-1 | 2023-11-30T08:39:50.245Z "INFO [gopang-payment,,]" 1 --- [container-0-C-1] c.g.paymentserver.Service.Payment_Start : KafkaPaymentReceived : {"order
_id":"2","amount":36000}
docker-compose-gopangpayment-1 | 2023-11-30T08:39:50.543Z "INFO [gopang-payment,,]" 1 --- [container-0-C-1] c.g.p.event.PaymentEventListener : Received message to paymentReq
uest-topic: 2번 결제실패
docker-compose-orderservice-1 | 2023-11-30T08:39:50.551Z "INFO [gopang-order,,]" 1 --- [container-0-C-1] c.g.orderservice.service.OrderService : 결제 상태 : PAYFAIL
```

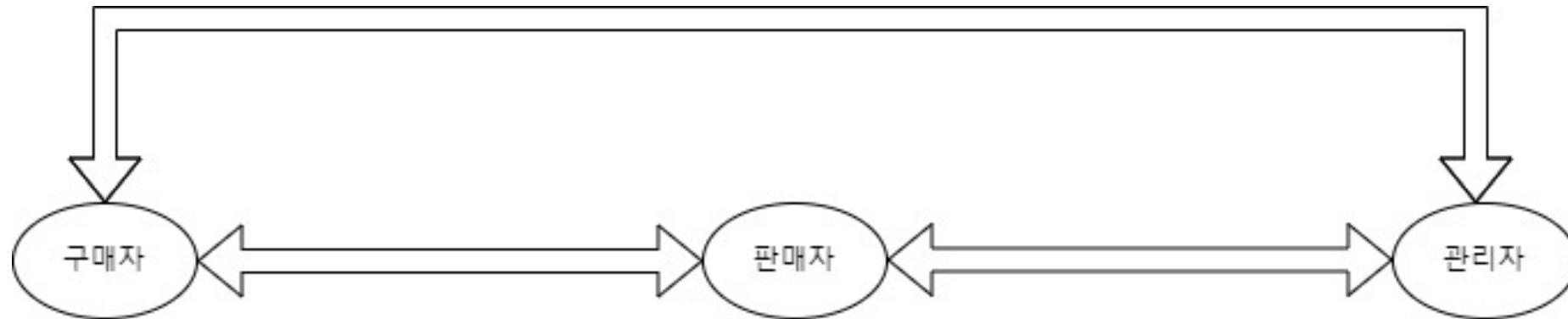
▶ Micrometer & Zipkin을 통한 분산추적 확인



Chat Counseling

채팅서버(Stomp)

상담 흐름(요구사항)



1. 구매자와 판매자의 상담
2. 판매자와 관리자의 상담
3. 구매자와 관리자의 상담

STOMP 프로토콜

STOMP는 TCP 또는 WebSocket 같은 양방향 네트워크 프로토콜 기반으로 동작한다.

이름에서 알 수 있듯, STOMP는 Text 지향 프로토콜이나, Message Payload에는 Text or Binary 데이터를 포함 할 수 있다.

위에서 언급한 **pub / sub**란 메세지를 공급하는 주체와 소비하는 주체를 분리해 제공하는 메세징 방법이다.

기본적인 컨셉을 예로 들자면 우체통(Topic)이 있다면 집배원(Publisher)이 신문을 우체통에 배달하는 행위가 있고, 우체통에 신문이 배달되는 것을 기다렸다가 빼서 보는 구독자(Subscriber)의 행위가 있다.

이때 구독자는 다수가 될 수 있다. pub / sub 컨셉을 채팅방에 빗대면 다음과 같다.

채팅방 생성 -> 채팅방 입장 -> 채팅방에서 메세지를 송수신

클라이언트는 메세지를 전송하기 위해 **SEND, SUBSCRIBE COMMAND**를 사용할 수 있다.

또한, **SEND, SUBSCRIBE COMMAND** 요청 Frame에는 메세지가 무엇이고,
누가 받아서 처리할지에 대한 Header 정보가 포함되어 있다.

이런 명령어들은 "destination" 헤더를 요구하는데 이것이 어디에 전송할지,
혹은 어디에서 메세지를 구독할 것 인지를 나타낸다.

위와 같은 과정을 통해 STOMP는 Publish-Subscribe 매커니즘을 제공한다.

즉 Broker를 통해 타 사용자들에게 메세지를 보내거나 서버가 특정 작업을 수행하도록 메세지를 보낼 수 있게 된다.

만약 **Spring**에서 지원하는 STOMP를 사용하면 Spring WebSocket 어플리케이션은 STOMP Broker로 동작하게 된다.

Spring에서 지원하는 STOMP는 많은 기능을 하는데

예를 들어 Simple In-Memory Broker를 이용해 SUBSCRIBE 중인 다른 클라이언트들에게 메세지를 보내준다.

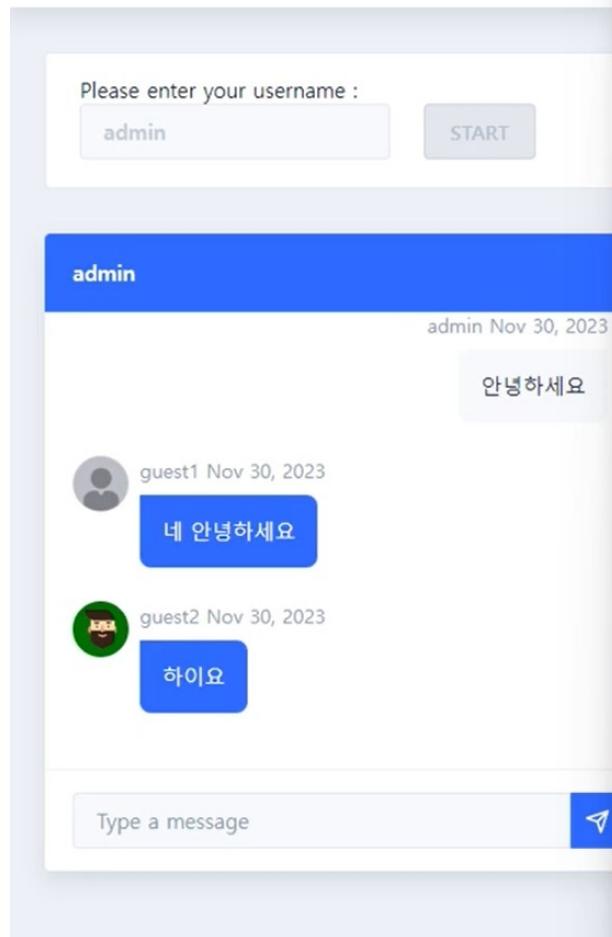
Simple In Memory Broker는 클라이언트의 SUBSCRIBE 정보를 자체적으로 메모리에 유지한다.

또한 RabbitMQ, ActiveMQ같은 외부 메세징 시스템을 STOMP Broker로 사용할 수 있도록 지원한다.

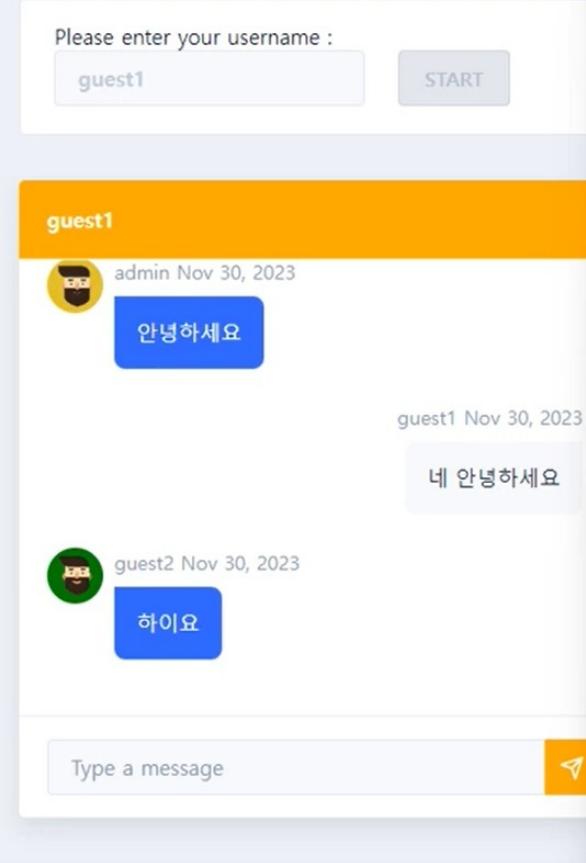
구조적인 면을 보자면, [스프링은 메세지를 외부 Broker에게 전달](#)하고, [Broker는 WebSocket으로 연결된 클라이언트에게 메세지를 전달](#)하는 구조가 되겠다.

이와 같은 구조 덕분에 HTTP 기반의 보안 설정과 공통된 검증 등을 적용할 수 있게 된다.

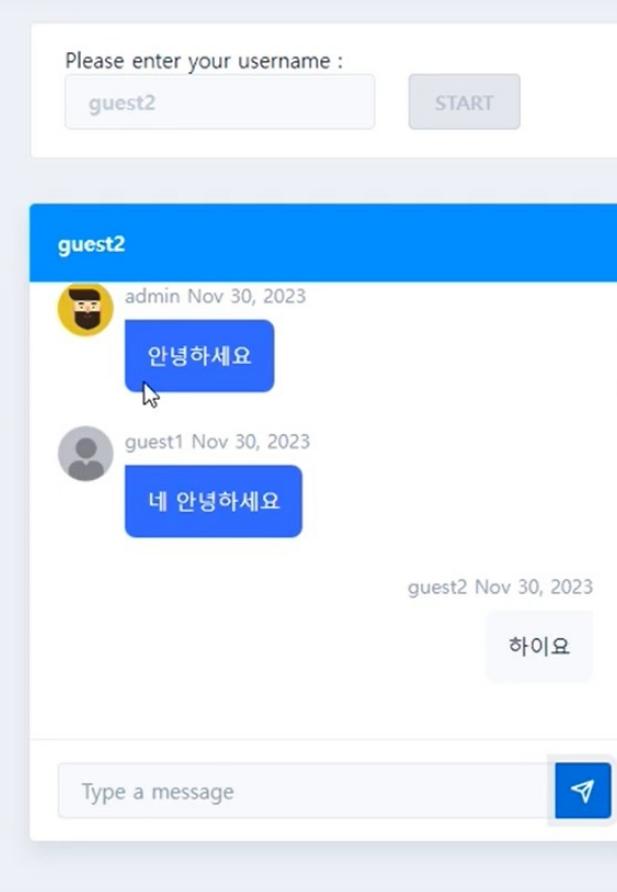
Real-time chat application using **Spring web-socket** and **Angular**.



Real-time chat application using **Spring web-socket** and **Angular**.



Real-time chat application using **Spring web-socket** and **Angular**.



Product

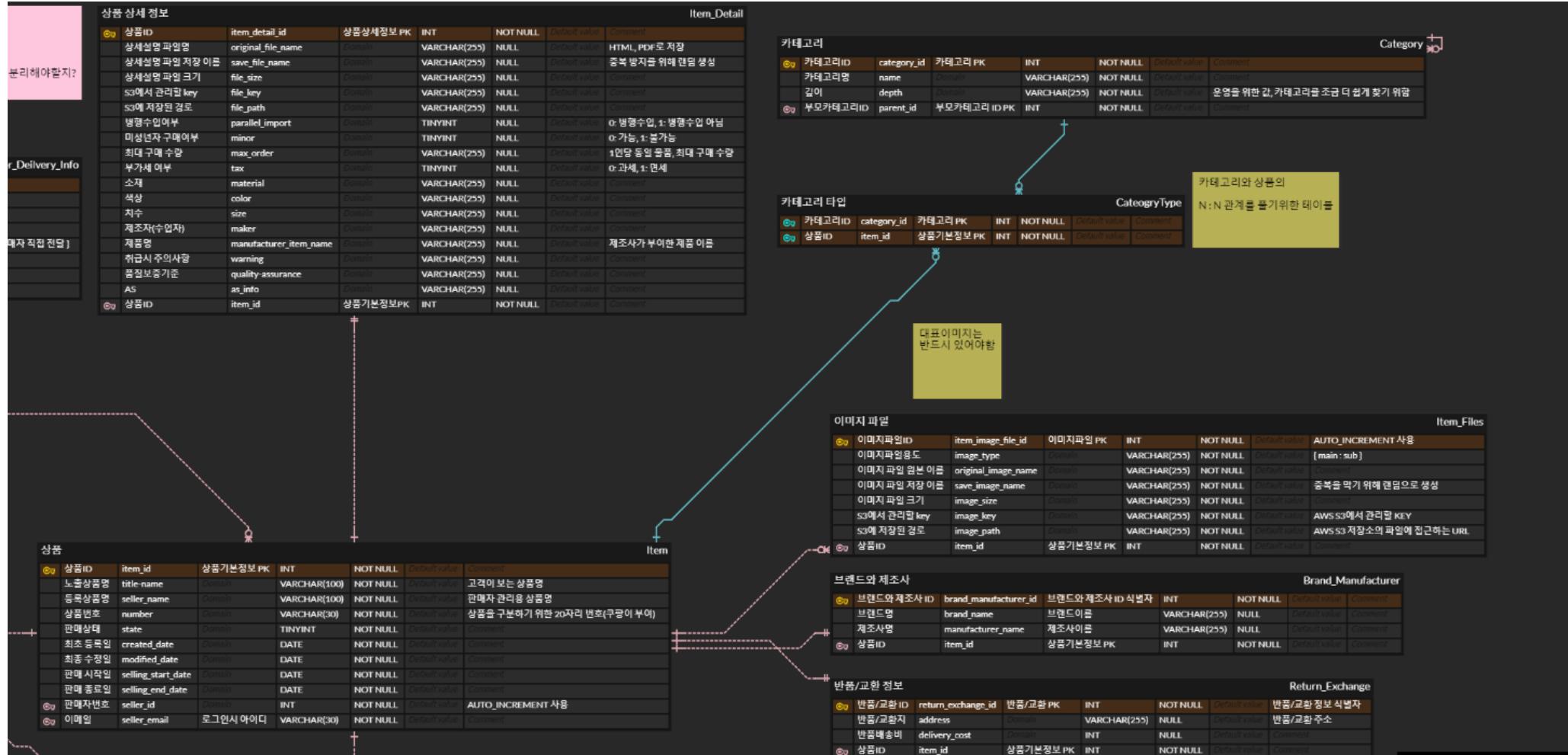
▶ 요구사항

1. 판매자는 상품을 등록, 수정할 수 있다.
2. 관리자는 카테고리를 등록, 수정, 삭제할 수 있다.
3. 회원은 상품을 전체 조회할 수 있다.
4. 각 Microservices는 Config server를 통해 설정 정보를 받는다.
5. 사용자는 API Gateway를 통해 각각의 Microservices에 접근할 수 있다.

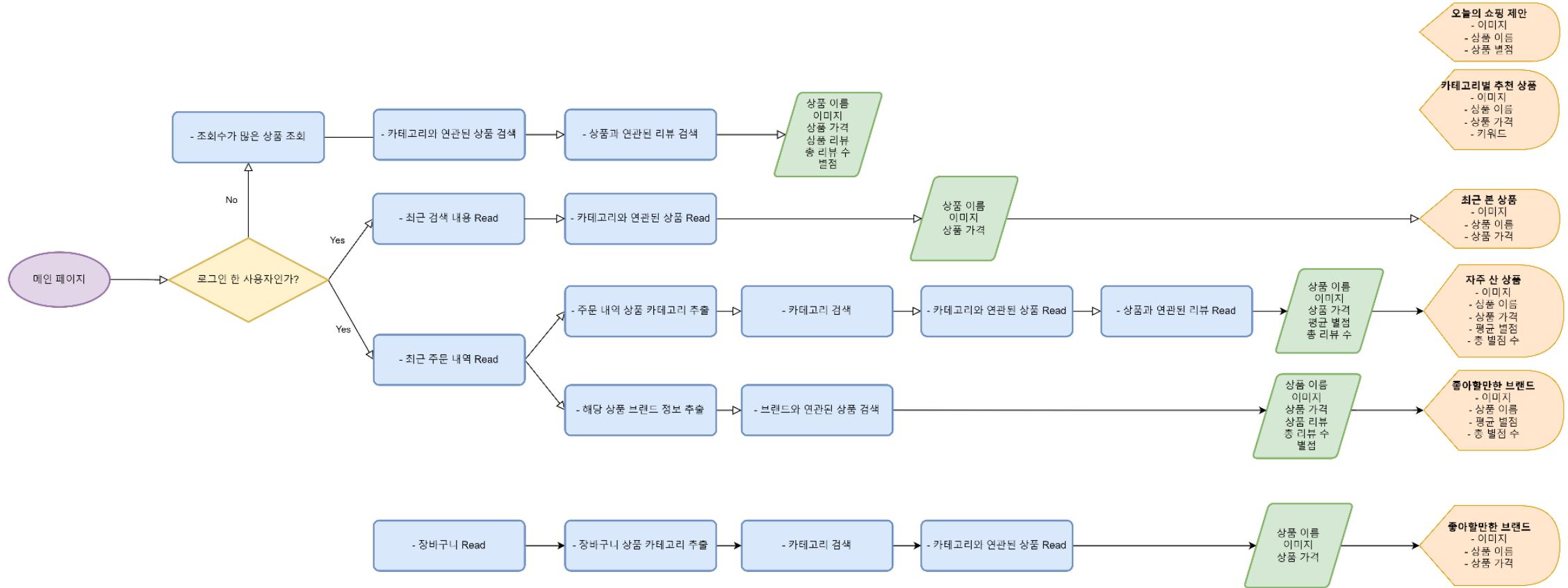
01 프로젝트 개요

Product

▶ Product ERD



▶ 상품 조회 FlowChart



▶ 어플리케이션 테스트(Eureka Discovery)

DS Replicas			
Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
GOPANG-GATEWAY	n/a (1)	(1)	UP (1) - aed9e48999d1:gopang-gateway:8072
GOPANG-ITEM	n/a (1)	(1)	UP (1) - fcbfa025c2bb:gopang-item:8081
GOPANG-OAUTH2	n/a (1)	(1)	UP (1) - 7eca048cec97:gopang-oauth2:8085
GOPANG-ORDER	n/a (1)	(1)	UP (1) - 5ae8c04beb1d:gopang-order:8082
GOPANG-PAYMENT	n/a (1)	(1)	UP (1) - e2e69f127f18:gopang-payment:8073
General Info			
Name	Value		
total-avail-memory	74mb		
num-of-cpus	8		
current-memory-usage	39mb (52%)		
server-upptime	00:06		
registered-replicas			
unavailable-replicas			
available-replicas			

▶ 어플리케이션 테스트 (API Gateway, Microservices Mapping)

```
# Server Configuration
server:
  port: 8072

# Eureka Configuration
eureka:
  instance:
    preferIpAddress: true
  client:
    registerWithEureka: true
    fetchRegistry: true
  serviceUrl:
    defaultZone: http://eurekaserver:8070/eureka/

spring:
  application:
    name: gopang-gateway
  config:
    activate:
      on-profile: default

# Spring Cloud Configuration
cloud:
  loadbalancer.ribbon.enabled: false

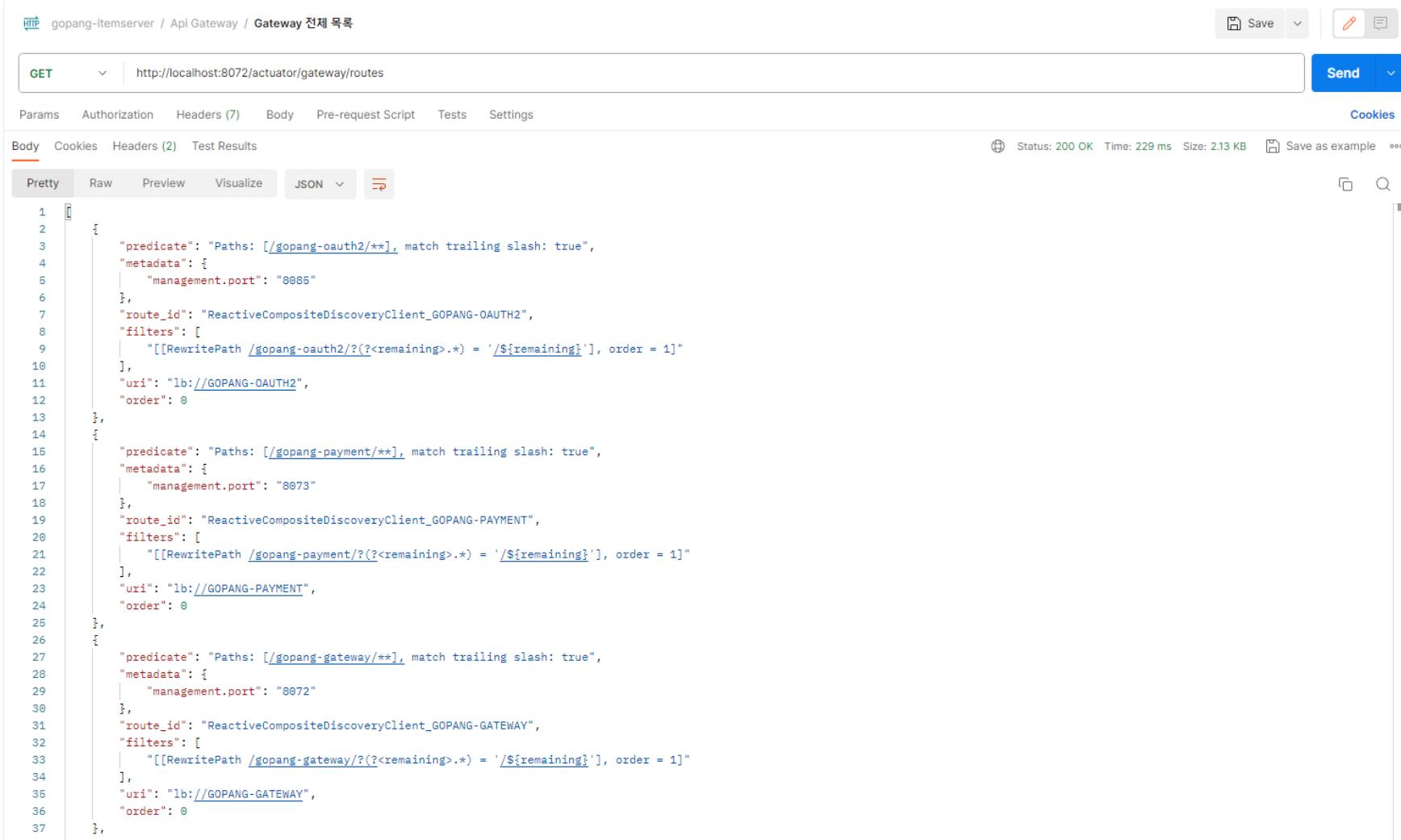
# Gateway Configuration
gateway:
  default-filters: DedupeResponseHeader=Access-Control-Allow-Origin Access-Control-Allow-Credentials
```

```
discovery.locator:
  enabled: true
  lowerCaseServiceId: true
routes:
  # Item server
  - id: gopang-item
    uri: lb://gopang-item # Eureka에 등록된 서비스 이름
    # 요청할 경로 조건
    predicates:
      - Path=/item/**
    filters:
      - RewritePath=/item/(?<path>.*), /${path}
      - RemoveRequestHeader=Cookie
      - RemoveRequestHeader=Set-Cookie

  # Order server
  - id: gopang-order
    uri: lb://gopang-order
    predicates:
      - Path=/order/**
    filters:
      - RewritePath=/order/(?<path>.*), /${path}
      - RemoveRequestHeader=Cookie
      - RemoveRequestHeader=Set-Cookie

  # OAuth2 Server
  - id: gopang-oauth2
    uri: lb://gopang-oauth2
    predicates:
```

▶ 어플리케이션 테스트 (API Gateway)



The screenshot shows the Postman application interface. A GET request is made to `http://localhost:8072/actuator/gateway/routes`. The response body is a JSON object containing three route definitions:

```
1 [
2   {
3     "predicate": "Paths: [/gopang-oauth2/**], match trailing slash: true",
4     "metadata": {
5       "management.port": "8085"
6     },
7     "route_id": "ReactiveCompositeDiscoveryClient_GOPANG-OAUTH2",
8     "filters": [
9       "[[RewritePath /gopang-oauth2/?(<remaining>.*)) = '/${remaining}'], order = 1]"
10    ],
11    "uri": "lb://GOPANG-OAUTH2",
12    "order": 0
13  },
14  {
15    "predicate": "Paths: [/gopang-payment/**], match trailing slash: true",
16    "metadata": {
17      "management.port": "8073"
18    },
19    "route_id": "ReactiveCompositeDiscoveryClient_GOPANG-PAYMENT",
20    "filters": [
21      "[[RewritePath /gopang-payment/?(<remaining>.*)) = '/${remaining}'], order = 1]"
22    ],
23    "uri": "lb://GOPANG-PAYMENT",
24    "order": 0
25  },
26  {
27    "predicate": "Paths: [/gopang-gateway/**], match trailing slash: true",
28    "metadata": {
29      "management.port": "8072"
30    },
31    "route_id": "ReactiveCompositeDiscoveryClient_GOPANG-GATEWAY",
32    "filters": [
33      "[[RewritePath /gopang-gateway/?(<remaining>.*)) = '/${remaining}'], order = 1]"
34    ],
35    "uri": "lb://GOPANG-GATEWAY",
36    "order": 0
37  }
]
```

The status bar at the bottom indicates `Status: 200 OK Time: 229 ms Size: 2.13 KB`.

▶ 어플리케이션 테스트(판매자 상품 등록)

The screenshot shows a Postman interface for testing a REST API endpoint. The URL is `http://localhost:8081/api/v1/seller/item`. The request method is `POST`. The `Body` tab is selected, showing a JSON payload:

```
1 {  
2   "categoryIds": 1,  
3   "itemSaveDto": {  
4     "sellerId": 1,  
5     "sellerDeliveryId": 2,  
6     "sellerREId": 3,  
7     "titleName": "Sample Item",  
8     "itemLabel": "Sample Seller",  
9     "sellState": "판매중",  
10    "sellStartDate": "2023-11-09",  
11    "sellEndDate": "2023-12-09"  
12  },  
13 };
```

The response status is `200 OK` with a time of `214 ms` and a size of `420 B`.

Below the main interface, there is a detailed view of the JSON response body:

```
1 {  
2   "itemId": 2,  
3   "titleName": "Sample Item",  
4   "sellerName": "Sample Seller",  
5   "state": "판매중",  
6   "sellStartDate": "2023-11-09",  
7   "sellEndDate": "2023-12-09",  
8   "sellerId": 1,  
9   "sellerDeliveryId": 2,  
10  "sellerREId": 3,  
11  "itemDetailId": 2,  
12  "itemOptionIds": [  
13    3,  
14    4  
15  ],  
16  "brandManufacturerIds": 2  
17 };
```

▶ 어플리케이션 테스트(판매자 상품 수정)

The screenshot shows the Postman application interface for testing a REST API. The URL is `http://localhost:8081/api/v1/seller/item/1`. The request method is `PUT`. The `Body` tab is selected, displaying the following JSON payload:

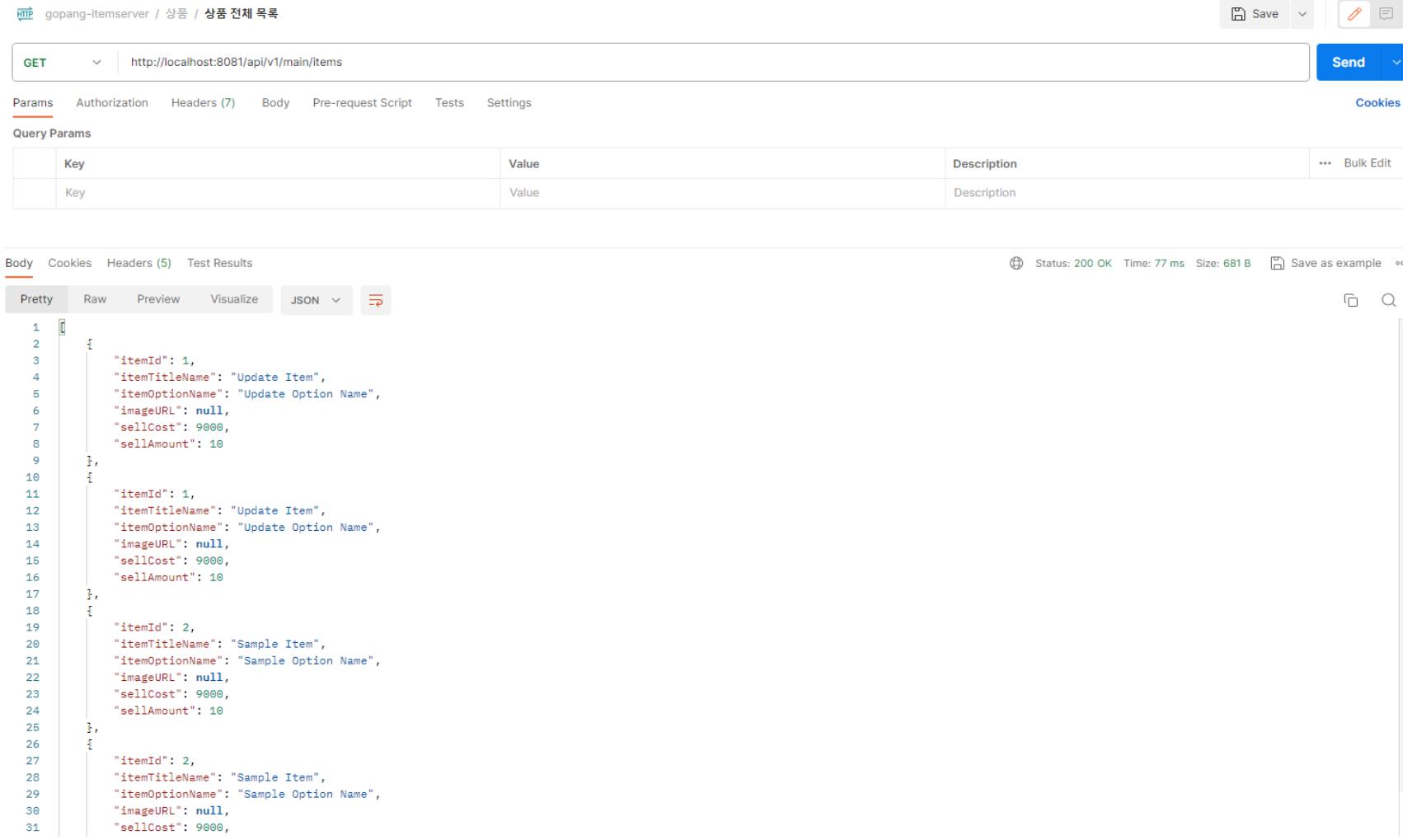
```
50     ... ],
51     ...
52
53
54     "brandManufacturerUpdateDto": [
55         {
56             "brandManufacturerId": 1,
57             "brandName": "Update Brand",
58             "manufacturerName": "Update Manufacturer"
59         }
60     ]
61 }
```

The response status is `200 OK` with a time of `96 ms` and a size of `419 B`.

Below the body, the response JSON is shown in a code editor-like view:

```
1
2     "itemId": 1,
3     "titleName": "Update Item",
4     "itemLabel": "Update Seller",
5     "state": "판매중",
6     "sellStartDate": "2023-11-09",
7     "sellEndDate": "2023-12-09",
8     "sellerId": 1,
9     "sellerDeliveryId": 2,
10    "sellerREID": 3,
11    "itemDetailId": 1,
12    "itemOptionIds": [
13        1,
14        2
15    ],
16    "brandManufacturerIds": 1
17 }
```

▶ 어플리케이션 테스트(상품 전체목록)



HTTP gopang-itemserver / 상품 / 상품 전체 목록

Save |  

Send |  

GET http://localhost:8081/api/v1/main/items

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	...

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 77 ms Size: 681 B Save as example ...

Pretty Raw Preview Visualize JSON 

```
1 {
2   "itemId": 1,
3   "itemTitleName": "Update Item",
4   "itemOptionName": "Update Option Name",
5   "imageURL": null,
6   "sellCost": 9000,
7   "sellAmount": 10
8 },
9 {
10   "itemId": 1,
11   "itemTitleName": "Update Item",
12   "itemOptionName": "Update Option Name",
13   "imageURL": null,
14   "sellCost": 9000,
15   "sellAmount": 10
16 },
17 {
18   "itemId": 2,
19   "itemTitleName": "Sample Item",
20   "itemOptionName": "Sample Option Name",
21   "imageURL": null,
22   "sellCost": 9000,
23   "sellAmount": 10
24 },
25 {
26   "itemId": 2,
27   "itemTitleName": "Sample Item",
28   "itemOptionName": "Sample Option Name",
29   "imageURL": null,
30   "sellCost": 9000,
```

▶ 어플리케이션 테스트(카테고리 등록, 수정, 삭제)

The screenshot shows three separate Postman requests for managing categories:

- Request 1 (POST):** Creates a new category. The body is a JSON object with fields: name ("운동화"), parentId (1), and depth (1). Response: Category created successfully.
- Request 2 (PUT):** Updates an existing category. The body is a JSON object with fields: name ("남성구두"), parentId (2), and depth (2). Response: Category updated successfully.
- Request 3 (DELETE):** Deletes a category by ID. Response: Category deleted successfully.

▶ 어플리케이션 테스트(카테고리 전체 조회)

The screenshot shows the Postman application interface for testing an API endpoint. The URL is `http://localhost:8081/api/v1/categories`. The response status is `200 OK`, time is `58 ms`, and size is `577 B`.

Body tab content (Pretty JSON view):

```
1 {
2     "categoryId": 1,
3     "name": "Root Category",
4     "depth": 0,
5     "children": [
6         {
7             "categoryId": 2,
8             "name": "Child Category 1",
9             "depth": 1,
10            "children": [
11                {
12                    "categoryId": 4,
13                    "name": "Grandchild Category 1",
14                    "depth": 2,
15                    "children": []
16                },
17                {
18                    "categoryId": 5,
19                    "name": "Grandchild Category 2",
20                    "depth": 2,
21                    "children": []
22                },
23                {
24                    "categoryId": 6,
25                    "name": "Grandchild Category 3",
26                    "depth": 2,
27                    "children": []
28                }
29            ]
30        }
31    ]
32}
```

Member

▶ Member ERD

회원						
	회원번호	user_id	회원 식별자	INT	NOT NULL	Default value
	이메일	user_email	로그인시 아이디	VARCHAR(30)	NOT NULL	Default value
	비밀번호	user_password	Domain	VARCHAR(20)	NOT NULL	Default value 이메일 형식에 맞게 작성
	이름	user_name	Domain	VARCHAR(30)	NOT NULL	Default value Comment
	전화번호	user_phonenumber	Domain	VARCHAR(20)	NOT NULL	Default value 전화번호 형식에 맞게 작성
	우편번호	user_zipcode	Domain	VARCHAR(5)	NOT NULL	Default value Comment
	주소	user_address	Domain	VARCHAR(50)	NOT NULL	Default value Comment
	상세주소	user_detail_address	Domain	VARCHAR(50)	NOT NULL	Default value Comment

▶ OAuth2.x

OAuth 2.x은 무엇인가?

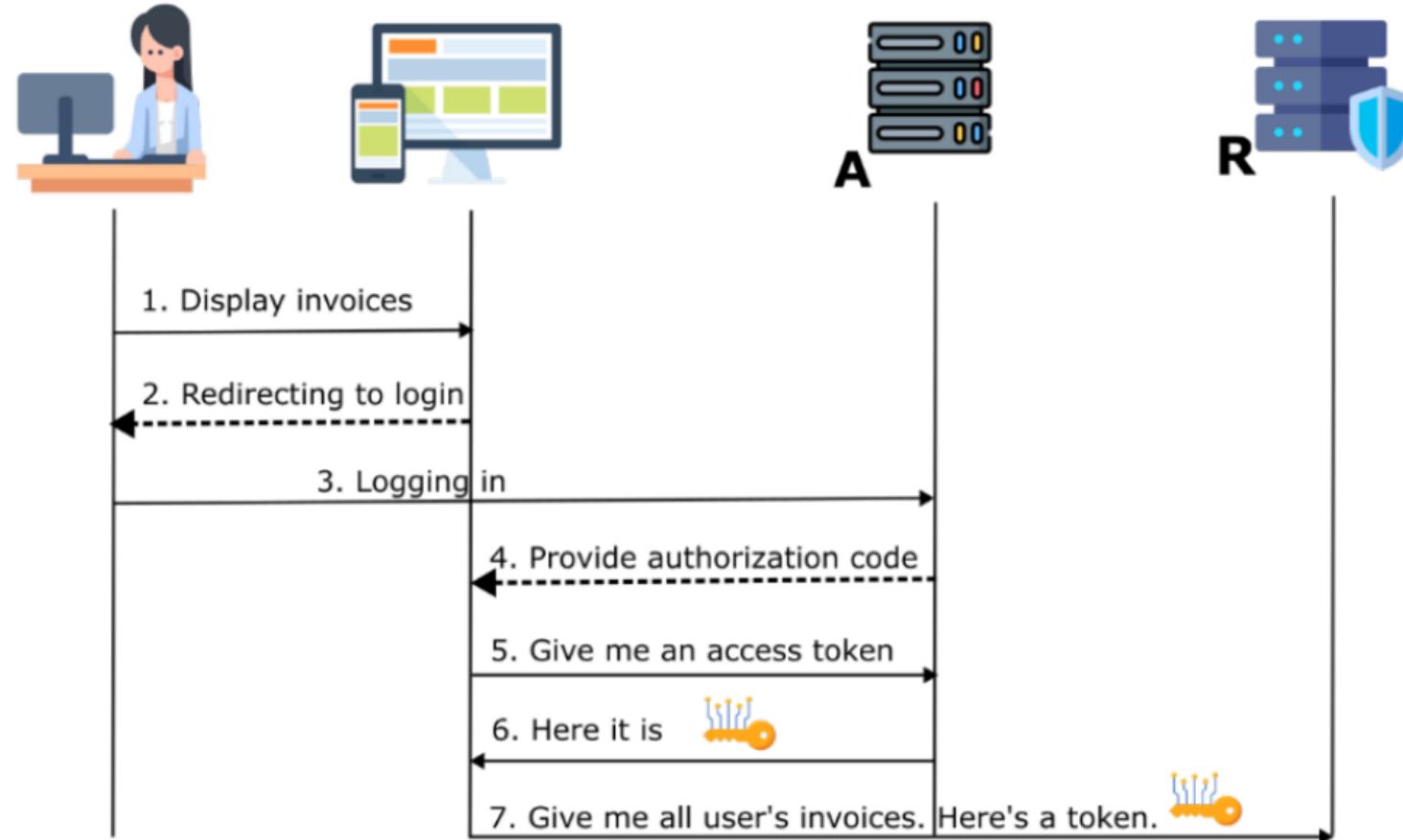
OAuth 2.x은 자신이 소유한 리소스에 소프트웨어 애플리케이션이 접근할 수 있도록 허용해줌으로써 접근권한을 위임해주는 프로토콜이다.

즉 클라이언트가 리소스 소유자에게 리소스에 대한 접근 권한을 요청하고, 요청 결과를 이용해 토큰을 요청하고 전달받은 토큰을 이용해 클라이언트가 해당 리소스에 접근하는 프로토콜이다.

(클라이언트 : 클라이언트는 웹 브라우저가 아닌, 리소스 소유자를 대신해 보호된 리소스에 접근하는 소프트웨어 요소 적어도 OAuth에선 클라이언트가 웹브라우저를 의미 하지 않음)

Bearer Token이란? 토큰을 소유한 사람에게 액세스 권한을 부여하는 일반적인 토큰 클래스

▶ Authorization Code Grant Type으로 토큰 가져오기



- 점선화살표는 redirection을 의미(요청 또는 응답 x)
- Authorization code ≠ Access token
- Access token : 클라이언트가 보호된 리소스에 접근하기 위해 필요한 토큰 (단계7)

단계4에서 Authorization code 대신에 직접 Access token을 주면 단계가 줄어 들 것이라고 생각할 수 있다.

OAuth1에서는 단계4에서 Access token을 바로 주었다 그것이 "**Implicit Grant Type**"이다.

하지만 이렇게 해버리면 **redirect**가 쉽게 Intercept될 수 있고 개인이 Access Token을 쉽게 얻을 수 있기 때문에 Authorization Code Grant Type을 사용했을 때 발생하는 Authorization Code를 가로채어 가도 Authorization Code만 가지고는 Access Token을 얻기에는 불가능하다.

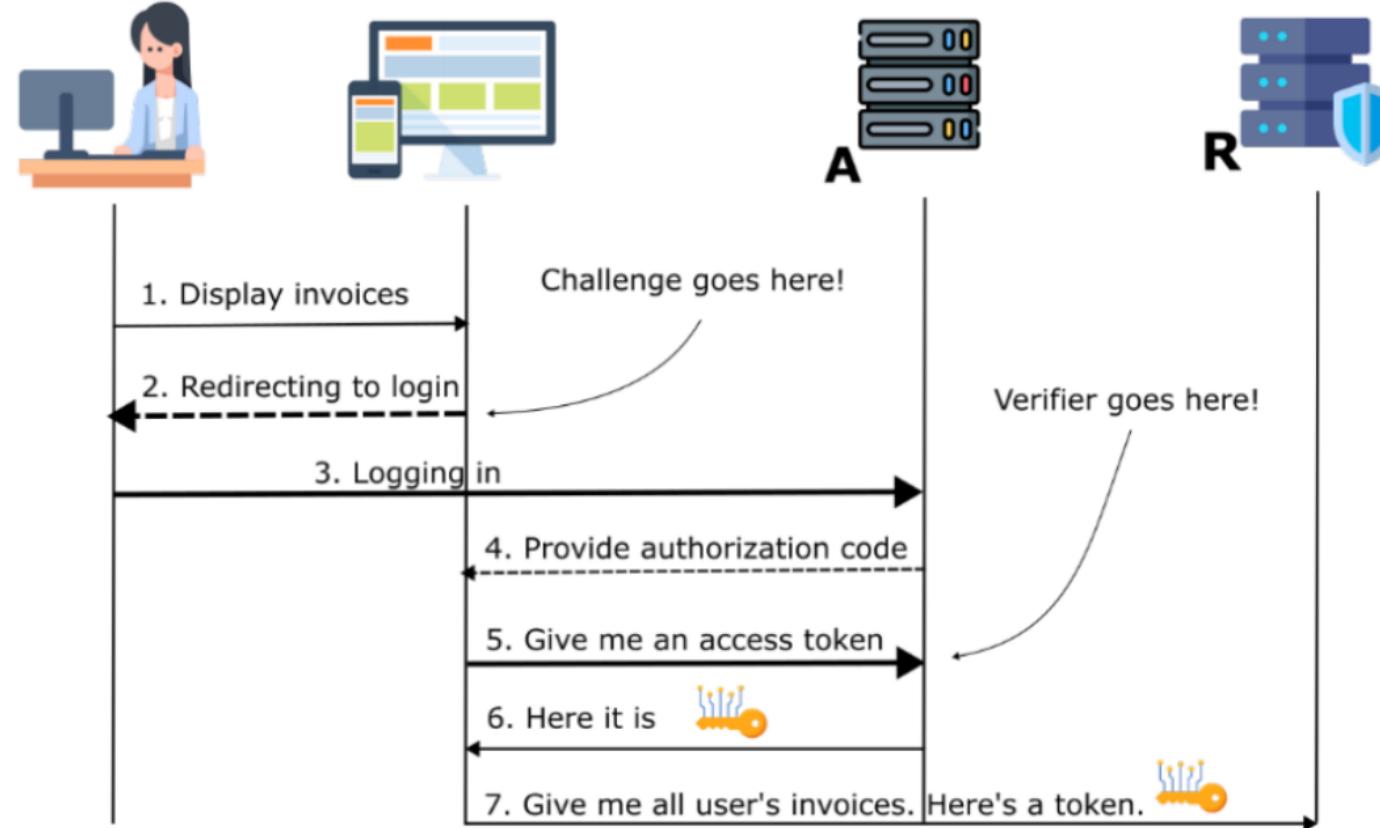
요청을 보내고 토큰을 얻기 위해서는 클라이언트 자격 증명도 알아야 하기 때문이다.

▶ Authorization Code Grant Type에 PKCE 보호 적용

나쁜 의도를 가진 개인이 인가 코드(authorization code)와 클라이언트 자격 증명을 얻는다면 액세스 토큰을 얻어내고 Resource Server로 요청을 보낼 수 있을 것이다.
이것을 방지하기 위해 PKCE(픽시)라는 보호기능을 추가한다.

- PKCE는 위 그림의 3단계와 5단계에 적용할 수 있다.
 1. client는 임의의 값 생성을 해줘야한다. 이값은 임의의 Byte String일 수 있고, 이 값을 "verifier"로 한다.
 2. client는 무작위로 생성된 값을 해시함수에 적용한다. 해시함수는 **출력을 입력으로 돌이킬 수 없는 특징을 가진 암호화 방법**이다. verifier에 해시함수를 적용한 결과를 challenge에 저장한다.

```
verifier = random();
challenge = hash(verifier);
```



client는 단계 3에서 verifier가 해시 함수로 적용된 challenge를 사용자 로그인 정보와 함께 전송한다.

authorization server는 Challenge를 유지하고, 단계 5에서 요청과 함께 클라이언트가 보낸 verifier를 전달 받았을 때, verifier가 Challenge와 일치하는 경우 액세스 토큰을 보내준다.

그래서 단계 4에서 Authorization code를 얻어내도 client가 아직 verifier를 전송하지 않았기 때문에 액세스 토큰을 얻어낼 수가 없다.

또한, 단계 3에서 challenge를 가로챈다 한들, 해시 함수로 생성되었기에 verifier 값을 알 수가 없다.

▶ Client Credentials Grant 으로 토큰 가져오기

일정한 프로세스의 타이머와 같이 서비스가 다른 서비스를 호출해야 하는 경우처럼 앱이 사용자(user) 개입 없이

서비스 간 승인을 받아야 할 때 이 grant type을 사용한다. (Order 서비스에서 Payment 서비스로 접근할 때)

특정 client(사용자)의 데이터 액세스 권한이 필요하지 않은 시스템 간의 인증에 적합하다.

이러면 앱은 client 자격 증명으로만 인증하면 된다.

1. 앱은 authorization server에서 액세스 토큰을 요청한다. 앱은 자신의 자격증명을 사용하여 인증한다.
2. 자격증명이 유효하면 authorization server는 액세스 토큰을 발급한다.
3. 앱은 액세스 토큰을 사용하여 resource server에 요청을 보낼 때 인증을 받는다.

▶ Refresh Token (갱신토큰)

기존 엑세스 토큰은 토큰 분실, 탈취 및 보안상의 이유로 수명이 10-15분으로 짧게 돼 있는데, 그러면 수명이 다하면 계속 다시 재인증 받고 token을 받는 앞에 과정을 또 해야 하는 번거로움 작업이 추가된다(즉 로그아웃되고 다시 로그인 과정을 계속 해야한다는 이야기)

그러나 갱신 토큰이 있으면 엑세스 토큰이 만료돼도 만료가 됐다고 리소스 서버에서 알려줘도 다시 로그 아웃하고 로그인하는 번거로움 작업이 필요없이 그냥 엑세스 토큰 대신 갱신토큰으로 갱신이 가능하다. 클라이언트가 권한 부여 서버로 갱신 토큰을 가지고 엑세스 토큰과 새로운 갱신 토큰을 받음 그걸 가지고 다시 보호된 리소스 서버에 접근.

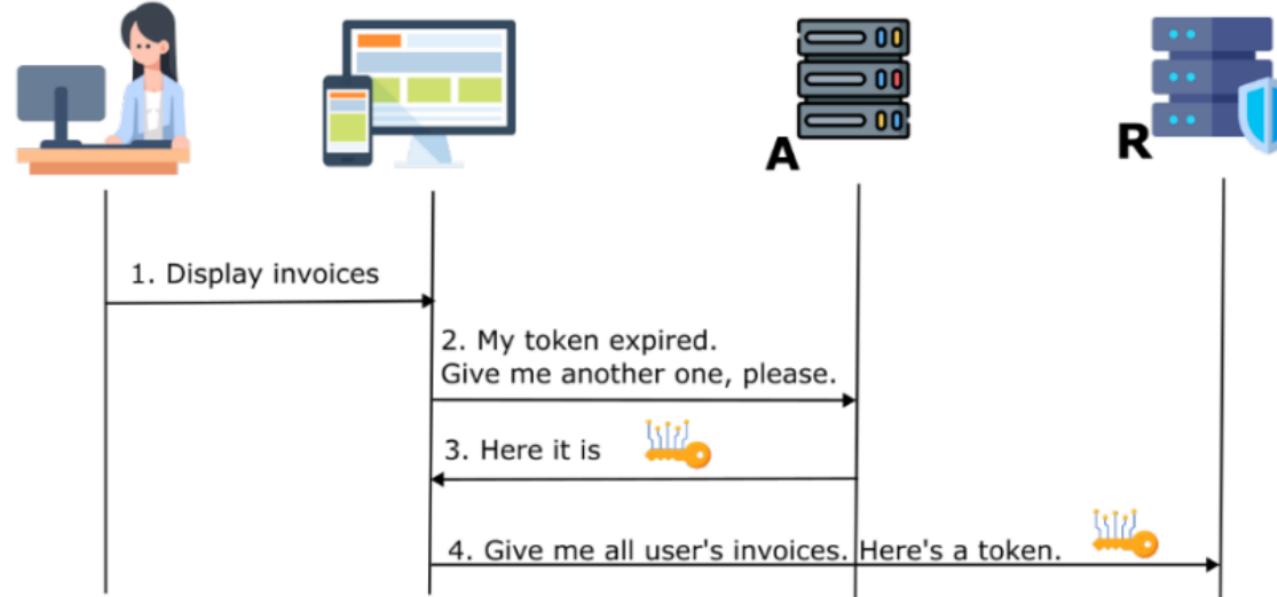
client가 토큰을 가지고 있고 토큰이 만료된 경우 두 가지 방법이 있다.

1. grant type 단계를 반복하여 새로운 액세스 토큰을 얻는 방법

예를 들어 authorization code grant type 일 경우엔 사용자에게 다시 로그인을 요청하는 것이 된다.

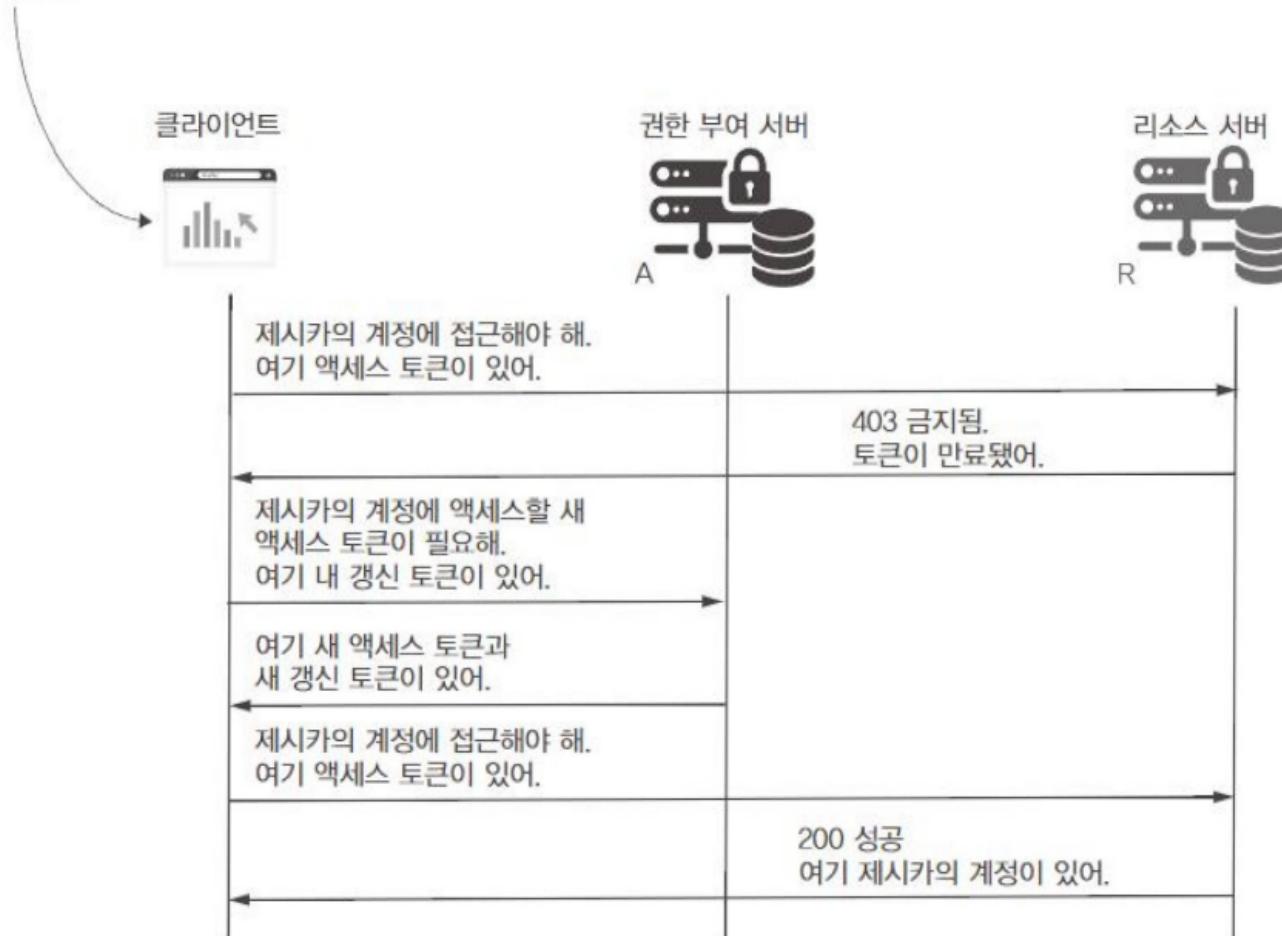
2. Refresh Token 사용

액세스 토큰이 만료 될 때마다 사용자에게 다시 로그인 하도록 요청하는 대신 refresh token을 사용하여 새로운 액세스 토큰을 얻을 수 있다.



1. 사용자는 일부 데이터를 요청한다. 즉 client가 권한 부여 서버로 접근해야 한다는 것.
2. client는 이전에 이미 인증했다는 것을 증명하기 위해 refresh token을 전송한다.
3. authorization server는 refresh token을 인식하고 client에게 새로운 액세스 토큰을 제공한다.
4. client는 새로운 액세스 토큰을 얻었고, 보호된 resource server 엔드 포인트에 접근이 가능.

클라이언트에 이전에 사용자 제시카에게
발급된 토큰이 있다.



▶ UserInfo Endpoint Configurer

```
@RestController
public class UserInfoController {

    @GetMapping("/userinfo")
    @ResponseBody
    public Map<String, Object> userInfo(@AuthenticationPrincipal OAuth2AuthenticatedPrincipal principal) {

        // OAuth2AuthenticatedPrincipal에서 사용자 정보 추출 및 반환
        Map<String, Object> userInfo = new HashMap<>();

        userInfo.put("sub", principal.getName()); // Subject

        // 추가적으로 필요한 클레임 정보 추가
        return userInfo;
    }
}
```

1 '.oidc()'는 OpenID Connect(OIDC)를 구성하는 부분으로, OIDC의 UserInfo 엔드포인트에 대한 설정을 제공합니다.

2. userInfoEndpoint 내부에서는 UserInfo 엔드포인트에 대한 다양한 구성을 할 수 있습니다.

- userInfoRequestConverter는 UserInfo 요청을 변환하는 컨버터를 설정합니다.
- userInfoRequestConverters는 UserInfo 요청을 변환하는 데 사용되는 컨버터를 소비하는 함수를 제공합니다.
- authenticationProvider는 UserInfo 엔드포인트에서 사용할 인증 프로바이더를 설정합니다.
- authenticationProviders는 사용할 인증 프로바이더를 소비하는 함수를 제공합니다.
- userInfoResponseHandler는 UserInfo 응답을 처리하는 핸들러를 설정합니다.
- errorResponseHandler는 오류 응답을 처리하는 핸들러를 설정합니다.
- userInfoMapper는 UserInfo 응답을 매핑하는 데 사용될 UserInfo 매퍼를 설정합니다.

UserInfo는 사용자에 대한 프로필 정보를 포함하는 것으로, 이 정보는 주로 클라이언트에 의해 사용, Authorization Server의 User Info 엔드포인트를 통해 클라이언트가 인가된 사용자의 프로필 정보를 요청할 수 있습니다.

OAuth2.x 프로토콜에서는 사용자 정보 엔드포인트를 통해 요청된 정보가 사용자의 권한에 따라 제공되고, 클라이언트가 해당 정보에 접근할 수 있도록 허용합니다.

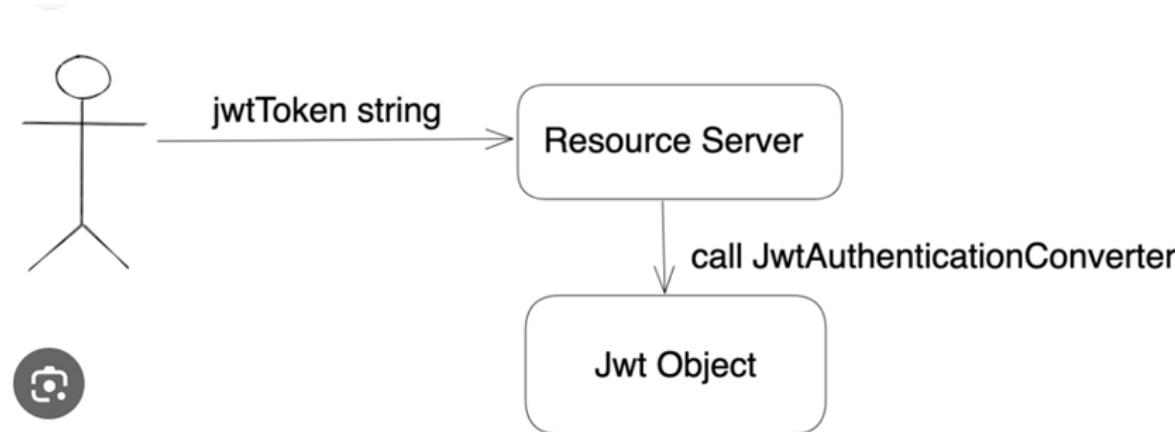
▶ OAuth 클라이언트 등록 (Register Client)

- OAuthServer가(권한 부여 서버) 클라이언트와 대화하기 위해 서버는 각 클라이언트에게 고유한 클라이언트 식별자를 할당해야 한다.
- 클라이언트의 정보를 서버의 구성 정보에 저장.
- 이 구성 정보에 클라이언트 인증 방법, 클라이언트의 권한 부여 타입 설정, 리다이렉트 URI 설정 클라이언트 scope 설정 및 객체 생성 및 속성 설정

▶ JSON Web Token (JWT)

- JWT 토큰은 권한 서버와 어떤 네트워크 api 호출 없이도 토큰 자체만으로 보호된 리소스에 접근 가능, 토큰에 보호된 리소스가 필요로 하는 정보를 토큰 정보에 담아 놓으면 됩니다.
- 권한 인가 서버는 그러한 정보를 토큰에 담고 그러한 정보가 담긴 토큰은 클라이언트에 전달이 되고 토큰의 구조가 불명확한 상태로 유지되기 때문에 클라이언트는 토큰의 내용을 알지 못합니다.
- 클라이언트는 그 토큰을 보호된 리소스에 전달하고 이때 보호된 리소스는 전달 받은 토큰의 내용을 알아야 합니다. 그래서 토큰을 파싱해 그 안에 담기 정보를 보고 접근을 허용할지 불허할지 판단합니다.

► Resorce Server



- 리소스 서버는 요청이 들어온 http 요청에서 OAuth 토큰을 추출해 그것을 파싱 및 검증하고 해당 토큰이 어떤 종류의 요청을 할 수 있는지 판단해야 합니다. 그러기 위해선 리소스 서버에서 Converter가 필요합니다.
- 토큰 권한 정보에 따라 후속 행위가 결정됨 즉 클라이언트가 각각 보호된 리소스 어디에 접근이 허용될지 결정이 됨 보통 scope 권한 범위에 따라 정합니다.

▶ RESTAPI

판매자번호	<u>seller_id</u>	PK	BIGINT	판매자 식별자
아이디(이메일)	<u>seller_email</u>	PK	VARCHAR	
비밀번호	<u>password</u>		VARCHAR	
이름	<u>seller_name</u>		VARCHAR	
핸드폰번호	<u>phone_number</u>		VARCHAR	
비즈니스형태	<u>business_type</u>	ENUM	{ 브랜드 오너/제조사, 위탁판매, 매입판매, 해외직구/병행수입 }	

1. OAuth2 Authorization Server에서 인가 코드 그랜트 타입을 사용한 코드를 Postman으로 운용 Test

The screenshot shows the Postman application interface. At the top, there's a header with 'HTTP' and 'OAuth2 / Authorization_code 얻기'. Below it, a 'GET' method is selected. The URL field contains the following URL with parameters highlighted by a blue box:

```
http://localhost:9090/oauth2/authorize?client_id=demo-client&response_type=code&redirect_uri=http://127.0.0.1:9191/login/oauth2/code/demo-client-oidc&scope=openid%20demo.read%20demo.write&state=STATE&code_challenge=6akqLtDVNzKsE7AxonsHGBQjHIYzyfQYRMy6iE1KXy&code_challenge_method=S256
```

Below the URL, there are tabs for 'Params' (selected) and 'Auth'. A 'Query Params' table is shown with the following data:

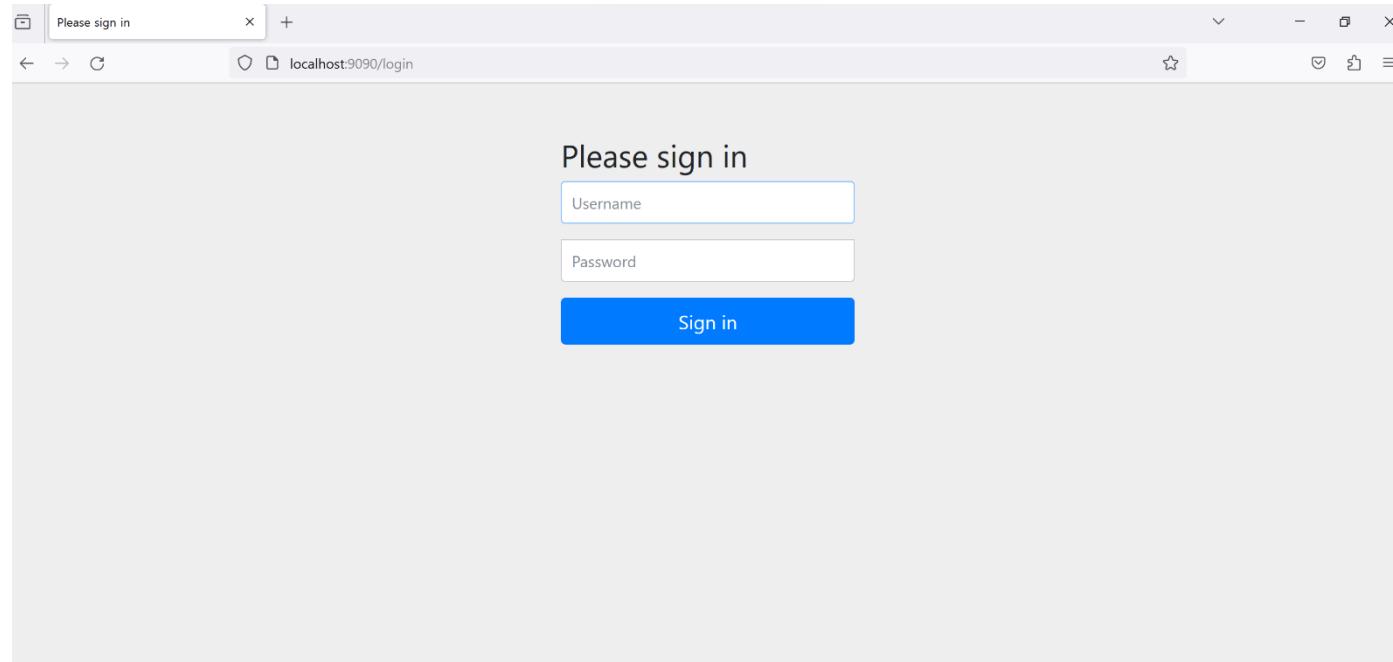
Key	Value	Description	... Bulk Edit
client_id	demo-client		
response_type	code		
redirect_uri	http://127.0.0.1:9191/login/oauth2/code/demo-clie...		
scope	openid%20demo.read%20demo.write		
state	STATE		
code_challenge	6akqLtDVNzKsE7AxonsHGBQjHIYzyfQYRMy6iE1IK...		
code_challenge_method	S256		

At the bottom of the table, there are empty rows for 'Key', 'Value', and 'Description'.

④ http://localhost:9090/oauth2/authorize?client_id=demo-client&response_type=code&redirect_uri=http://127.0.0.1:9191/login/oauth2/code/demo-client-oidc&scope=openid%20der

위 URL은 미인증 된 요청에 대해서 제공할 URL입니다.

```
http.exceptionHandling(exceptions -> exceptions
    .defaultAuthenticationEntryPointFor(
        new LoginUrlAuthenticationEntryPoint("/login"))
```



로그인 화면으로 이동이 되고 다음과 같이 인가 코드를 발급받는다.

ⓘ 127.0.0.1:9191/login/oauth2/code/demo-client-oidc?code=Lu4HRCrL4csuS0pfcopMLeVgzDQOhcSCFEfCDZFwzkuEQLWo7y7rgeSZL0T8I_5zUx_GPKq ☆

2. Access Token 발급하기

The screenshot shows the Postman application interface. At the top, it displays the URL `http://localhost:9090/oauth2/token`. Below the URL, there are tabs for **Params**, **Authorization**, **Headers (11)**, **Body** (which is currently selected), **Pre-request Script**, **Tests**, and **Settings**. On the right side, there are buttons for **Save**, **Send**, and other options.

The **Body** tab is set to **x-www-form-urlencoded**. The table below lists the parameters:

Key	Value	Description	Bulk Edit
grant_type	authorization_code		
code	Lu4HRCrL4csuS0pfcopMLeVgzDQOhcSCFEfCDZF wzkuEQLWo7y7rgeSzl0T8l_5zUx_GPKqIMj8fDRKh DkVtVPj7hYkRns56Aw-r7- ELxtBO5w5kCjEAFD0Psajiuzon		
redirect_uri			
client_id	demo-client		
client_secret	demo-client-secret		
code_verifier	nD1bnZS0SfvKS1ztg_Jhp4WyXy4omixGonKPZDo...		
Key	Value	Description	

Body Cookies Headers (11) Test Results

>Status: 200 OK Time: 466 ms Size: 1.78 KB Save as example ⋮

Pretty Raw Preview Visualize JSON ↻

1 {
2 "access_token": "eyJraWQiOiJhNWJlMWY1ZS03MGF1LTRiZmEtOGQ2NS1lNTU1YWU1NWRkYzMlCJhbGciOiJSUzI1NiJ9.
eyJzdWIiOiJ1c2VyIiwiYXVkiJoiZGVtby1jbGllbnQiLCJuYmYi0jE3MDEzMTQ2MzUsInNjb3BIjpBImlRlbW8ucmVhZCI
aWQixSwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdDo5MDkwIiwiZXhwIjoxNzAxMzE00TM1LCJpYXQi0jE3MDEzMTQ2MzV9.
ffa8dRLoQc9b0QIX1gSK15Rw2Gr-YWt8uTHbfvglttJnhYaq5fKAIxRpeof0Vs6WdAWEL4zwM5bXbQSqhwlojuWE1hbLD9XzQApG4KNVmXXknG8H-IYaCIgT
Ss2ciZ42ayZ-2Wl_Lo0ZsF6LEF1aS3FchvJp-BCskD_MQDwFBOEojkGLrRNan80vv35EdEAdIwdfFCKpgSA3s1KD1mEyB-bcTS1wg18Suy-PMf9LkKXxbP6p
lwCIBdSVuNBiPCfk96kgxjCzFwdl2YMGm25xPDNy18pKYVTVFQHrVnStHbtYdYUtbhv_t-kysDGjQtfHnuqz3J3QtM-mqrYZ5Q84A",
3 "refresh_token":
4 "6jXTdiNWD_bBaL0GUIRKIHRE0_ST_2LG-sRqQ4F107sL0213hhh7NcGyGftfVJv8nKA2ouuduMv_07LJuPG7PQM1XL55vruuuqeYI_DB9Wh4gluLMCLhc00
W11Bo7w7k",
5 "scope": "demo.read demo.write openid",
6 "id_token": "eyJraWQiOiJhNWJlMWY1ZS03MGF1LTRiZmEtOGQ2NS1lNTU1YWU1NWRkYzMlCJhbGciOiJSUzI1NiJ9.
eyJzdWIiOiJ1c2VyIiwiYXVkiJoiZGVtby1jbGllbnQiLCJhenAiOiJkZW1vLWNsaWVudCI
sImlzcyI6Imh0dHA6Ly9sb2NhbGhvc3Q6OTA5MCIsImV4cCI6
MTcwMTMxNjQzNSwiaWF0IjoxNzAxMzE0NjM1fQ.
sxRBwJ_U-OYnS_NsOGxxx858hWsTiWEmUDjtWFTJJVs7-GrUIij2LPBbPKcRtw418CSh6pfFWL6hVJnAQjQmZPRKoog7CJHUVACAYHhLRockWw4pxG7aMxXW
V2ZyUfqCNP_Yu_bn4FGHAbfzA1ZGcBVqY-3IUd92Cpt0K1Xow6JI60XGy57mvBu7ftk6bkXgCv7cr7r410vUq5cf1Sej2qTSVxW_2S2U4kEuwmH9yi0NYQc5
ttFxMX1oSRKB6L1WgNZkIYJa9vy0J85U1GmV0CVygVMuHVjmKg3z6zpAddfv56D8fgXInDxw7ZRrVJIRjsaDcIDIIdMI
dU9q7t1YGcw",
7 "token_type": "Bearer",
8 "expires_in": 300

3. Refresh Token 요청하기

```
"refresh_token":  
    "6jXTdiNWD_bBaL0GUiRKIHREO_ST_2LG-sRqQ4F107sL0213hhh7NcGyGftfVJv8nKA2ouuduMv_07LJuPG7PQM1XL55vruuuqeYI_DB9Wh4gluLMCLHc00  
    W11Bo7w7k",
```

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:9090/oauth2/token
- Body (form-data):**

grant_type	refresh_token
client_id	demo-client
refresh_token	6jXTdiNWD_bBaL0GUiRKIHREO_ST_2LG-sRqQ4F107sL0213hhh7NcGyGftfVJv8nKA2ouuduMv_07LJuPG7PQM1XL55vruuuqeYI_DB9Wh4gluLMCLHc00 W11Bo7w7k
client_secret	demo-client-secret
- Response Body (Pretty JSON):**

```
1  "access_token": "eyJraWQiOiJhNWJlMWY1ZS03MGFlLTRiZmEtOGQ2NS1lNTU1YWU1NWRkYzMlCJhbGciOiJSUzI1NiJ9.  
eyJzdWIiOiJc2ViIwiYXvkIjoiZGVtby1jbG1bnQiLCJuYmYj0jE3MDEzM04DksInNjb3B1IpbImRlbW8ucmVhZCIIsImRlbW8ud3JpdGUiLCJvcGVu  
aWQiXSwiaXnzIjoiaHR0cDovL2xvY2FsaG9zdD5MDkwIiwiXhwIjoxNzAxMzE1MTg5LCJpYXQiOjE3MDEzM04D19.  
BWu7vtT03tuEcXb21Enw8DwpuavW-HdJnqbzDyCPkCFKwudzE69sVtqM2AKVD8Nq9SarigPFAkeE5Tx782et0mpB7ph-58ayaajCK2bkusKf-xCIrF_xl  
CeRaCRWf5Xj1oc2HaoCo3rF1E5BnHsXs_i0VTDdedAkomz6GsDbThwq12TSs0PXcPaOoJ7AZATKIFd-Jz0N7BLQE0Yp46zoJzXkdUFKcXifRGF  
s109T50pSZXrZXjicreq4So-azgaf994na0NYVYUAooyHxviZl1X091f9URZDNeKvLezeHXfmhjhSpNrLLntE9aLMouX5DYfqzGQ",  
"refresh_token":  
    "6jXTdiNWD_bBaL0GUiRKIHREO_ST_2LG-sRqQ4F107sL0213hhh7NcGyGftfVJv8nKA2ouuduMv_07LJuPG7PQM1XL55vruuuqeYI_DB9Wh4gluLMCLHc00  
    W11Bo7w7k",  
"scope": "demo.read demo.write openid",  
"id_token": "eyJraWQiOiJhNWJlMWY1ZS03MGFlLTRiZmEtOGQ2NS1lNTU1YWU1NWRkYzMlCJhbGciOiJSUzI1NiJ9.  
eyJzdWIiOiJc2ViIwiYXvkIjoiZGVtby1jbG1bnQiLCJhenAiOjkZw1vLWnsawVuDCIsImLzcyl6Imh0dHA6Ly9sb2Nhbgvc3Q60TA5MCIsImV4cCI6  
MTcwMTMxNjY40SwiaWF0IjoxNzAxMzE00Dg5fQ.  
U_166Fgyb3b61kYB_WwJy8Mw6T30gk8cg0m0Vo2KfKZV1R9KoSsPPPhQtIRQG6Wp8H37UC966sDR16fnGMyn98p_o08sPiDvW5ndFTKFAvuz9ULhcTC  
JKzkv03EGUPsYl_PRnfEwAv1DKuJPMxhxQTdg0NgxHHkeCkQpwWjnnuqNmR0A3NvkT7j2XyyFDVjoi-dsyzqCnjMo66kLdXkifqf1GjEkoQeGtIaDhBWJ92  
HC03g5qkBmQetm2Dd1b2iAWFc7kGOX7Xdxg8y2fLdhjaYlhs-wqm0e2C4mgnmm2yTGYw2cBuxRpj0AvxuNc4Sy5j0zPyv1KhNLSA",  
"token_type": "Bearer",  
"expires_in": 299
```

Refresh Token 요청, 새로운 Access Token 발급

4. UserInfo 엔드포인트

The screenshot shows the Postman interface with the following details:

- Header Bar:** Shows various API endpoints: GET Authorization_code, GET Open id Configuration, GET userinfo (selected), POST GetAccessToken, POST GetRefreshToken, and others.
- Request Section:**
 - Method: GET
 - URL: http://localhost:9090/userinfo
 - Authorization: Bearer Token (selected)
 - Headers (8): A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)".
 - Body: Empty
 - Pre-request Script: Empty
 - Tests: Empty
 - Settings: Empty
- Response Section:**
 - Status: 200 OK
 - Time: 41 ms
 - Size: 348 B
 - Save as example
- Body Tab:** Displays the JSON response:

```
1 {  
2   "sub": "user"  
3 }
```

5. logout 요청

HTTP OAuth2 / logout

Save |  

GET http://localhost:9090/connect/logout?id_token_hint=eyJraWQiOiJhNWJIMWY1ZS03MGFILTRiZmEtOGQ2NS1INTU1YWU1NWRkYzN... Send

Params • Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	id_token_hint	eyJraWQiOiJhNWJIMWY1ZS03MGFILTRiZmEtOGQ2NS1INTU1YWU1NWRkYzN...			
<input checked="" type="checkbox"/>	client_id	demo-client			
<input checked="" type="checkbox"/>	post_logout_redirect_uri	http://127.0.0.1:9191			
	Key	Value	Description		

Response

Could not send request

Error: connect ECONNREFUSED 127.0.0.1:9191 | [View in Console](#)

[Learn more about troubleshooting API requests](#)



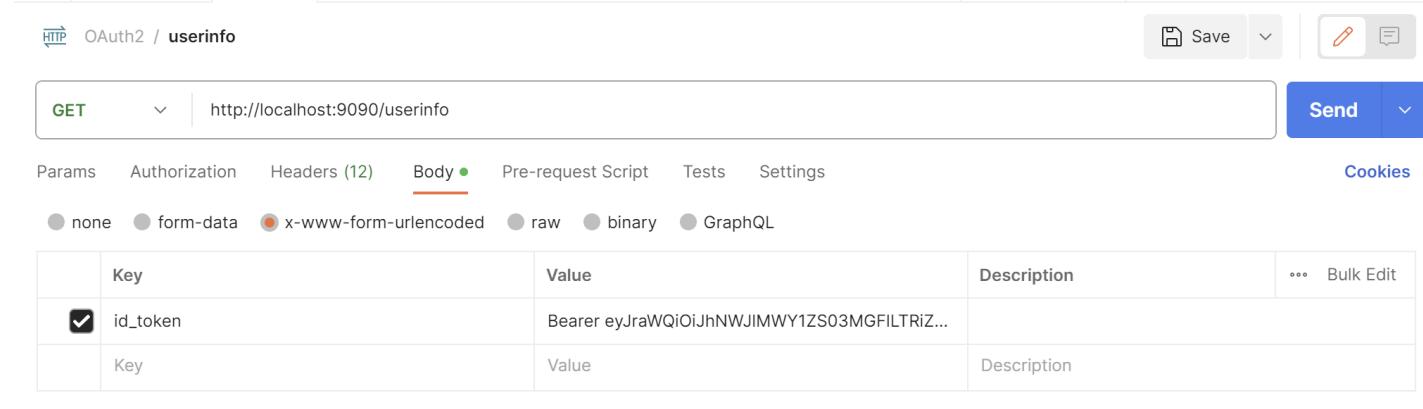
6. Token 삭제

The screenshot shows the Postman application interface. At the top, there is a header bar with the text "OAuth2 / revoke : 토큰 삭제". On the right side of the header are "Save", "Edit", and "Send" buttons. Below the header, the method is set to "POST" and the URL is "http://localhost:9090/oauth2/revoke".
The "Body" tab is selected, showing a "form-data" structure with three fields:

- token: eyJRaWQiOiJhNWJIMWY1ZS03MGFILTRiZmEtOGQ...
- client_id: demo-client
- client_secret: demo-client-secret

Below the body, there are tabs for "Params", "Authorization", "Headers (10)", "Pre-request Script", "Tests", "Settings", and "Cookies".
At the bottom, there are tabs for "Body", "Cookies (1)", "Headers (10)", and "Test Results". The "Body" tab is selected. It shows a "Pretty" view of the JSON data sent to the server. The status bar at the bottom indicates "Status: 200 OK", "Time: 83 ms", and "Size: 293 B".

7. Token 삭제확인



The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:9090/userinfo
- Body:** x-www-form-urlencoded
- Params:** id_token (Value: Bearer eyJraWQiOiJhNWJIMWY1ZS03MGFILTRiZ...)

Test Results:

- Status: 200 OK
- Time: 5 ms
- Size: 1.62 KB

Body (Pretty):

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7   <meta name="description" content="">
8   <meta name="author" content="">
9   <title>Please sign in</title>
10  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/css/bootstrap.min.css" rel="stylesheet"
11      integrity="sha384-Y6pD6FV/Vv2HJnA6t+vslU6fwYXjCFtcEpHbNJ0lyAFsXTsjBbfDjzALEQsN6M" crossorigin="anonymous">
12  <link href="https://getbootstrap.com/docs/4.0/examples/signin/signin.css" rel="stylesheet"
13      crossorigin="anonymous" />
14 </head>
```

Review

▶ Review ERD

상품 리뷰 정보

PK	리뷰번호	review_id	리뷰 식별자	INT	NOT NULL	Default value	Comment
제목	title	Domain	VARCHAR(255)	NOT NULL	Default value	Comment	
내용	content	Domain	TEXT	NULL	Default value	Comment	
별점	star	Domain	INT	NOT NULL	Default value	Comment	
좋아요	liked	Domain	Type	NULL	Default value	Comment	
생성일	created_date	Domain	DATE	NOT NULL	Default value	Comment	
수정일	modified_date	Domain	DATE	NOT NULL	Default value	Comment	
상품ID	item_id	상품기본정보 PK	INT	NOT NULL	Default value	Comment	
회원번호	member_id	식별자	INT	NOT NULL	Default value	AUTO_INCREMENT 사용	
이메일	member_email	로그인시 아이디	VARCHAR(30)	NOT NULL	Default value	이메일 형식에 맞게 작성	

+
↓
☰

리뷰 이미지 파일

PK	이미지파일ID	review_image_file_id	이미지파일 PK	INT	NOT NULL	Default value	AUTO_INCREMENT 사용
이미지파일용도	image_type	Domain	VARCHAR(255)	NOT NULL	Default value	{ main : sub }	
이미지 파일 원본 이름	original_image_name	Domain	VARCHAR(255)	NOT NULL	Default value	Comment	
이미지 파일 저장 이름	save_image_name	Domain	VARCHAR(255)	NOT NULL	Default value	중복을 막기 위해 랜덤으로 생성	
이미지 파일 크기	image_size	Domain	VARCHAR(255)	NOT NULL	Default value	Comment	
S3에서 관리할 key	image_key	Domain	VARCHAR(255)	NOT NULL	Default value	AWS S3에서 관리할 KEY	
S3에 저장된 경로	image_path	Domain	VARCHAR(255)	NOT NULL	Default value	AWS S3 저장소의 파일에 접근하는 URL	
리뷰번호	review_id	리뷰식별자	INT	NOT NULL	Default value	Comment	

1. 리뷰 작성

- 리뷰를 작성하기 위해서 로그인이 필요하다.
- 리뷰 데이터에는 텍스트, 이미지가 포함된다.
- 별점 부여
- 리뷰 작성 날짜, 회원이름 - 회원이름 중간자는 표시되지 않아야 한다, 도움이 돼요, 안 돼요

2. 리뷰 읽기/조회

- 모든 리뷰를 읽을 수 있어야 한다.
- 회원이름 클릭 시 특정 회원이 작성한 리뷰를 모아서 볼 수 있어야 한다.
- 회원 정보에는 해당 리뷰어의 도움 개수와 작성 개수가 나타난다.
- 정렬 : 베스트 순, 최신 순
- 별점 통계 별로 리뷰를 볼 수 있어야 한다. (최고~나쁨)

3. 리뷰 관리

- 작성자 본인만 수정하거나 삭제할 수 있어야 한다.
- 판매자가 부적절한 리뷰를 삭제할 수 있어야 한다.

method	권한	uri	action
GET	user	/api/v1/item/{reviewId}	사용자가 상품 목록을 조회한다 (상품 상세 페이지)
GET	user	/api/v1/mypage/review	사용자가 리뷰 목록을 조회한다 (마이 쿠팡 페이지)
POST	user	/api/v1/mypage/orderlist/reviewpost	사용자가 리뷰를 작성한다
PUT	user	/api/v1/mypage/review/{reviewId}	사용자가 리뷰를 수정한다
DELETE	user	/api/v1/mypage/review/{reviewId}	사용자가 리뷰를 삭제한다

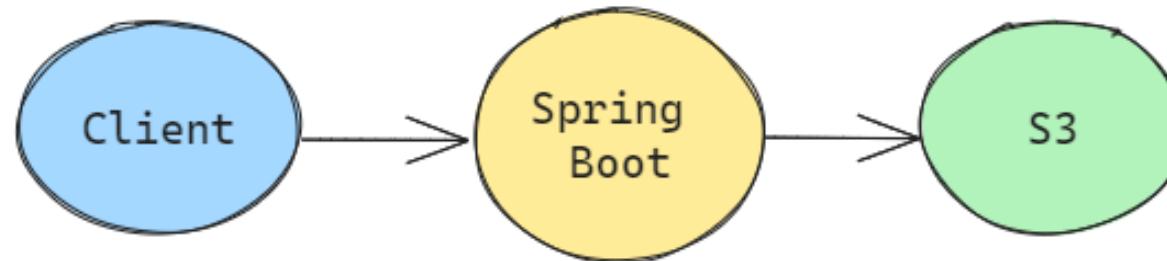
1. 상품별 리뷰

- Review 번호, 제목, 내용, 별점, 좋아요, 생성일, 수정일, 상품ID, 회원번호, 이메일

2. 마이페이지 리뷰 (리뷰관리)

- 이미지 파일 ID, 이미지 파일 용도, 이미지 파일 원본 이름, 이미지 파일 저장 이름, 이미지 파일 크기, S3에서 관리할 key, S3에 저장할 경로, 리뷰 번호

▶ S3 Image Upload



1. client에서 첨부파일을 첨부하면 spring에서 제공하고 있는 MultipartFile 인터페이스를 이용하여 파일을 업로드(서버에 전송) 한다.

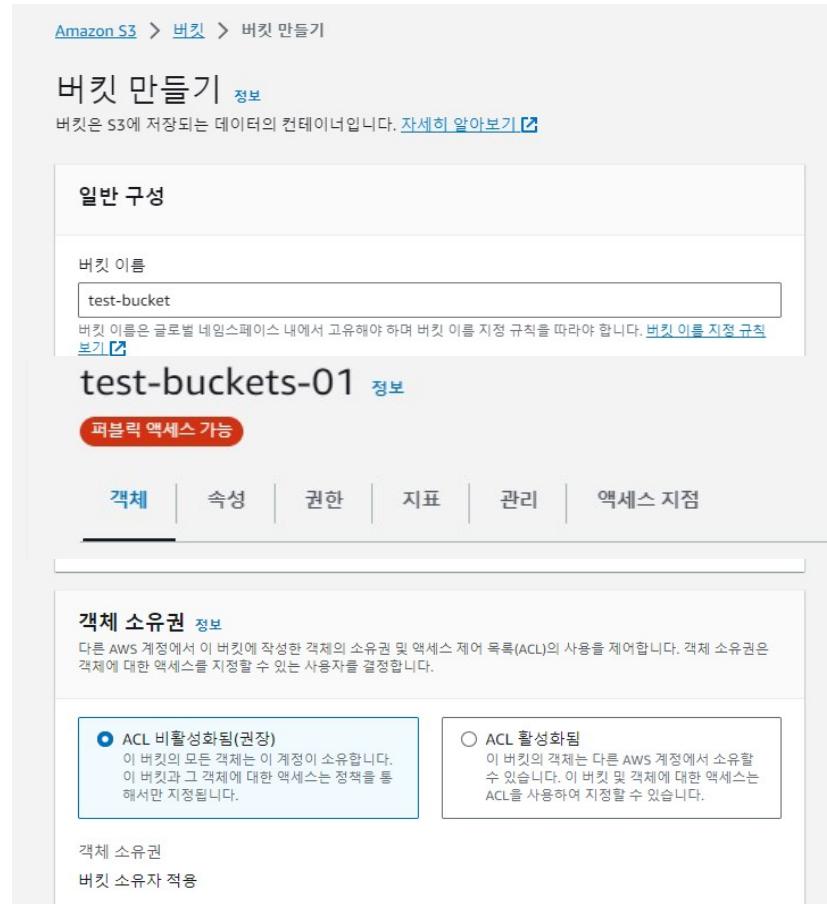
(MultipartFile -> 파일의 이름, 크기, 내용과 같은 업로드된 파일의 내용에 엑세스하는 방법을 제공)

2 .MultipartFile을 객체로 변환한다.

3. 스트림 데이터로 변환된 객체를 s3 bucket에 업로드한다.

4. s3에 저장된 객체 url DB imagePath에 저장한다.

▶ S3 Image Bucket



1. S3 버킷 생성

- 퍼블릭 액세스가 가능하도록 AWS Policy Generator에서 버킷 정책을 편집 한다.
- IAM 사용자 등록한다.
- S3 사용 권한을 부여하기 위해 AmazonS3FullAccess를 검색해서 체크한다.

2. build.gradle 설정

3. aws.properties 파일 설정

(accesskey, secretkey 보안을 위해 .gitignore 파일에 등록)

4. FileUploadController 설정

5. 이미지 리사이징 : 효율적인 서버 운영을 위해서 사용한다.

aws lambda - 내가 요청받고 처리하는 서버가 없는 대신 AWS 서버가 일을 처리해준다. 사용한 시간에 한해 비용을 지불한다.

▶ S3 Image Upload Resizing

- S3 스토리지에 있는 이미지 파일들을 한 개당 5MB의 이미지를 최대 100개의 파일로 업로드하기
원본 사이즈의 이미지를 그대로 올리게 되면, 로딩이 늦어지게 된다.
따라서, 빠른 이미지의 로딩을 위해서는 이미지 리사이징이 필요하다.
- 이미지 사이즈를 줄이려면, 서버에서 직접 이미지 크기를 조정하고, Serverless Lambda를 활용한다.
- 다수의 이미지 파일 저장

단일 이미지와 다른 것 : form 태그의 **multipart/form-data**

다수 이미지 형태의 이미지를 보내려면 통신할 때 헤더에 multipart/form-data가 추가되어야 한다.

CI

Continuous Integration (CI): 코드 변경을 지속적으로 통합하여 코드 품질 향상과 충돌 방지 효과를 목표로 하는 소프트웨어 방법론

CI의 중요성

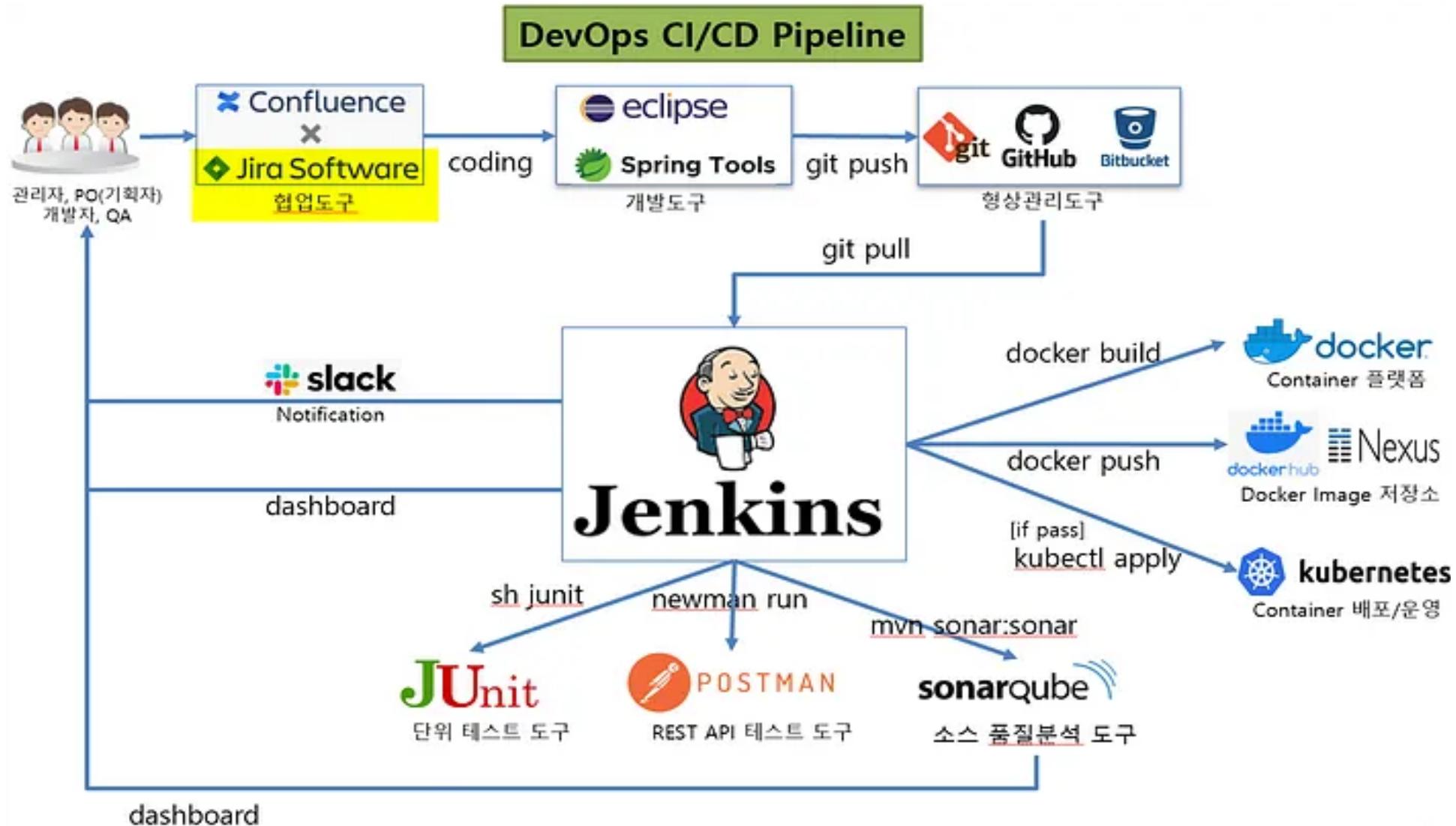
- 개발 효율 증대:** 변경 사항을 빠르게 통합하여 개발자들이 신속하게 협업할 수 있습니다.
- 품질 향상:** 자동화된 빌드와 테스트를 통해 소프트웨어 품질을 유지하고 향상시킵니다.

젠킨스 소개

- **Jenkins (jenkins)**: CI를 구현하기 위한 다양한 기능을 제공하는 오픈 소스 자동화 도구
 - 뛰어난 확장성: 플러그인 아키텍처를 통해 다양한 환경과 도구에 대한 지속적 통합을 쉽게 구현

젠킨스의 역할

- **자동화된 빌드**: 변경 사항을 자동으로 빌드하여 일관된 환경에서 작업할 수 있도록 지원합니다.
- **자동화된 테스트**: 다양한 테스트를 자동으로 실행하여 코드 변경에 따른 오류를 미리 감지하고 예방합니다.
- **지속적인 통합**: 변경 사항을 지속적으로 통합하여 개발자 간 협업을 원활하게 합니다.
- **자동 배포**: 빌드 및 테스트가 성공하면 자동으로 배포되어 개발 주기를 단축하고 신뢰성을 높입니다.



01 프로젝트 개요

CI

1. aws ec2 우분투에 젠킨스 설치하기

Linux

Jenkins – an open source automation server which enables developers around the world to reliably build, test, and deploy their software

<https://www.jenkins.io/doc/book/installing/linux/>

 Jenkins

 젠킨스 설치 전 로컬 머신에 **openjdk-17-jre** 설치를 먼저 진행해야한다

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

[Continue](#)

 - sudo cat {위의 경로} → 입력 시 나오는 password를 칸에 입력하기
- 설치 오류 발생 시 공식 홈페이지에서 버전 업데이트 확인

01 프로젝트 개요

CI

b. Jenkins Github Servers 설정

- Jenkins 관리 -> 시스템 설정 -> Github Servers

Settings / Developer settings

The screenshot shows the GitHub 'Developer settings' page under 'Personal access tokens'. A red box highlights the 'Personal access tokens' link in the sidebar. The main content area displays instructions for building GitHub Apps and a large black button labeled '- Generate new token'.

GitHub Apps

New GitHub App

OAuth Apps

Personal access tokens

Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing on the GitHub API. You can also read more about building GitHub Apps in our developer documentation.

- Generate new token

Settings / Developer settings

The screenshot shows the GitHub 'Personal access tokens' page. A red box highlights the 'Generate new token' button in the top right corner. The sidebar shows 'GitHub Apps' and 'OAuth Apps' links, with 'Personal access tokens' being the active tab.

GitHub Apps

OAuth Apps

Personal access tokens

Personal access tokens

Generate new token

Revoke all

01 프로젝트 개요

CI

b. Jenkins Github Servers 설정

- Jenkins 관리 -> 시스템 설정 -> Github Servers

GitHub

GitHub Servers

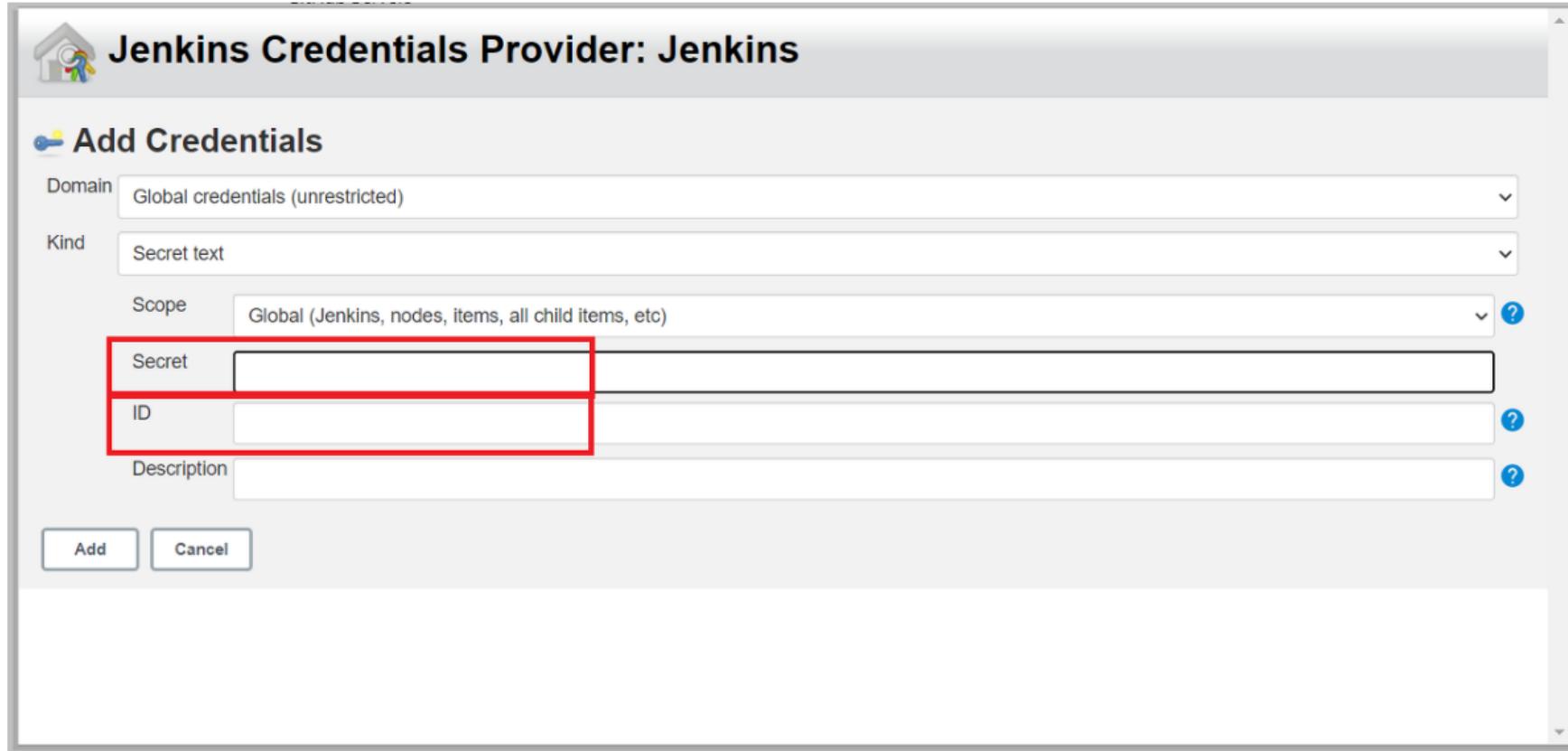
GitHub Server	
Name	<input type="text"/>
API URL	<input type="text" value="https://api.github.com"/>
Credentials	<input type="button" value="- none -"/> <input type="button" value="Add"/>
<input type="button" value="Test connection"/>	
<input type="checkbox"/> Manage hooks	
<input type="button" value="고급..."/>	
<input type="button" value="삭제"/>	
<input type="button" value="Add GitHub Server ▾"/>	
<input type="button" value="고급..."/>	

01 프로젝트 개요

CI

b. Jenkins Github Servers 설정

- Jenkins 관리 -> 시스템 설정 -> Github Servers



- Secret : 위에서 생성한 github access token 입력
- ID : 식별자(원하는 값 입력)

01 프로젝트 개요

CI

c. Jenkins 파이프라인 github 관련 설정

The screenshot shows the Jenkins General configuration page for a project. The top navigation bar includes tabs for General, 소스 코드 관리, 빌드 유발, 빌드 환경, Build, and 빌드 후 조치. The General tab is selected.

The configuration fields include:

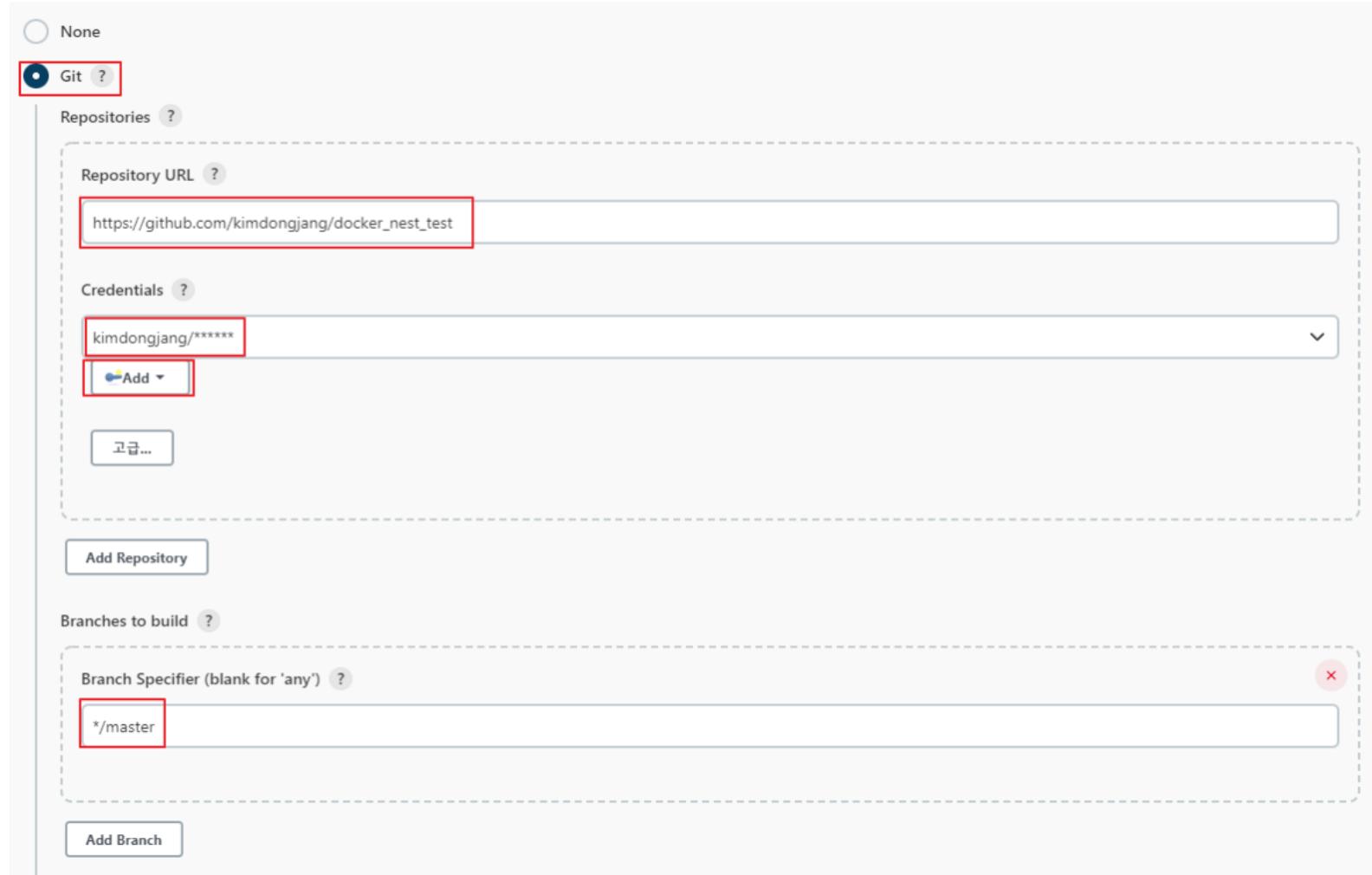
- 설명**: A large text area with a placeholder "[Plain text] 미리보기".
- GitHub project**: A checked checkbox. Below it is a "Project url" field containing the URL `https://github.com/kimdongjang/docker_nest_test/`, which is highlighted with a red rectangle.
- Advanced Options**: Several checkboxes for advanced build configurations:
 - This build requires lockable resources
 - Throttle builds
 - 오래된 빌드 삭제
 - 이 빌드는 매개변수가 있습니다
 - 빌드 안함
 - 필요한 경우 concurrent 빌드 실행
- Buttons**: Two "고급..." buttons at the bottom.

// `https://github.com/{본인 깃허브 유저네임}/{본인 깃허브 프로젝트}/`

01 프로젝트 개요

CI

c. Jenkins 파이프라인 github 관련 설정

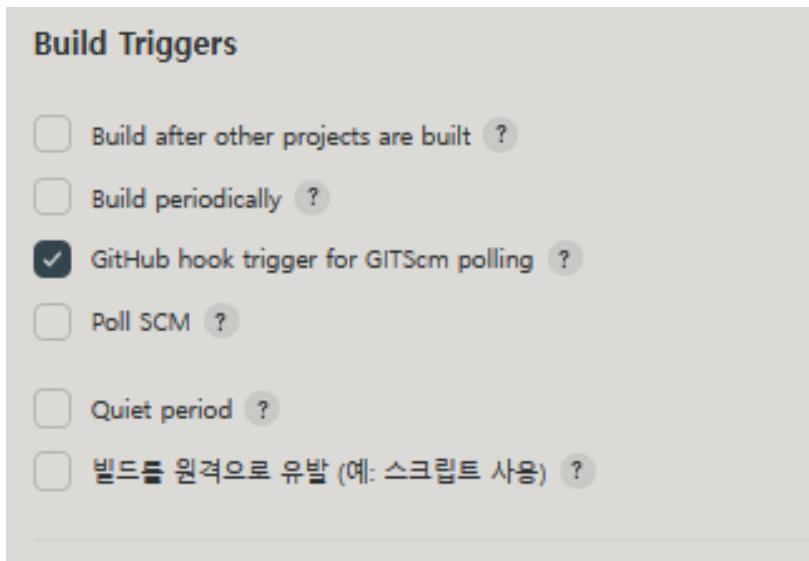


// 프로젝트가 Private일 경우 Credentials kind를 Username with Password로 설정하기

01 프로젝트 개요

CI

c. Jenkins 파이프라인 github 관련 설정



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

Content type

Secret

Which events would you like to trigger this webhook?

Just the push event.

Send me everything.

Let me select individual events.

Active
We will deliver event details when this hook is triggered.

Add webhook

01 프로젝트 개요

CI

3. AWS ECR 설정

a. ECR Repository 생성

The screenshot shows the AWS ECR console interface. On the left, there's a sidebar with navigation links: 'Amazon Elastic Container Registry', 'Private registry', 'Public registry', 'Repositories', 'ECR public gallery', 'Amazon ECS', 'Amazon EKS', 'Getting started', and 'Documentation'. The main area has a breadcrumb trail: 'Amazon ECR > 리포지토리'. It shows two tabs: 'Private' (selected) and 'Public'. Below this, a table lists a single repository:

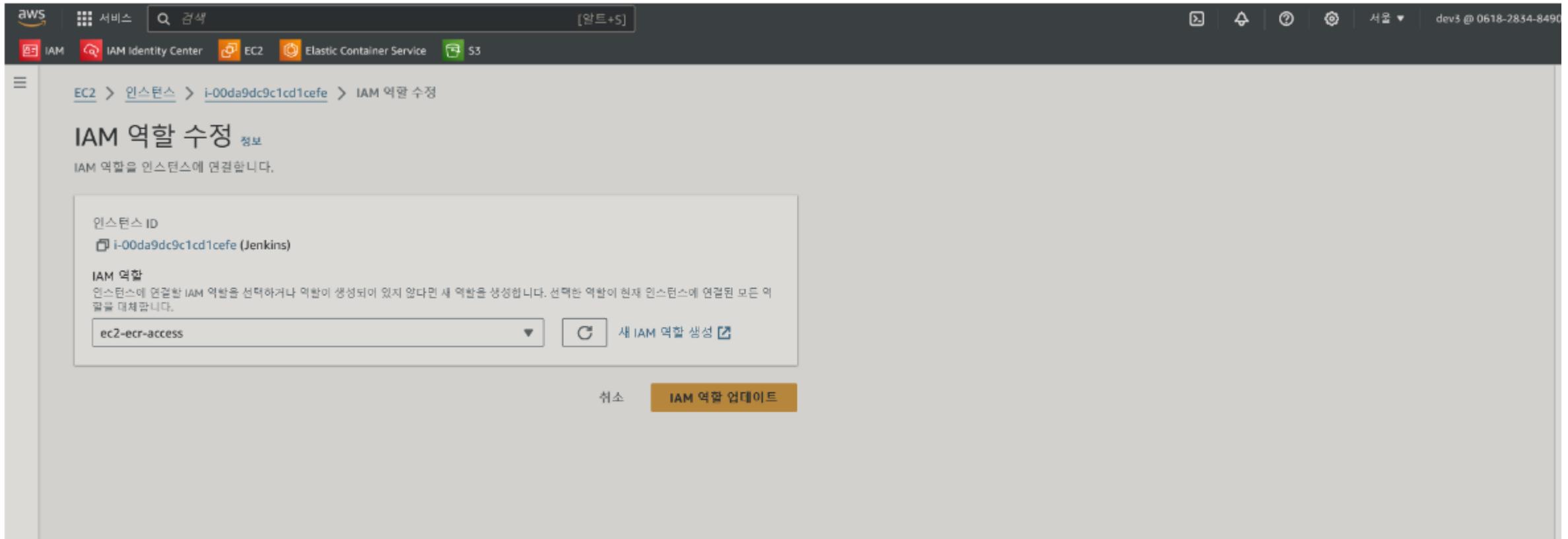
리포지토리 이름	URI	생성 날짜	태그 변경 불가능	스캔 빈도	암호화 유형	풀스루 케시
gopang	061828348490.dkr.ecr.ap-northeast-2.amazonaws.com/gopang	2023년 11월 28일, 14:19:49 (UTC+09)	비활성화됨	수동	AES-256	비활성

01 프로젝트 개요

CI

3. AWS ECR 설정

b. IAM Role 생성 후 ECR 사용을 위한 권한 설정



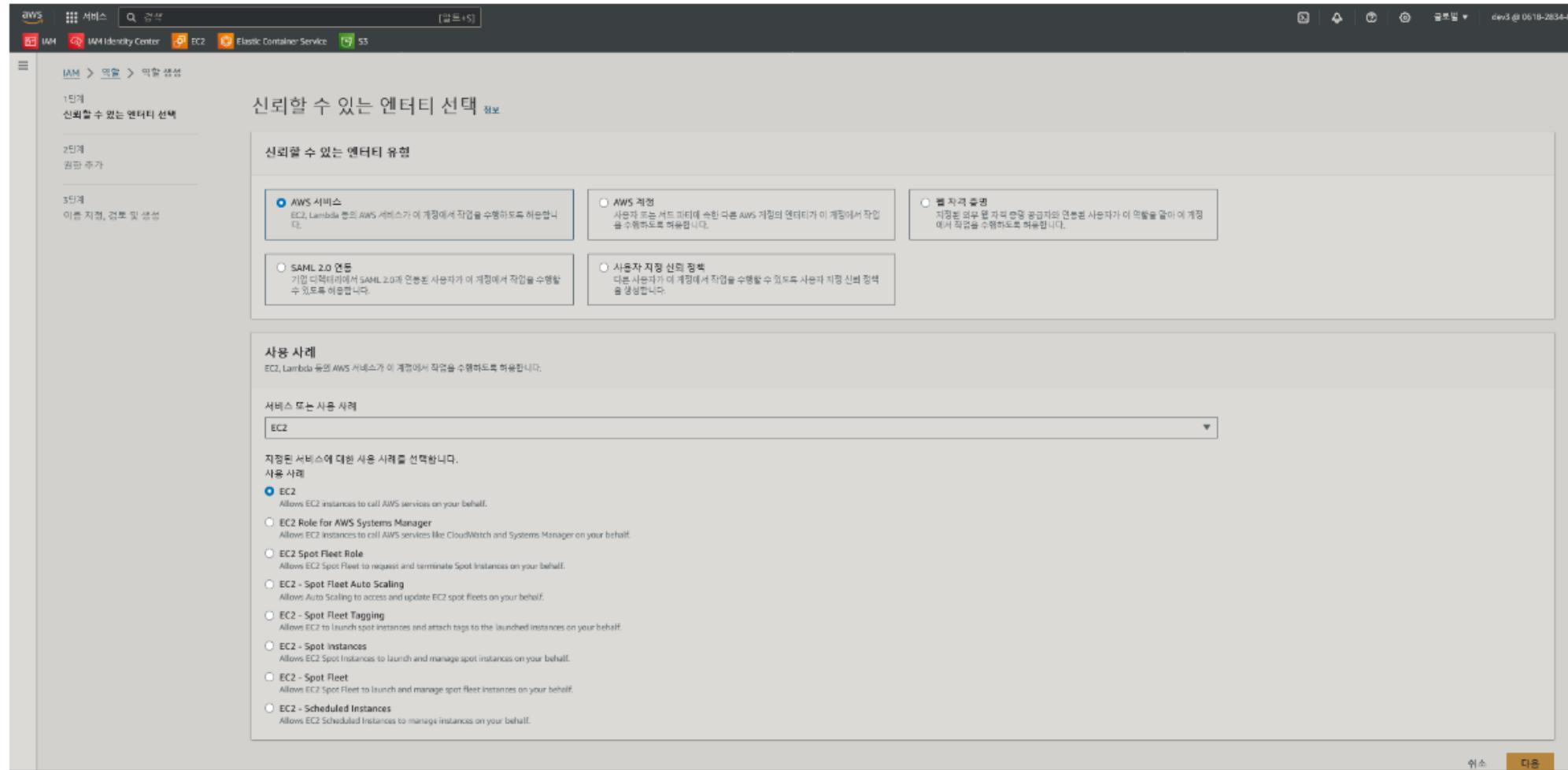
* 새 IAM 역할 생성 클릭

01 프로젝트 개요

CI

3. AWS ECR 설정

b. IAM Role 생성 후 ECR 사용을 위한 권한 설정



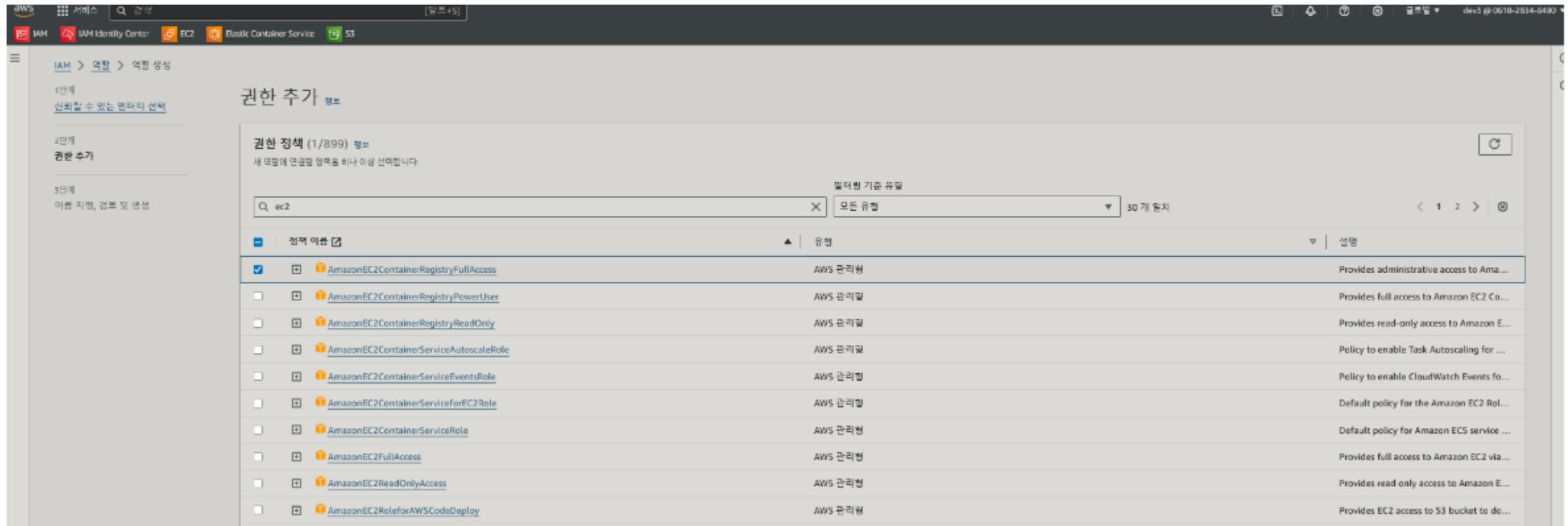
* EC2

01 프로젝트 개요

CI

3. AWS ECR 설정

b. IAM Role 생성 후 ECR 사용을 위한 권한 설정



The screenshot shows the AWS IAM Policy Editor interface. On the left, there's a sidebar with three sections: '1단계' (선택할 수 있는 엔터티 선택), '2단계' (권한 추가), and '3단계' (이름 지정, 검토 및 생성). The main area is titled '권한 추가' (Add Permissions) and displays a search bar with 'ec2' and a dropdown menu set to '모든 유형' (All types). Below the search bar is a table listing various AWS managed policies. One policy, 'AmazonEC2ContainerRegistryFullAccess', is selected and highlighted with a blue border. The table columns include '정책 이름' (Policy Name), '설명' (Description), and '설명' (Description).

정책 이름	설명	설명
<input checked="" type="checkbox"/> AmazonEC2ContainerRegistryFullAccess	AWS 관리형	Provides administrative access to Amaz...
<input type="checkbox"/> AmazonEC2ContainerRegistryPowerUser	AWS 관리형	Provides full access to Amazon EC2 Co...
<input type="checkbox"/> AmazonEC2ContainerRegistryReadOnly	AWS 관리형	Provides read-only access to Amazon E...
<input type="checkbox"/> AmazonEC2ContainerServiceAutoscaleRole	AWS 관리형	Policy to enable Task AutoScaling for ...
<input type="checkbox"/> AmazonEC2ContainerServiceEventsRole	AWS 관리형	Policy to enable CloudWatch Events fo...
<input type="checkbox"/> AmazonEC2ContainerServiceForEC2Role	AWS 관리형	Default policy for the Amazon EC2 Rol...
<input type="checkbox"/> AmazonEC2ContainerServiceRole	AWS 관리형	Default policy for Amazon ECS service ...
<input type="checkbox"/> AmazonEC2FullAccess	AWS 관리형	Provides full access to Amazon EC2 via...
<input type="checkbox"/> AmazonEC2ReadOnlyAccess	AWS 관리형	Provides read only access to Amazon E...
<input type="checkbox"/> AmazonEC2RoleforAWSCodeDeploy	AWS 관리형	Provides EC2 access to S3 bucket to do...

* AmazonEC2ContainerRegistryFullAccess

01 프로젝트 개요

CI

3. AWS ECR 설정

c. AWS Credentials 생성

The screenshot shows the Jenkins 'New credentials' configuration page for creating an AWS Credential. The 'Kind' dropdown is set to 'AWS Credentials'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' field is empty. The 'Description' field is empty. The 'Access Key ID' and 'Secret Access Key' fields are also empty. Under 'IAM Role Support', there is a dropdown menu currently set to '고급'. At the bottom is a blue 'Create' button.



- ID : 식별 값(원하는 값 입력)
- Description: 설명(필수X)
- Access Key ID: 발급 받은 IAM Access Key 입력
- Secret Access Key: 발급 받은 IAM Secret Access Key 입력

4. Jenkins Pipeline Script

- 사전 설정



- 스크립트 실행 전 로컬 머신에 설치해야 할 도구

1. git
2. gradle
3. docker
5. awscli & aws configure 설정

- 스크립트 실행 전 로컬 머신에 설치해야 할 플러그인

1. Docker
2. Docker Pipeline
3. Amazon ECR plugin

01 프로젝트 개요

CI

4. Jenkins Pipeline Script

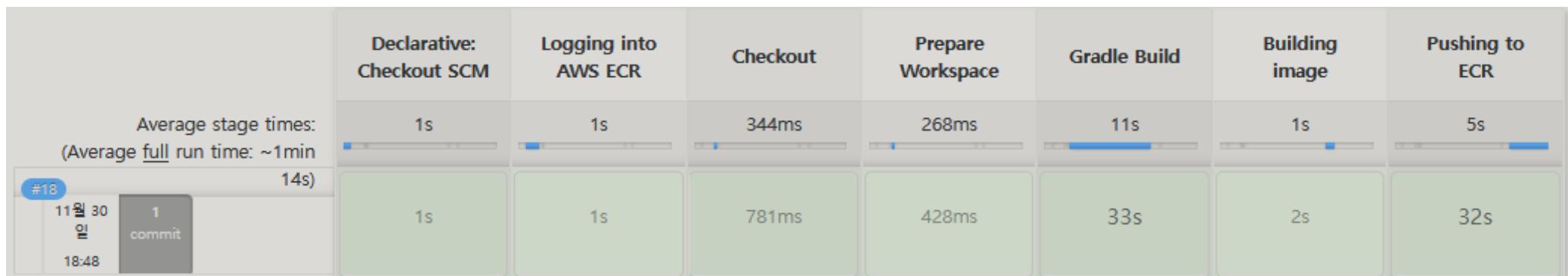
```
pipeline {
    agent any
    environment {
        AWS_ACCOUNT_ID="061828348490"
        AWS_DEFAULT_REGION="ap-northeast-2"
        IMAGE_REPO_NAME="gopang"
        IMAGE_TAG="v1"
        REPOSITORY_URI = "061828348490.dkr.ecr.ap-northeast-2.amazonaws.com/gopang"
    }
    stages {
        stage('Logging into AWS ECR') {
            steps {
                script {
                    sh """aws ecr get-login-password --region ${AWS_DEFAULT_REGION} | docker login --username AWS --password-stdin ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_DEFAULT_REGION}.amazonaws.com"""
                }
            }
        }
        stage('Checkout') {
            steps {
                git branch: 'main',
                credentialsId: 'gopang',
                url: 'https://github.com/JavaBrewer/geo-test1.git'
            }
        }
        stage('Prepare Workspace') {
            steps {
                script {
                    // Gradle Wrapper execution permission
                    sh 'chmod +x gradlew'
                }
            }
        }
        stage('Gradle Build') {
            steps {
                script {
                    // Run Gradle build
                    sh './gradlew clean build'
                }
            }
        }
        // Building Docker images
        stage('Building image') {
            steps{
                script {
                    dockerImage = docker.build "${IMAGE_REPO_NAME}:${IMAGE_TAG}"
                }
            }
        }
        // Uploading Docker images into AWS ECR
        stage('Pushing to ECR') {
            steps{
                script {
                    sh """docker tag ${IMAGE_REPO_NAME}:${IMAGE_TAG} ${REPOSITORY_URI}:$IMAGE_TAG"""
                    sh """docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_DEFAULT_REGION}.amazonaws.com/${IMAGE_REPO_NAME}:${IMAGE_TAG}"""
                }
            }
        }
    }
}
```

01 프로젝트 개요

CI

4. Jenkins Pipeline Script

- 실행 확인



Amazon Elastic Container Registry

Private registry Public registry Repositories Summary Images Permissions Lifecycle Policy Repository tags

ECR public gallery Amazon ECS Amazon EKS

Getting started Documentation

Amazon ECR > 리포지토리 > gopang

gopang

이미지 (1)

선택	이미지 태그	아티팩트 유형	푸시 위치	크기 (MB)	이미지 URI	다이제스트	스캔 상태	취약성
<input type="checkbox"/>	v1	Image	2023년 11월 30일, 18:49:51 (UTC+09)	284.89	URI 복사	sha256:8dd65411905ea3...	-	-

01 프로젝트 개요

▶ 기획 의도

1. 모듈화와 유연성 강화

목표: 쿠팡 클론 프로젝트는 마이크로서비스 아키텍처를 통해 각 기능을 독립적인 서비스로 모듈화하고, 각 서비스 간의 의존성을 최소화하여 유연성을 향상시킬 것입니다.

이점: 새로운 기능 추가 또는 기존 기능 변경 시 해당 기능에 대한 서비스만 업데이트할 수 있어 빠른 개발 및 배포가 가능해지며, 시스템의 확장성이 향상됩니다.

2. 서비스 디스커버리 및 로드 밸런싱 적용

목표: 서비스 디스커버리와 로드 밸런싱 기술을 적용하여 서비스 간의 통신을 원활하게 관리하고 부하를 분산시킬 것입니다.

이점: 사용자에 대한 부하를 고르게 분산하여 안정적이고 빠른 서비스를 제공하며, 서비스의 다운타임을 최소화합니다.

3. 데이터 관리와 일관성 유지

목표: 데이터베이스를 각 마이크로서비스에 맞게 선택하고, 데이터 일관성을 유지하는 방안을 마련할 것입니다.

이점: 각 서비스의 독립된 데이터베이스를 관리하여 데이터 일관성을 유지하고, 필요에 따라 관계형 또는 NoSQL 데이터베이스를 선택하여 데이터 관리의 효율성을 극대화할 수 있습니다.

4. 보안 및 모니터링 강화

목표: 각 마이크로서비스 간의 보안을 강화하고, 로깅 및 모니터링 시스템을 도입하여 시스템의 안전성과 신뢰성을 확보할 것입니다.

이점: 보안 측면에서 각 서비스의 독립성을 유지하며, 모니터링을 통해 시스템의 이상 징후를 신속하게 감지하여 대응할 수 있습니다.

01 프로젝트 개요

▶ 기대 효과

시장 이해 및 경쟁력 강화: 쿠팡은 대한민국을 중심으로 매우 성공적인 전자상거래 플랫폼 중 하나입니다.

쿠팡 클론 프로젝트를 통해 해당 시장의 동향, 소비자 요구 사항, 경쟁 구도 등을 더 깊이 이해할 수 있습니다.

기술적 경험 증진: 프로젝트를 통해 특정 기술 스택이나 아키텍처에 대한 경험을 얻을 수 있습니다.

이는 실무에서의 기술 역량을 향상시키고, 새로운 기술 도입에 대한 테스트 베드로 활용될 수 있습니다.

맞춤형 기능 및 서비스 추가: 프로젝트를 통해 얻은 경험을 기반으로, 기존의 비즈니스 모델을 확장하거나 새로운 기능을 추가함으로써 더 다양한 서비스를 제공할 수 있습니다.

비용 절감 및 효율화: 이미 검증된 비즈니스 모델과 기술 스택을 활용하면 초기에는 개발 및 운영 비용을 절감 할 수 있습니다. 또한, 쿠팡의 운영 모델에서 배울 점을 찾아 효율화에 기여할 수 있습니다.

1) 사전회의 쿠팡에서 구현할 기능과 구현하지 않을 기능을 선별합니다.

2) 서비스 구성 및 분배와 구현

회원 : 사용자 정보를 관리하는 서비스로, 사용자 생성, 수정, 삭제, 조회 등을 처리합니다.

주문 : 주문 생성, 주문 상태 변경, 주문 처리 등을 수행합니다.

결제 : 주문에 대한 결제를 처리하는 서비스입니다.

리뷰 : 제품에 대한 리뷰는 상품의 품질, 배송 서비스 등에 대한 피드백을 제공합니다.

상품 : 전자제품, 패션, 식품, 생활용품 등 다양한 상품을 보유하고 있습니다.

상담 : 온라인에서 제품에 대한 정보를 얻거나 구매에 대한 도움이 필요한 고객들을 위해 고객 서비스를 제공합니다.

기획 : 전체적인 틀과 구성을 계획하고 목표를 설정합니다.

CI/CD : 소프트웨어 개발에서 적용되는데, 항상 안정적이고 최신 기능을 반영하기 위한 소프트웨어 업데이트 및 배포를 지속적으로 진행할 수 있도록 도와줍니다.

3) 서비스 외 구성

게이트웨이 (Gateway) : 클라이언트와 통신하는 역할로 API 게이트웨이를 사용합니다. 예를 들어, Spring Cloud Gateway, 혹은 Nginx 등을 사용할 수 있습니다.

서비스디스커버리및로드밸런싱: 서비스들이 자동으로 등록되고 검색될 수 있도록 하는 서비스 디스커버리를 구성합니다. 필요시 로드 밸런서를 사용하여 트래픽을 분산시킵니다.

데이터베이스 : 각 서비스의 데이터베이스를 분리하여 독립적으로 관리합니다.

4) 마이크로서비스 통합 Test 후 평가

5) 분석 및 모니터링 서비스의 성능 및 사용 현황을 모니터링하는 도구를 도입 합니다. 로그 및 메트릭 데이터를 수집하여 중앙 집중식으로 관리할 수 있도록 합니다.

▶ 활용 장비 및 개발 환경

- JDK 17
- Spring Boot : 3.1.5
- Spring Security
- Spring Web
- Spring Data JPA
- Spring Cloud
- Spring Actuator
- Spring Resilience4j
- MariaDB
- Redis
- Apache Kafka
- Eureka Discovery
- Spring API Gateway
- Spring OAuth2 Authorization server
- Spring Load Balance
- **Spring WebFlux**
- Docker Desktop
- Docker Compose
- Kubernetes(K8S)
- Portone API
- AWS
- WebSocket / STOMP
- Vault
- MicroMeter
- Jenkins
- AWS ECR
- Zipkin
- Tomcat
- Netty
- Nginx
- Angular JS
- ElasticSearch
- Kinaba
- Prometheus & Grafana
- Gmeter
- Github
- IntelliJ
- Postman

02 프로젝트 수행 결과

결과 제시 2. Docker-Compose

The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. The sidebar on the left includes links for Docker Desktop, Upgrade plan, Containers (selected), Images, Volumes, Dev Environments (BETA), Docker Scout, Learning center, Extensions (selected), and Add Extensions.

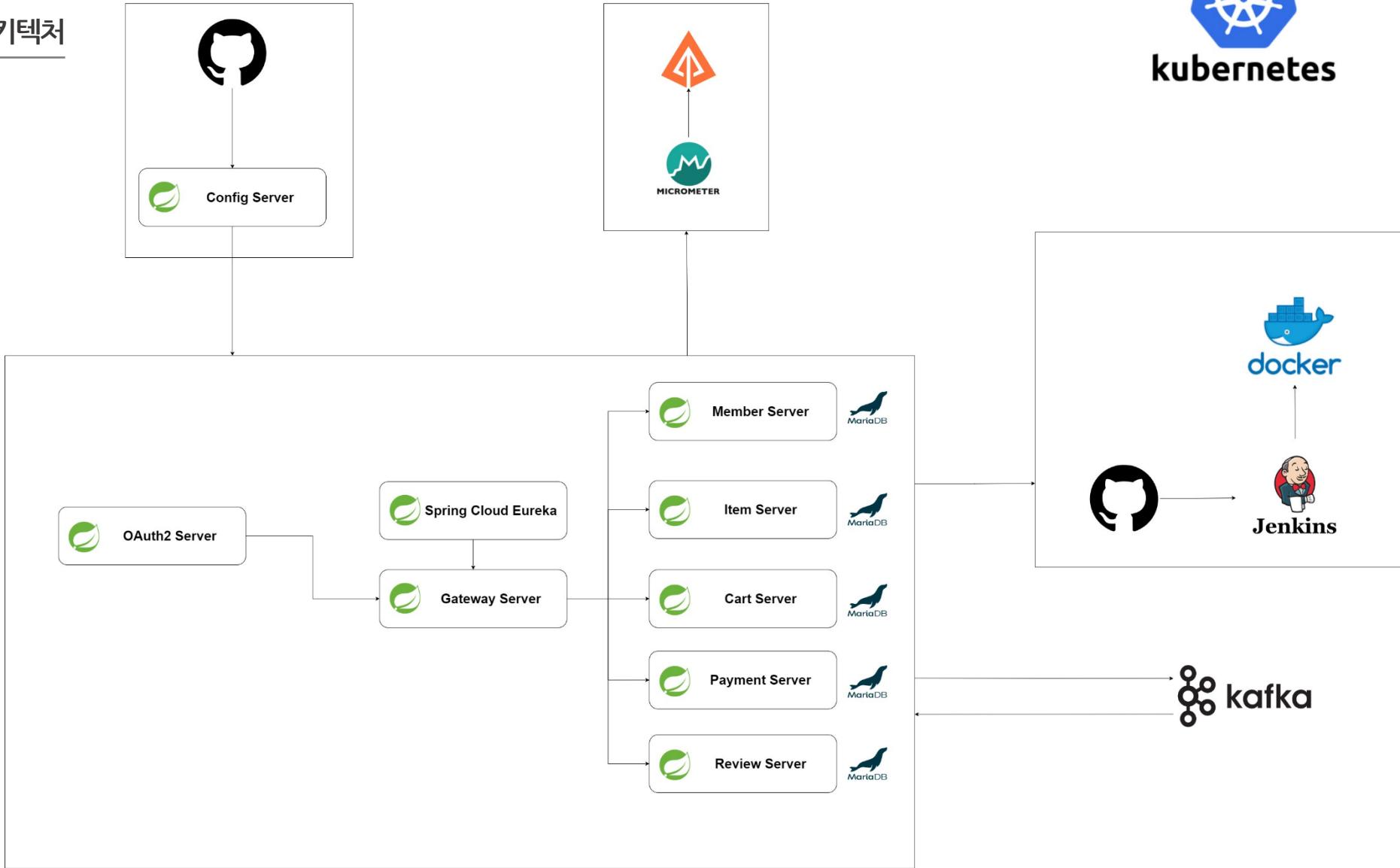
At the top, there are performance metrics: Container CPU usage (3.51% / 600%) and Container memory usage (2.97GB / 11.35GB). A search bar and a 'Ctrl+K' keyboard shortcut are also present.

The main area displays a table of running containers:

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
docker-compose		Running (11/1)	3.51%		3 minutes ago	⋮ ⋮ ⋮
configserver-1	gopang/configserver:0.0.1-SNAPSHOT	Running	0.1%	8071:8071	4 minutes ago	⋮ ⋮ ⋮
angular-app	nginx-server:web-server	Running	0%	80:80	4 minutes ago	⋮ ⋮ ⋮
zookeeper-1	bitnami/zookeeper:latest	Running	0.09%	2181:2181	4 minutes ago	⋮ ⋮ ⋮
zipkin-1	openzipkin/zipkin	Running	0.09%	9411:9411	4 minutes ago	⋮ ⋮ ⋮
database-1	mariadb:latest	Running	0.02%	3306:3306	4 minutes ago	⋮ ⋮ ⋮
kafka-1	bitnami/kafka:latest	Running	1.87%	9092:9092	4 minutes ago	⋮ ⋮ ⋮
eurekaserver-1	gopang/eurekaserver:0.0.1-SNAPSHOT	Running	0.11%	8070:8070	3 minutes ago	⋮ ⋮ ⋮
gopangitems-1	gopang/itemserver:0.0.1-SNAPSHOT	Running	0.1%	8081:8081	3 minutes ago	⋮ ⋮ ⋮
gopangpayment-1	gopang/paymentserver:0.0.1-SNAPSHOT	Running	0.51%	8083:8083	3 minutes ago	⋮ ⋮ ⋮
orderservice-1	gopang/orderserver:0.0.2-SNAPSHOT	Running	0.55%	8082:8082	3 minutes ago	⋮ ⋮ ⋮
gatewayserver-1	gopang/gatewayserver:0.0.1-SNAPSHOT	Running	0.07%	8072:8072	3 minutes ago	⋮ ⋮ ⋮

02 프로젝트 수행 결과

결과 제시 3. 마이크로 서비스 아키텍처



감사합니다
