인력과 운영비 절감을 위한

무인 스마트 팝업스토어 관리 서비스

PoP!



리테일 Team.POPI

멘토 | 윤혜원

팀장/백엔드 | 오형서

프론트엔드 | 심수연, 유영서, 정성엽

백엔드 | 박상욱, 박정연, 조영상, 최현태

LG CNS AM INSPIRE 1기

Contents

01
프로젝트 개요
- Mission
- 프로젝트 주제
- Persona
- 차별점

02
서비스 소개
- 메인 시나리오
- UI

03
프론트엔드
- 기술 스택
- 성능 측정
- 성능 & UX 개선
- E2E Test

04
백엔드
- 서비스 & 인프라
- 핵심 로직
- 성능 개선
- 테스트
- 로깅 & 모니터링

05
마무리
- 협업 과정
- 클로징

01

프로젝트 개요

01 Misson



기획안을 모두 반영하려 했지만
현실적인 제약도 고려해야 했어요

We tried to reflect the entire plan,but had to consider real-world constraints.

기존 주제

매대 앞 상품을 D415 카메라로 인식해
결품 여부를 파악하고, 자동으로 발주까지 연결되는
무인 스마트스토어 시스템을 만들어보세요

한계점

카메라의 설치 위치나 거리 및 각도 범위 등
기술적 제약으로 결품 인식에 어려움이 있었고,
결제 데이터를 기반으로 결품 파악이 가능했어요

01 Misson

IDEA

거리기반 측정 카메라를 사용하여 **체류시간을 측정하고 구매전환율 데이터**를 제공하고

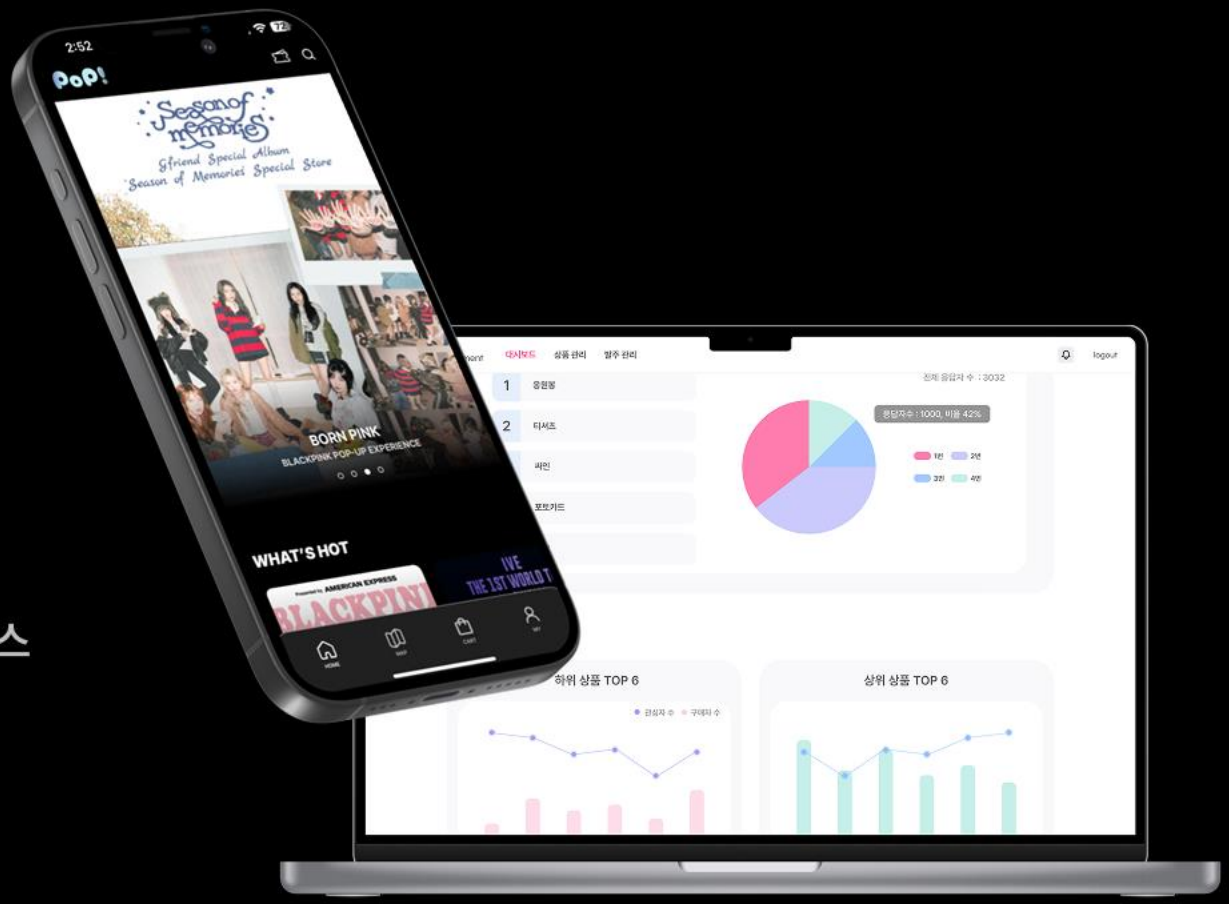
팝업스토어라는 구체적인 시나리오를 특정해 **사용자 서비스**도 함께 개발했어요

PoP! 무인 스마트 팝업스토어 관리 서비스

02 프로젝트 주제

POPI

무인 스마트 팝업스토어 관리 서비스



What is POPI?

무인 스마트 팝업스토어 운영을 위한 **All-In-One** 관리 서비스

- 고객 **예약**부터 **결제**까지 어플을 통해 수행
- 실시간 데이터 분석 및 **차트** 제공
- 분석된 데이터 기반으로 **발주 프로세스** 수행

03 Persona

효율적으로 팝업을 운영하고 싶은 브랜드 마케팅 담당자



팝업스토어 운영자

정우식

32세 남자

PIPO 엔터 마케팅 팀장

Needs

- 마케팅에 활용할 수 있는 체류 시간 같은 다양한 데이터를 수집해 브랜드 이미지를 개선하고 싶다.
- 매번 발주 타이밍이나 수량을 예측하기 어렵다. 필요한 만큼만 정확히 발주할 수 있으면 좋겠다.
- 인건비 부담이 크다. 현장 인력을 최소화해 더 효율적인 매장 운영을 원한다.

아이돌을 좋아하고 트렌드에 민감한 MZ 여대생



팝업스토어 사용자

김서연

24세 여자

K-POP 아이돌 덕질이 취미

Needs

- 데이트 코스로 밥 먹고 카페 가는 건 지겹고, 근처에 재밌는 팝업이 있으면 좋겠다.
- 팝업 입장 대기줄은 항상 너무 길다. 미리 예약할 수 있다면 편할 것 같다.
- 인기 많은 팝업은 결제 줄도 길어서, 사고 싶은 것만 빨리 사서 나가고 싶다.

04 차별점

	팝플리	팝가	캐치테이블	테이블링	POPI
자체적인 예약 서비스 운영	✕	✕	○	○	○
사전 설문조사 여부	✕	✕	✕	✕	○
구매전환율 분석 (카메라)	✕	✕	✕	✕	○
관리자 알림 기능	△	△	○	○	○
인앱 결제	✕	✕	○	△	○
현장 대기 QR 코드	✕	○	○	○	○
재고 관리 기능	✕	✕	✕	✕	○
발주 기능	✕	✕	✕	✕	○

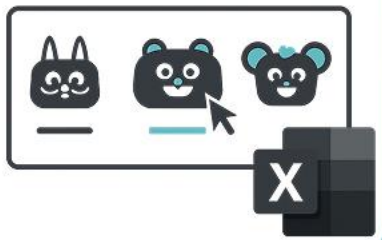
- 팝플리: https://www.popply.co.kr/
- 팝가: https://popga.co.kr/
- 캐치테이블: https://app.catchtable.co.kr/
- 테이블링: https://www.tabling.co.kr/

02

서비스 소개

01 메인 시나리오

01 팝업 등록



하나의 운영자 계정으로
여러 팝업을 관리

02 팝업 예약



팝업을 사전 예약
QR기반 간편 입장

03 상품 결제



상품 QR 스캔
장바구니 담기
모바일 간편 결제

04 발주 제안



재고 조회
발주 요청
발주 제안 알림

05 대시 보드



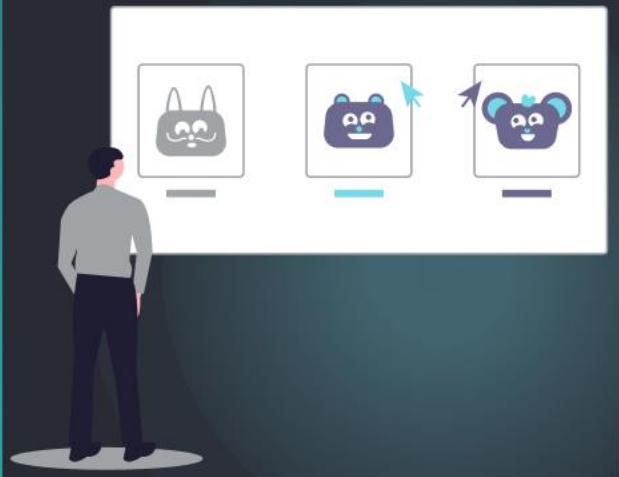
다양한 방문자 데이터를
대시보드로 제공

01 메인 시나리오

1. 팝업 등록

팝업 등록 시나리오 1

하나의 운영자 계정으로
여러 팝업을 관리할 수 있어요



팝업 등록 시나리오 2

새로운 팝업 스토어를
운영자 웹에서 생성할 수 있어요



팝업 등록 시나리오 3

팝업에서 판매할 상품을
엑셀 파일 또는 직접 등록해요

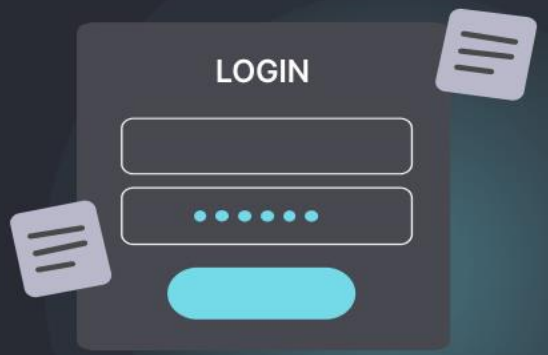


01 메인 시나리오

2. 팝업 예약

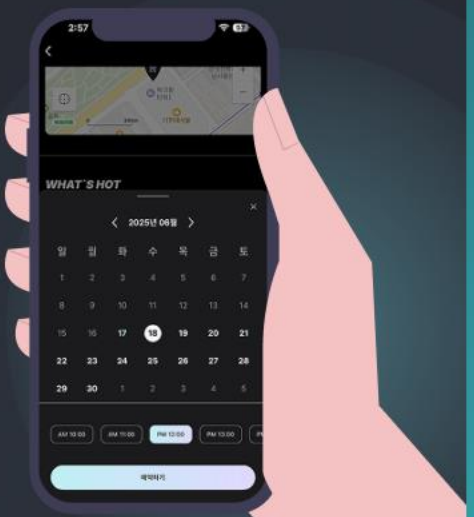
팝업 예약 시나리오 1

사용자 앱에서
로그인을 진행해요



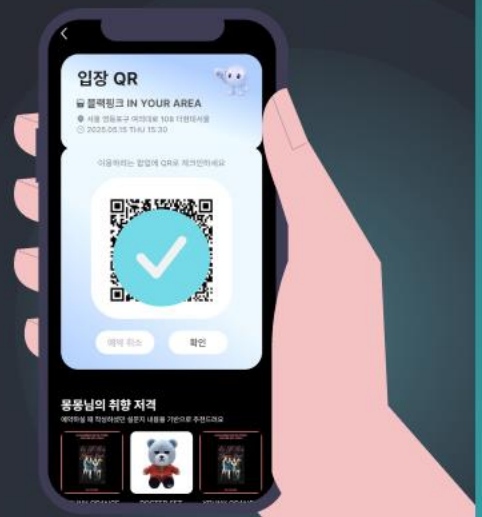
팝업 예약 시나리오 2

팝업스토어를 선택해
원하는 날짜에 예약해요



팝업 예약 시나리오 3

팝업 예약을 완료해
입장 QR을 발급받아요



01 메인 시나리오

3. 상품 결제

상품 결제 시나리오 1

사용자가 예약했던 팝업에
입장 QR을 찍어 방문해요



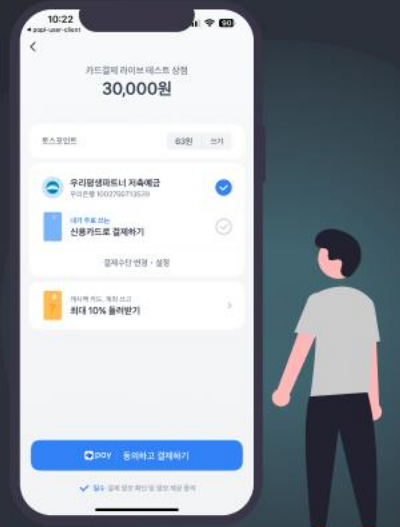
상품 결제 시나리오 2

상품을 골라 앱 내 카메라로
상품 QR을 인식시켜요



상품 결제 시나리오 3

장바구니에 담긴 상품을
앱 내에서 편하게 결제해요



01 메인 시나리오

4. 발주 제안

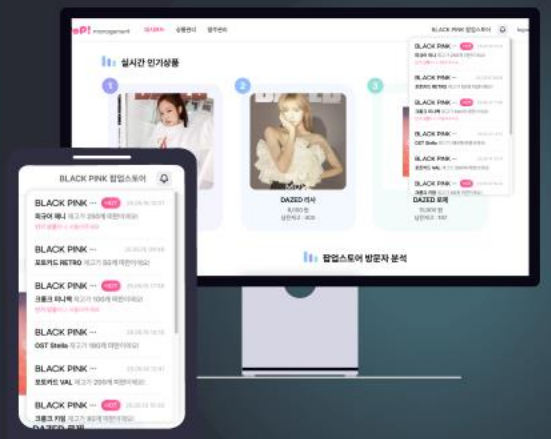
발주 제안 시나리오 1

사용자가 상품을 결제해
발주 기준 재고에 도달했어요



발주 제안 시나리오 2

운영자 웹에서 실시간으로
발주 제안 알림을 받아요



발주 제안 시나리오 3

운영자는 제안 받은 발주를
진행할지 결정할 수 있어요



01 메인 시나리오

5. 대시보드

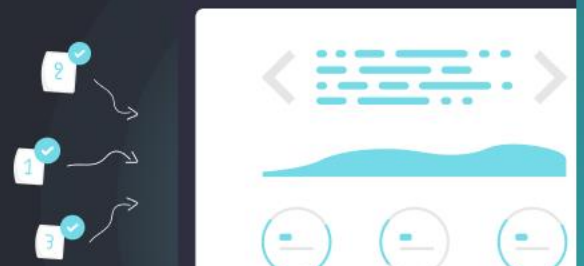
대시보드 시나리오 1

상품에 대한 관심도 데이터를 카메라로 측정해요



대시보드 시나리오 2

사용자 앱과 카메라로 얻은 데이터를 분석해요



대시보드 시나리오 3

대시보드로 매장 데이터를 확인할 수 있어요



시연 영상



02 UI - 사용자

HOME TAP

사용자는 팝업스토어를
탐색하고 예약할 수 있어요





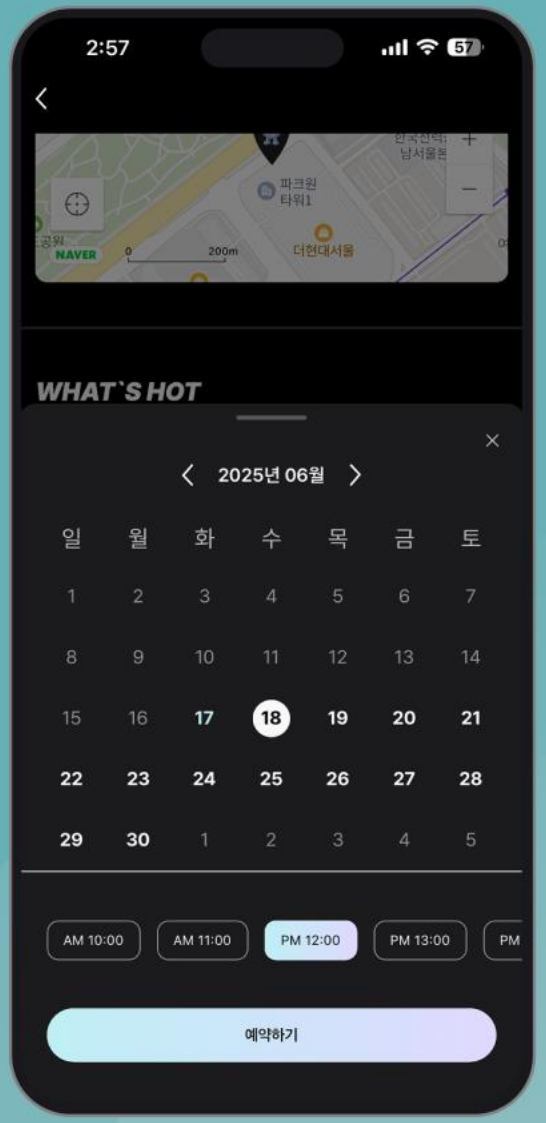


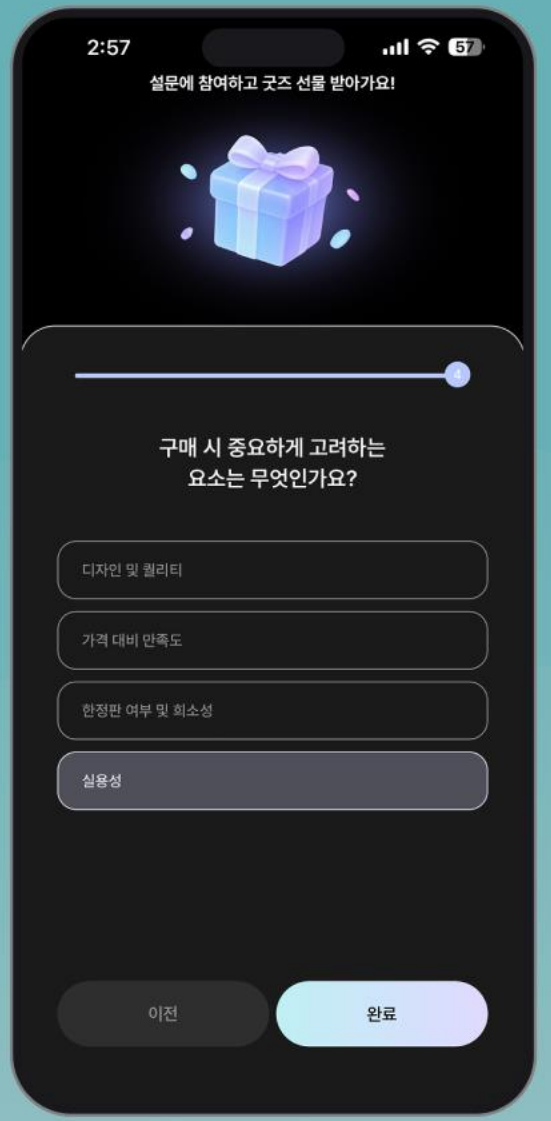
02 UI - 사용자

HOME TAP

예약과 설문조사를 하고
팝업 입장 QR을 발급받아요







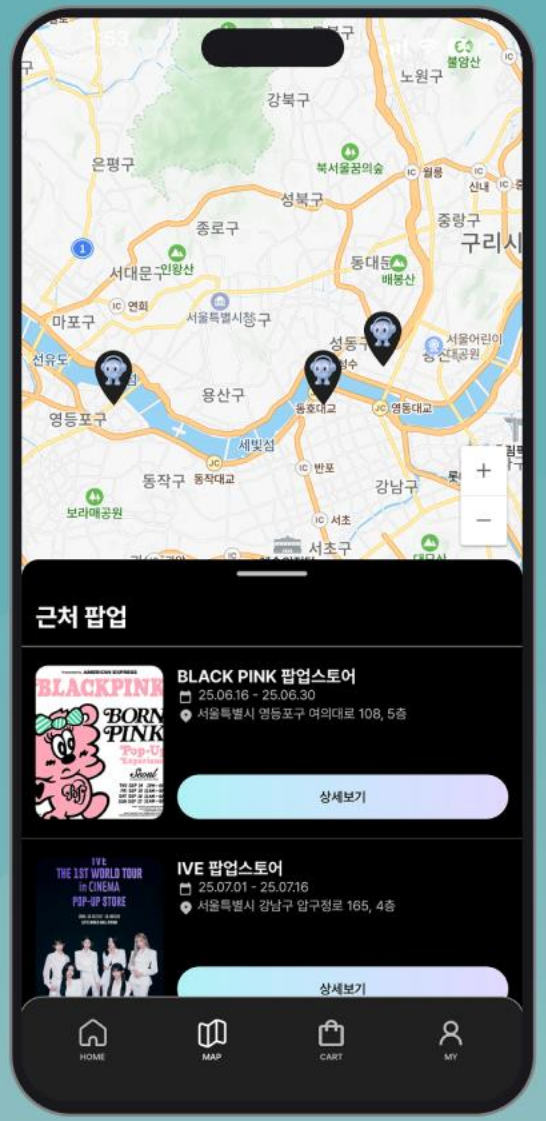


02 UI - 사용자

MAP TAP

주변에 등록된 팝업스토어를 탐색하고 상세 페이지로 이동할 수 있어요







02 UI - 사용자

CART TAP

QR을 스캔해 자동으로 상품을 장바구니에 담아요





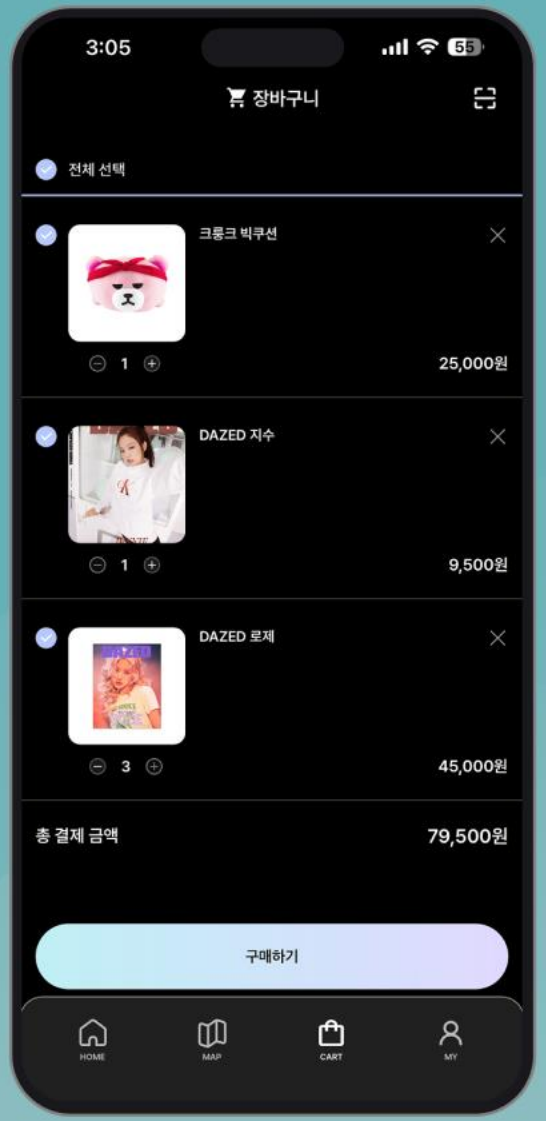


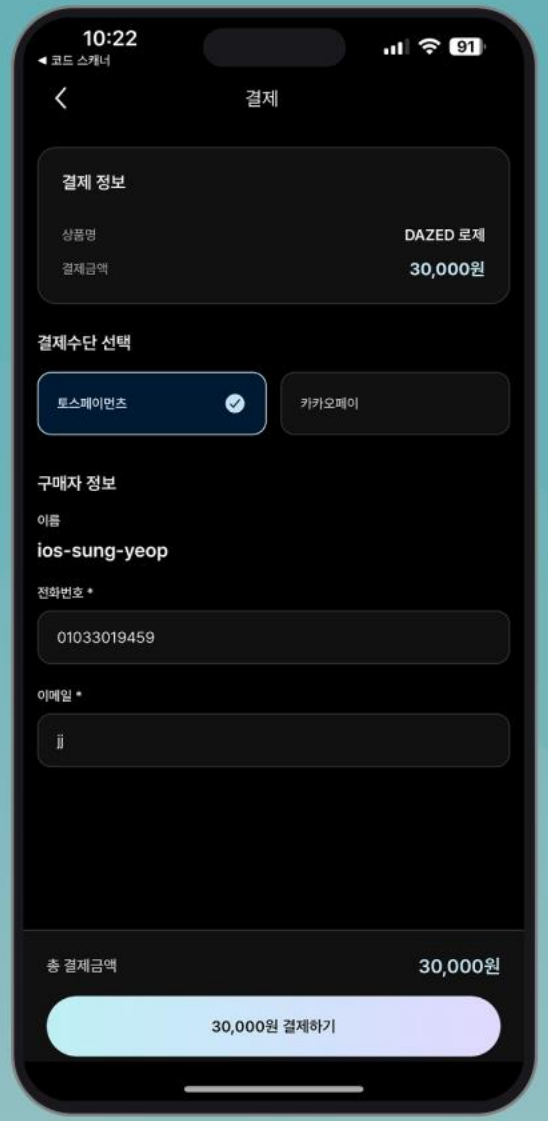
02 UI - 사용자

CART TAP

장바구니에 담은 상품들을
간편하게 앱 내에서 결제해요







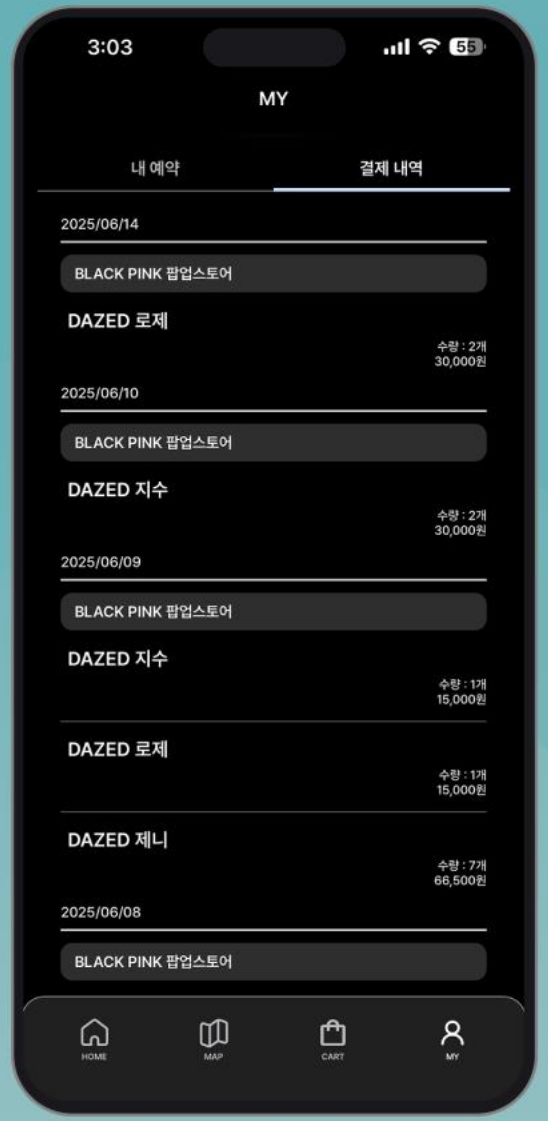
02 UI - 사용자

MY TAP

예약한 팝업과 결제 내역을 확인할 수 있어요

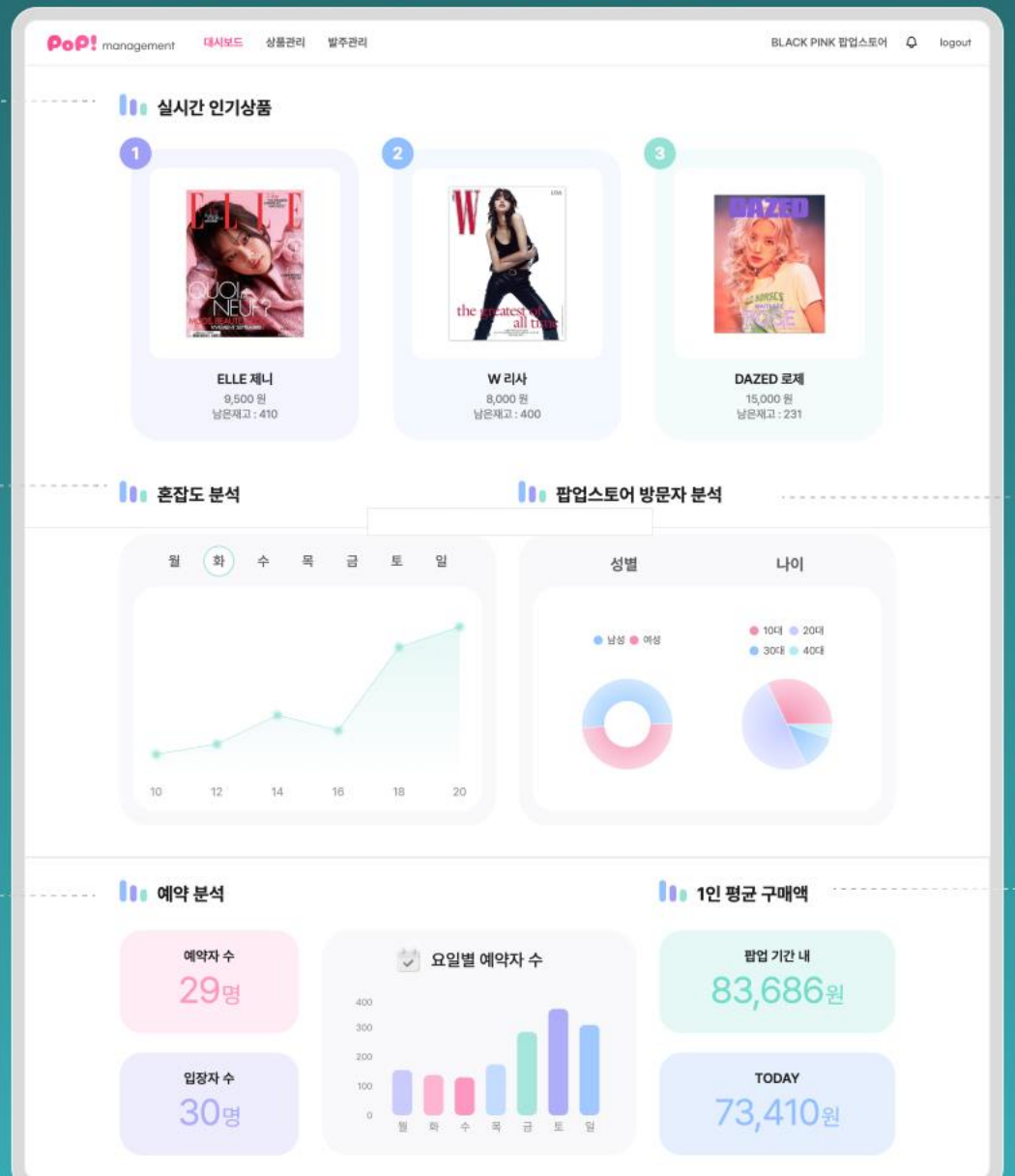






02 UI - 운영자 대시보드

- 대시보드의 모든 데이터들은 운영자가 설정한 단위로 업데이트 됩니다



실시간 인기상품 TOP3



실제 구매율 +
카메라 체류시간 + 접근거리 데이터

혼잡도 분석

요일별 해당 시간대 입장자 수 평균



예약 분석

예약자 수와 실제 입장자 수
요일별 예약자 수

방문자 분석

요일별 해당 시간대 입장자 수 평균



구매액 분석

팝업 기간 전체, TODAY 1인 평균 구매액

02 UI - 운영자 대시보드

- 대시보드의 모든 데이터들은 운영자가 설정한 단위로 업데이트 됩니다

실시간 인기상품 TOP3

실제 구매율 +
카메라 체류시간 + 접근거리 데이터



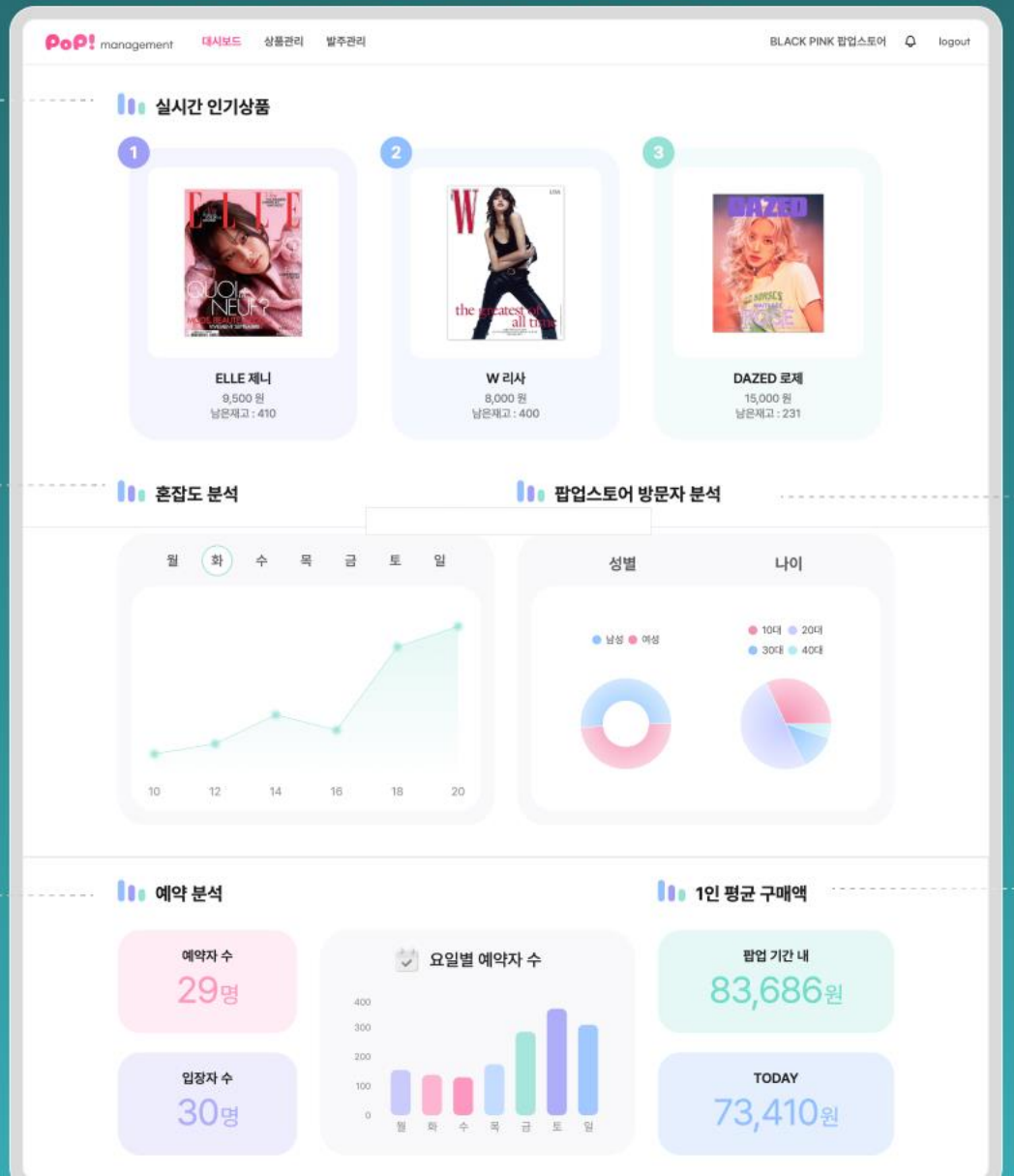
혼잡도 분석

요일별 해당 시간대 입장자 수 평균



예약 분석

예약자 수와 실제 입장자 수
요일별 예약자 수



방문자 분석

요일별 해당 시간대 입장자 수 평균

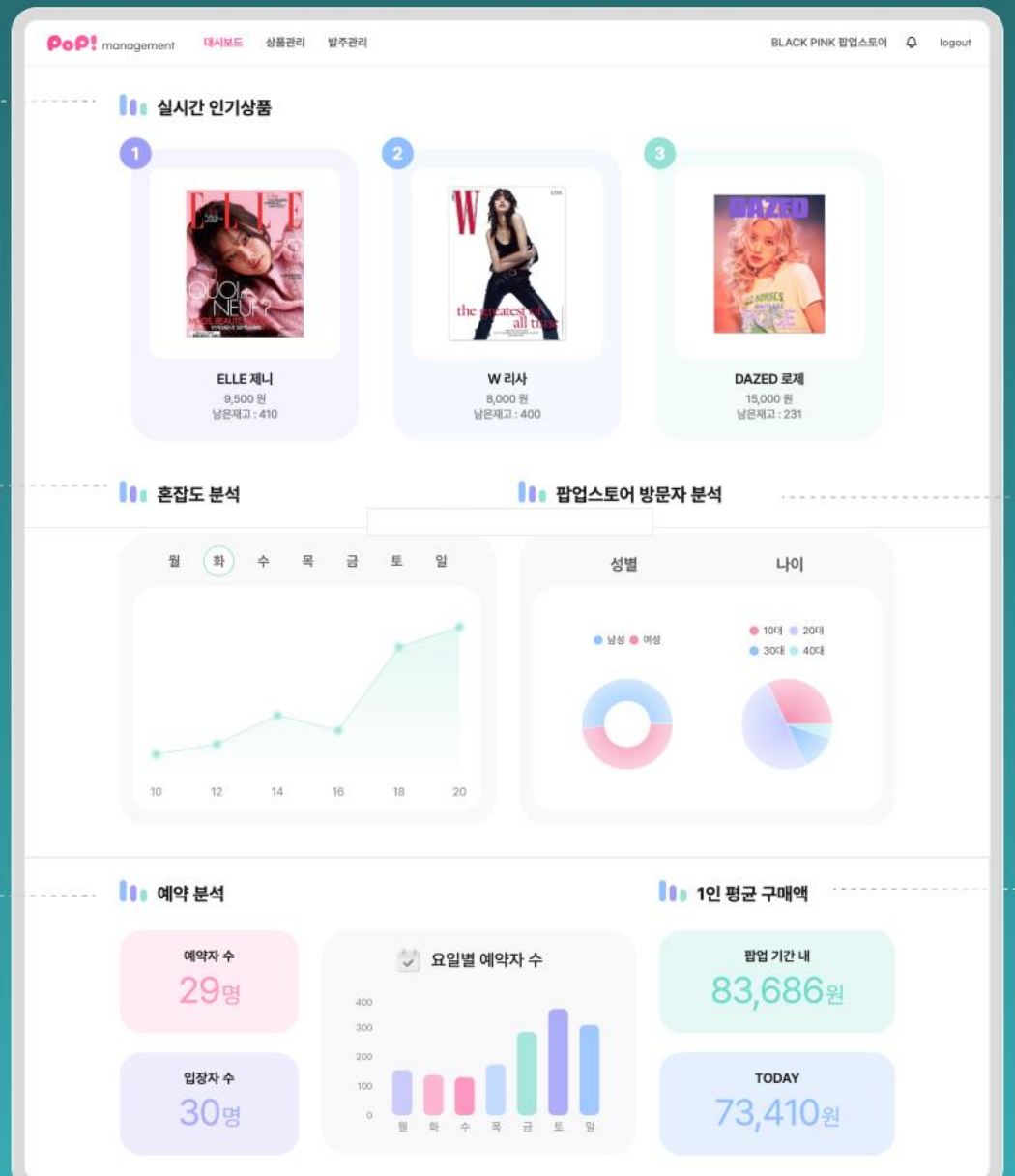


구매액 분석

팝업 기간 전체, TODAY 1인 평균 구매액

02 UI - 운영자 대시보드

- 대시보드의 모든 데이터들은 운영자가 설정한 단위로 업데이트 됩니다



실시간 인기상품 TOP3



실제 구매율 +
카메라 체류시간 + 접근거리 데이터

혼잡도 분석

요일별 해당 시간대 입장자 수 평균



예약 분석

예약자 수와 실제 입장자 수
요일별 예약자 수

방문자 분석

요일별 해당 시간대 입장자 수 평균



구매액 분석

팝업 기간 전체, TODAY 1인 평균 구매액

02 UI - 운영자 대시보드

- 대시보드의 모든 데이터들은 운영자가 설정한 단위로 업데이트 됩니다

실시간 인기상품 TOP3



실제 구매율 +
카메라 체류시간 + 접근거리 데이터

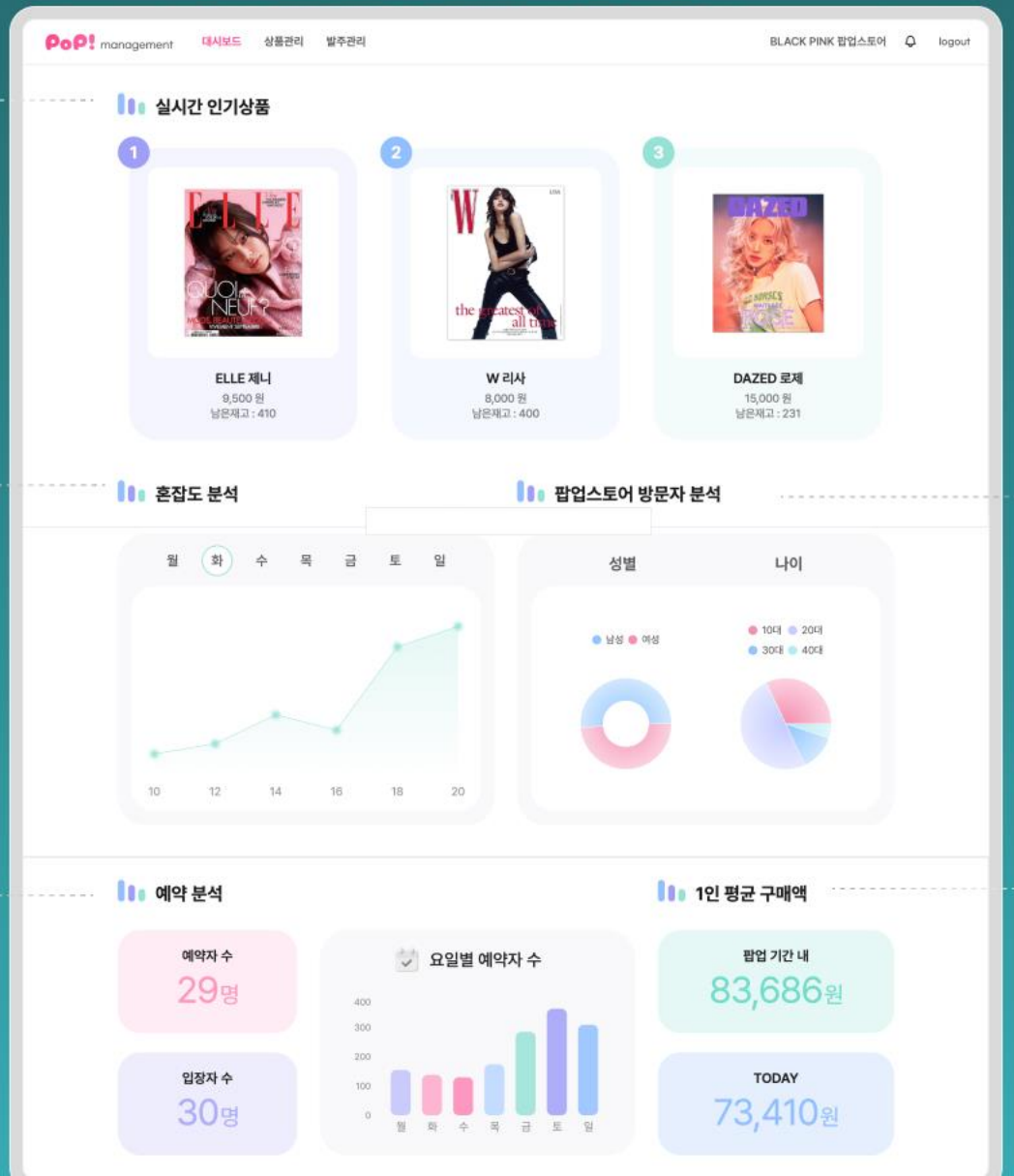
혼잡도 분석

요일별 해당 시간대 입장자 수 평균



예약 분석

예약자 수와 실제 입장자 수
요일별 예약자 수



방문자 분석

요일별 해당 시간대 입장자 수 평균



구매액 분석

팝업 기간 전체, TODAY 1인 평균 구매액

02 UI - 운영자 대시보드

- 대시보드의 모든 데이터들은 운영자가 설정한 단위로 업데이트 됩니다

설문지 분석

예약할 때 작성한 설문지 답변 통계



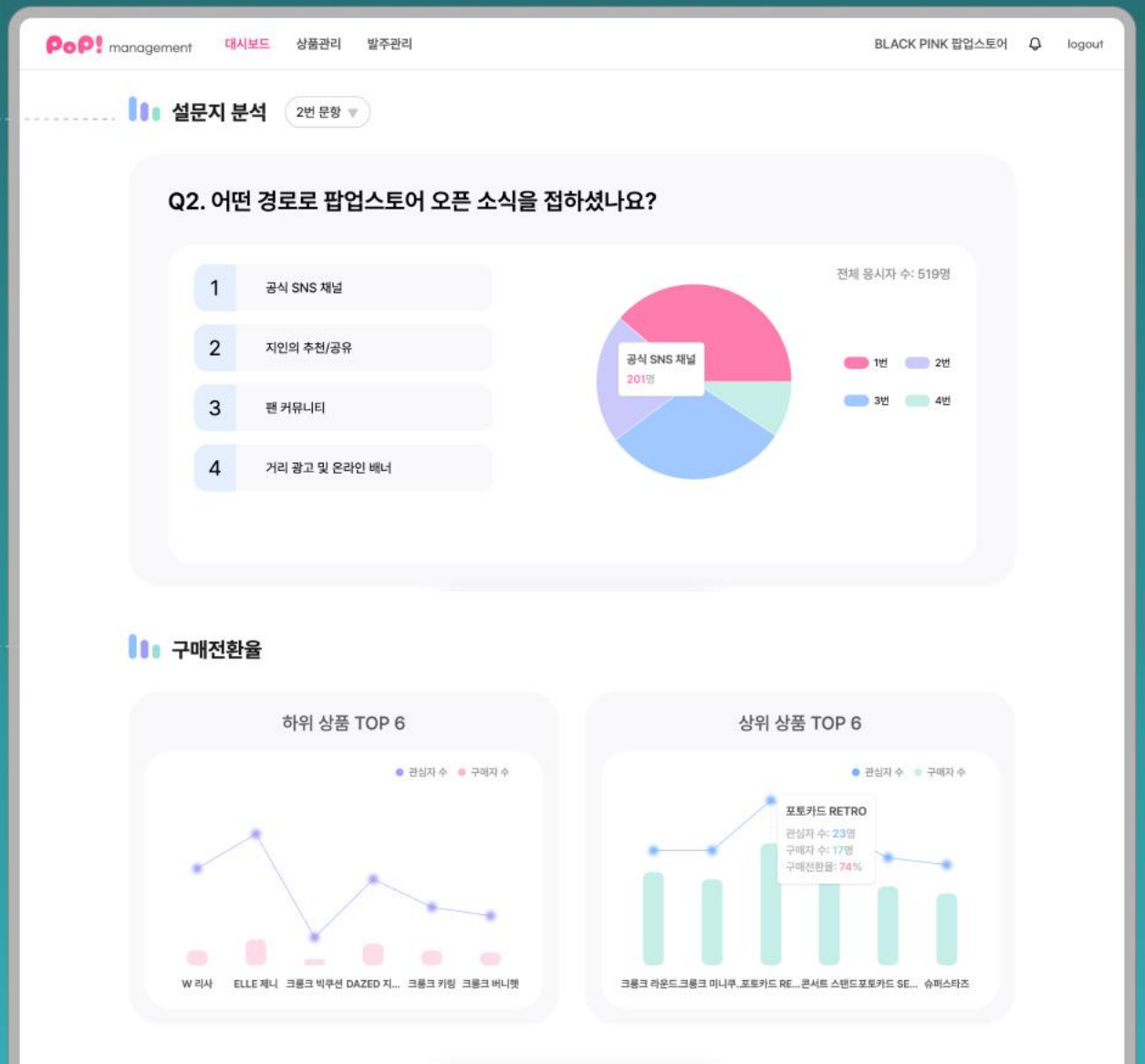
구매전환율

카메라로 분석한 관심자 수와 실 구매자 수를 비교한 구매전환율 분석

관심자 수: 체류 시간 + 접근 거리

구매전환율: 구매자 수 / 관심자 수 * 100





02 UI - 운영자 대시보드

- 대시보드의 모든 데이터들은 운영자가 설정한 단위로 업데이트 됩니다

설문지 분석

예약할 때 작성한 설문지 답변 통계



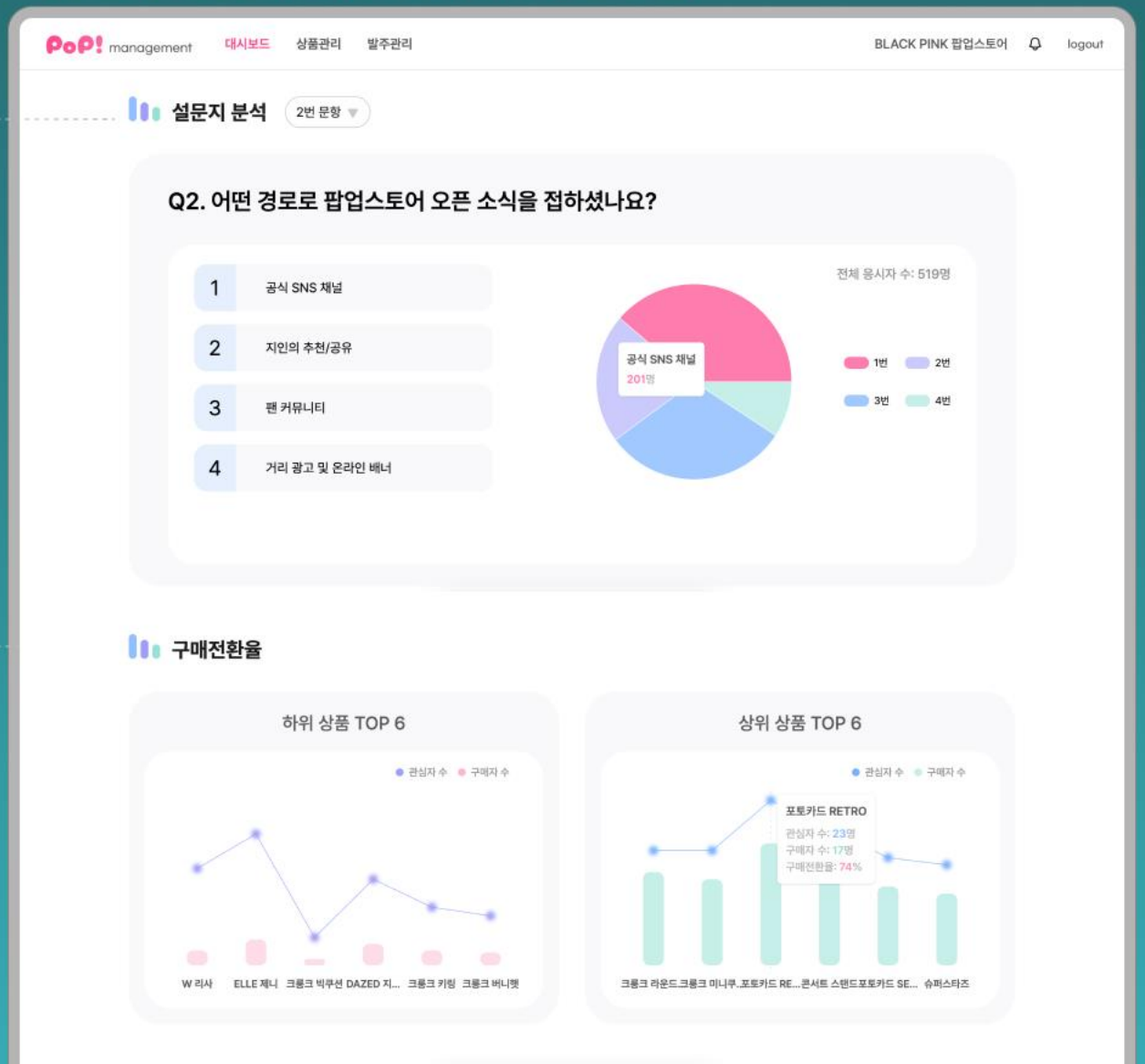
구매전환율

카메라로 분석한 관심자 수와 실 구매자 수를 비교한 구매전환율 분석

관심자 수: 체류 시간 + 접근 거리

구매전환율: 구매자 수 / 관심자 수 * 100





03

프론트엔드

01 기술 스택

운영자

Web

Software


React


TypeScript


Storybook


MSW


Zustand


TailwindCSS


Vite


TanStack Query

Test


Playwright

CI/CD


Github Action


S3


CloudFront

01 기술 스택

사용자

App

Software



React Native



TypeScript



Expo CLI



Zustand



Styled Components



TanStack Query

Test



Jest

CI



Github Action

01 기술 스택

입장 QR Reader

Software



Next



TypeScript



Electron



TailwindCSS



TanStack Query

Why Expo?

왜 Expo를 사용했나요?

React Native CLI vs Expo CLI

Android & iOS 플랫폼 모두 개발해야하는 상황

여러 개의 네이티브 모듈을 사용해야하는 상황

사용할 모듈



네이버 지도



FCM



PortOne



OpenID
Certified

Why Expo?

Expo CLI를 선택한 이유

React Native CLI

비교적 자유로운 네이티브 기능 사용

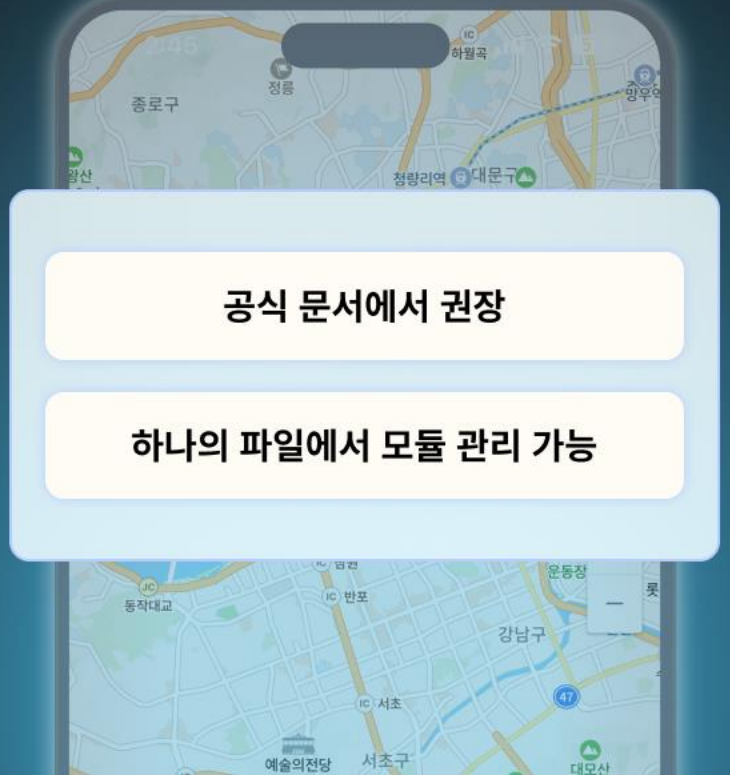
두 플랫폼에서 각각 기능 설정 필요



Expo CLI

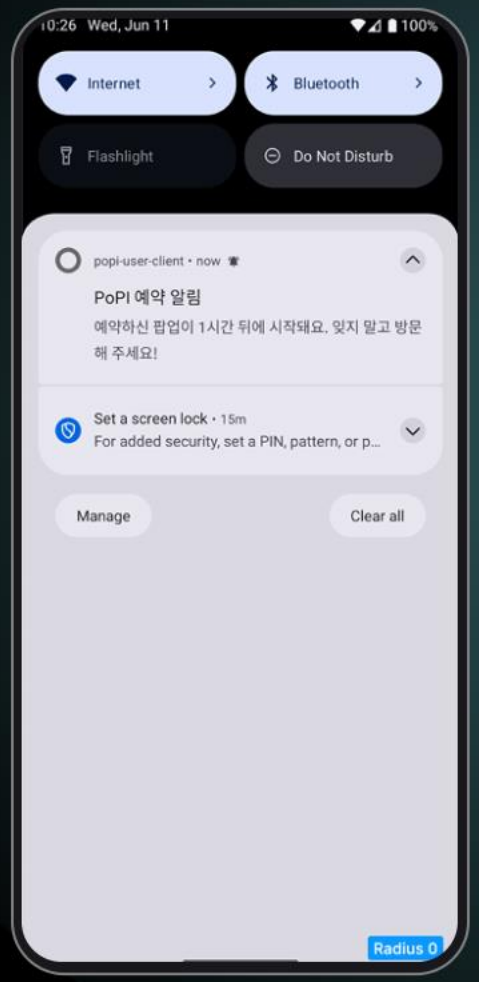
공식 문서에서 권장

하나의 파일에서 모듈 관리 가능



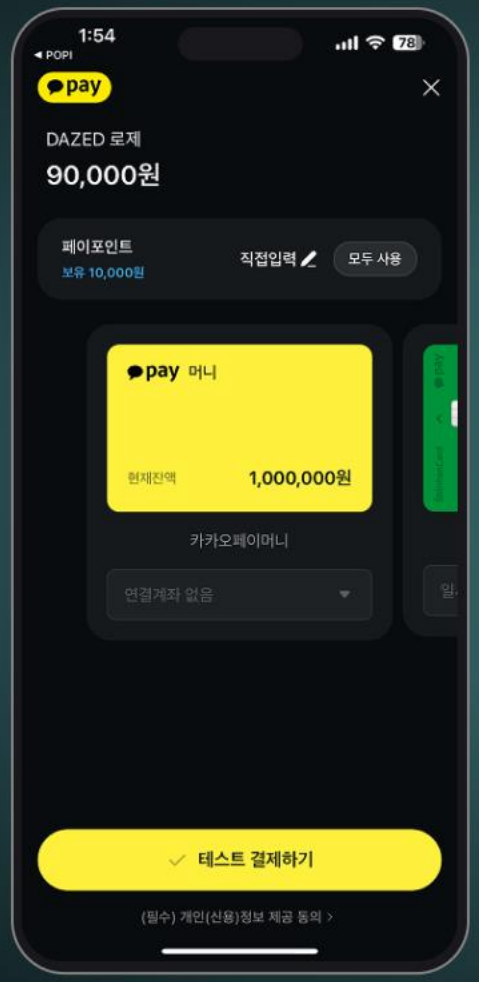
01 기술 스택

Android



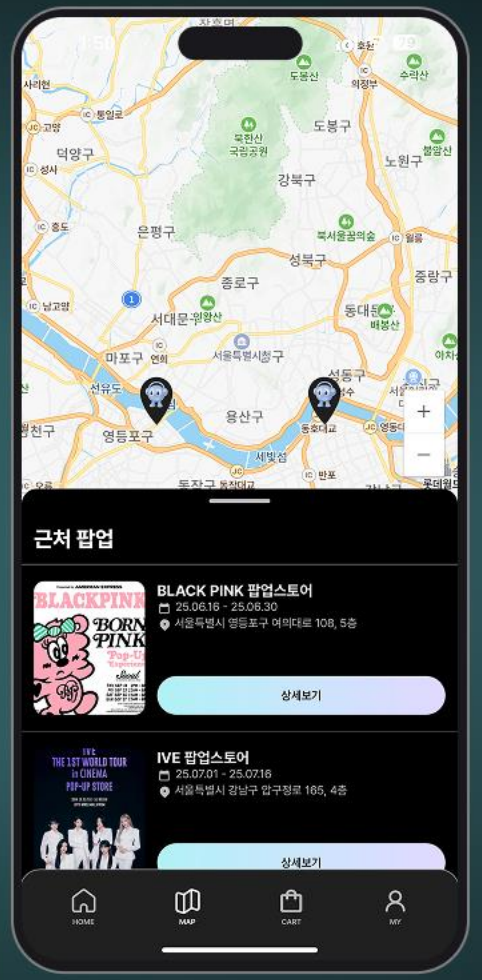
FCM

iOS



PortOne

iOS



Naver map

Expo CLI 설정 파일에서

Android & iOS 모두 관리

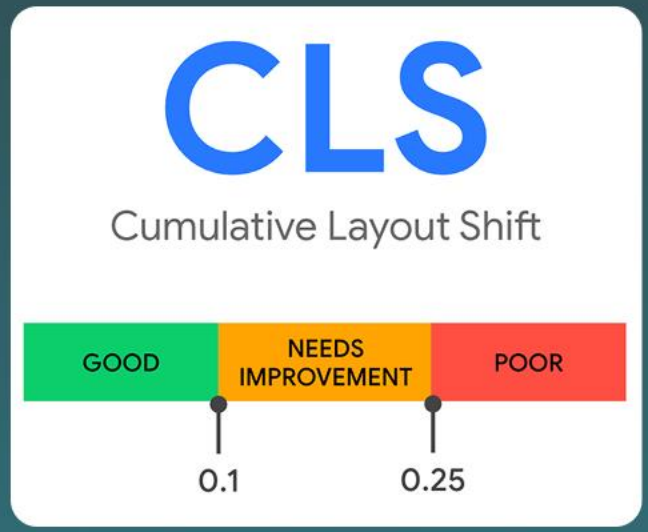
빌드 진행 시

크로스 플랫폼 일관성 유지

How to Calculate?

성능 측정 지표





유의미한 콘텐츠의 최초 렌더링 시간 (2.5초 권장)

레이아웃이 예상치 못하게 변경되는 빈도수

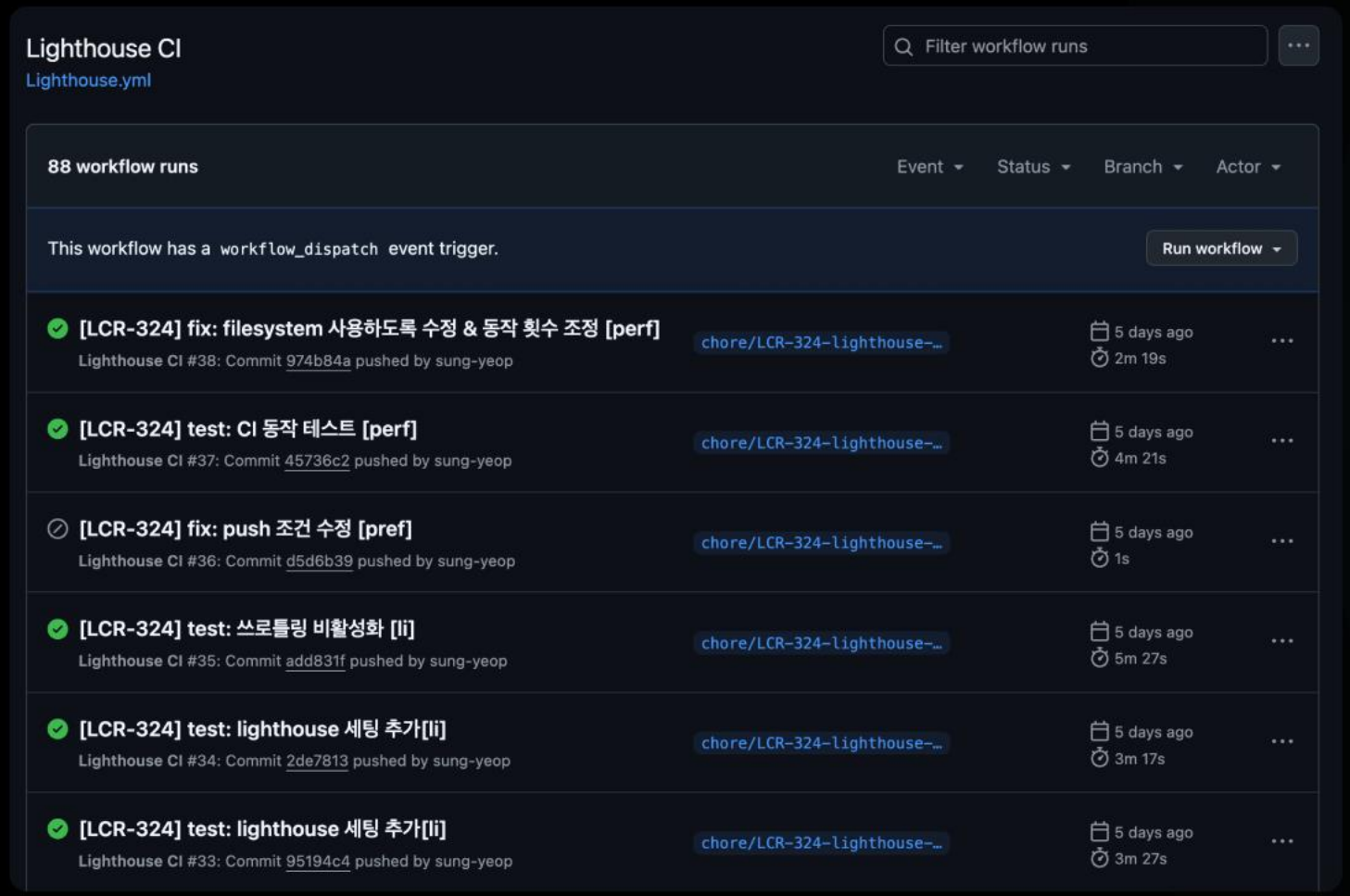
How to Calculate?

그렇다면 성능 측정은 어떻게 해야할까?

1. 프론트엔드 팀원들이 서로 다른 개발환경에서 측정을 진행해도 동일한 결과가 나와야 한다

2. 여러 번 실행해도 코드를 변경하지 않았다면 동일한 결과가 나와야 한다

02 성능 측정



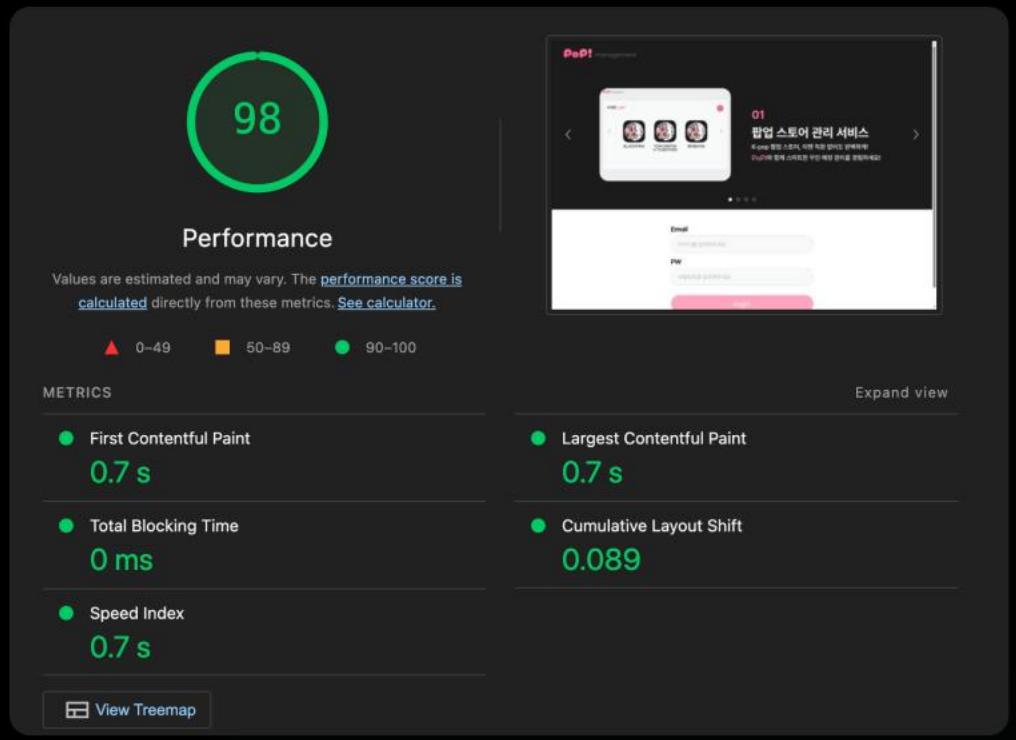
최대한 독립된 환경에서 측정하기 위해,

Lighthouse CI 도입

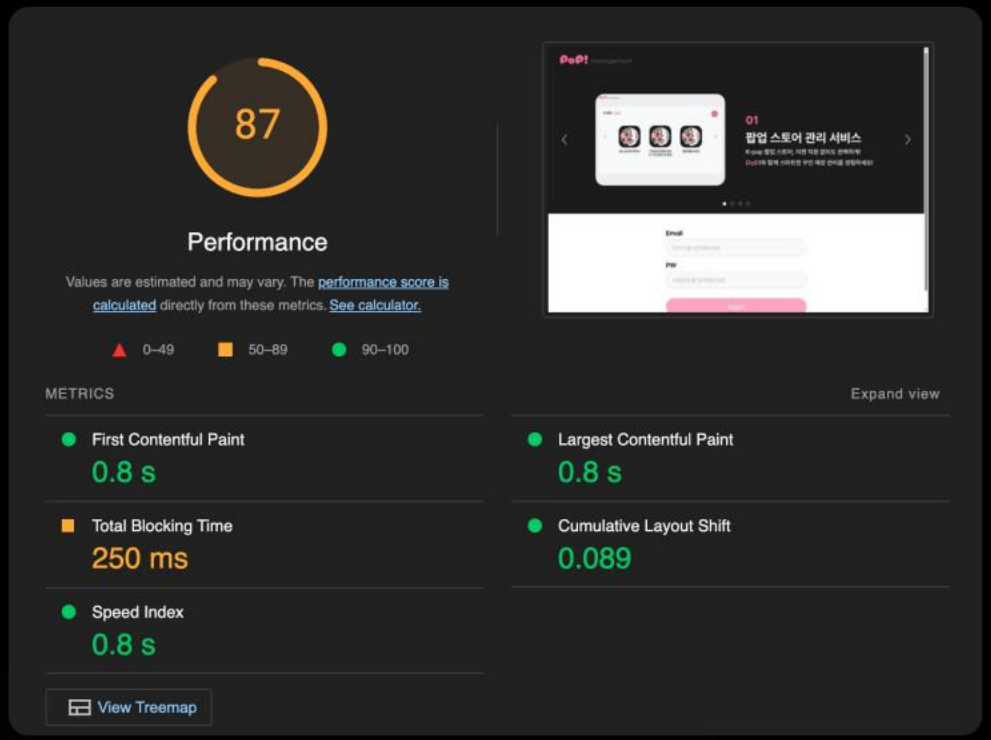
CI 과정

- ubuntu 설치
- Lighthouse CI 설치
- Chrominum 설치
- 성능 측정
- 결과 도출

02 성능 측정



≠



! 서로 다른 Git Action에서는 동일한 코드여도 다른 측정값이 도출되는 문제

로컬 테스트 5회 이후 평균

03 성능 & UX 개선

How to Upgrade?

온보딩 & 대시보드 최적화

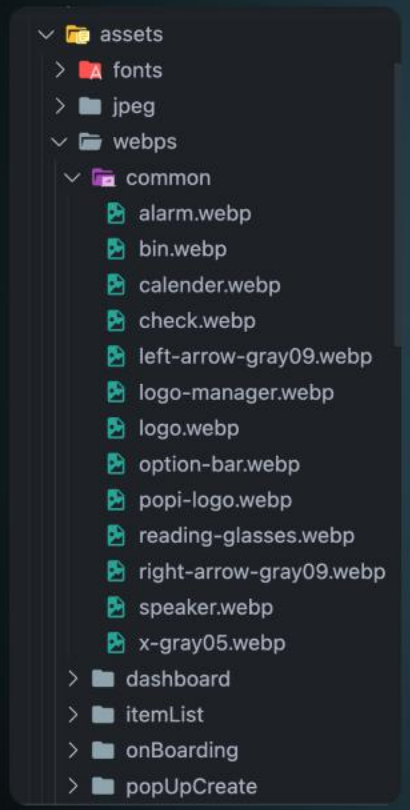
주요 페이지 성능 개선을 어떻게 했을까?

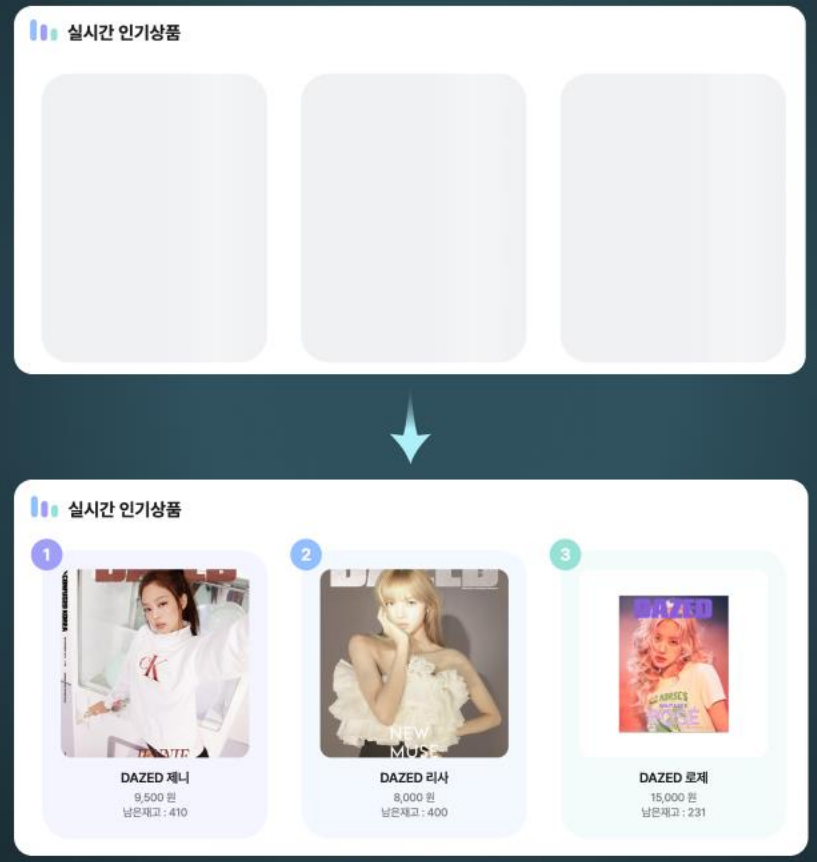
메인 배너 이미지를 PNG(850KB)

→ WebP(280KB)로 변환하여 67% 용량 감소

모든 이미지에 width, height 속성 추가로 레이아웃 시프트 방지

로딩 중 skeleton UI 적용으로 시각적 안정성 확보





03 성능 & UX 개선

	LCP	CLS
온보딩 페이지	1.4s → 1.2s	0.124 → 0.1
	14.3% 개선 (0.2초 단축)	**19.4%** 개선 (0.024 감소)
대시보드 페이지	2.3s → 1.0s	0.265 → 0.1
	56.5% 개선 (1.3초 단축)	**62.3%** 개선 (0.165 감소)

03 성능 & UX 개선

How to Upgrade?

캐시 전략 수립

TanStack Query의 QueryKey를 모듈화하여 구성

Mutation 성공 시 invalidateQueries 트리거
→ 데이터 일관성 보장

1. 캐시 정책 개요

이 문서는 각 기능 및 페이지별 캐시 처리 전략을 정의합니다.

2. 캐시 키 구조

```
// Example (페이지명, 섹션명 (Optional), 팝업 아이디, 나머지 아이디 ...
const CACHEKEY = ["Dashboard", "BestItems", 1]
```

3. 캐시 무효화 트리거 [관리자]

페이지/기능	트리거 조건	영향받는 캐시 키	무효화 방식
로그인	로그인 API 호출 및 성공 응답	all	즉시 삭제
팝업 등록	팝업 등록 API 호출 및 성공 응답	["popupList"]	즉시 삭제
팝업 삭제	팝업 삭제 API 호출 및 성공 응답	["popupList"]	즉시 삭제
상품 등록	상품 등록 API 호출 및 성공 응답	["itemList"]	즉시 삭제
상품 전체 등록	상품 전체 등록 API 호출 및 성공 응답	["itemList"]	즉시 삭제
상품 삭제	상품 삭제 API 호출 및 성공 응답	["itemList"]	즉시 삭제
상품 수정	상품 수정 API 호출 및 성공 응답	["itemList"]	즉시 삭제

4. 캐시 무효화 트리거 [사용자]

페이지/기능	트리거 조건	영향받는 캐시 키	무효화 방식
로그인	로그인 API 호출 및 성공 응답	all	즉시 삭제
팝업 예약	팝업 예약 API 호출 및 성공 응답	["reservation"]	즉시 삭제

AS-IS

기존 발주 알림은 어떻게 진행됐나요?



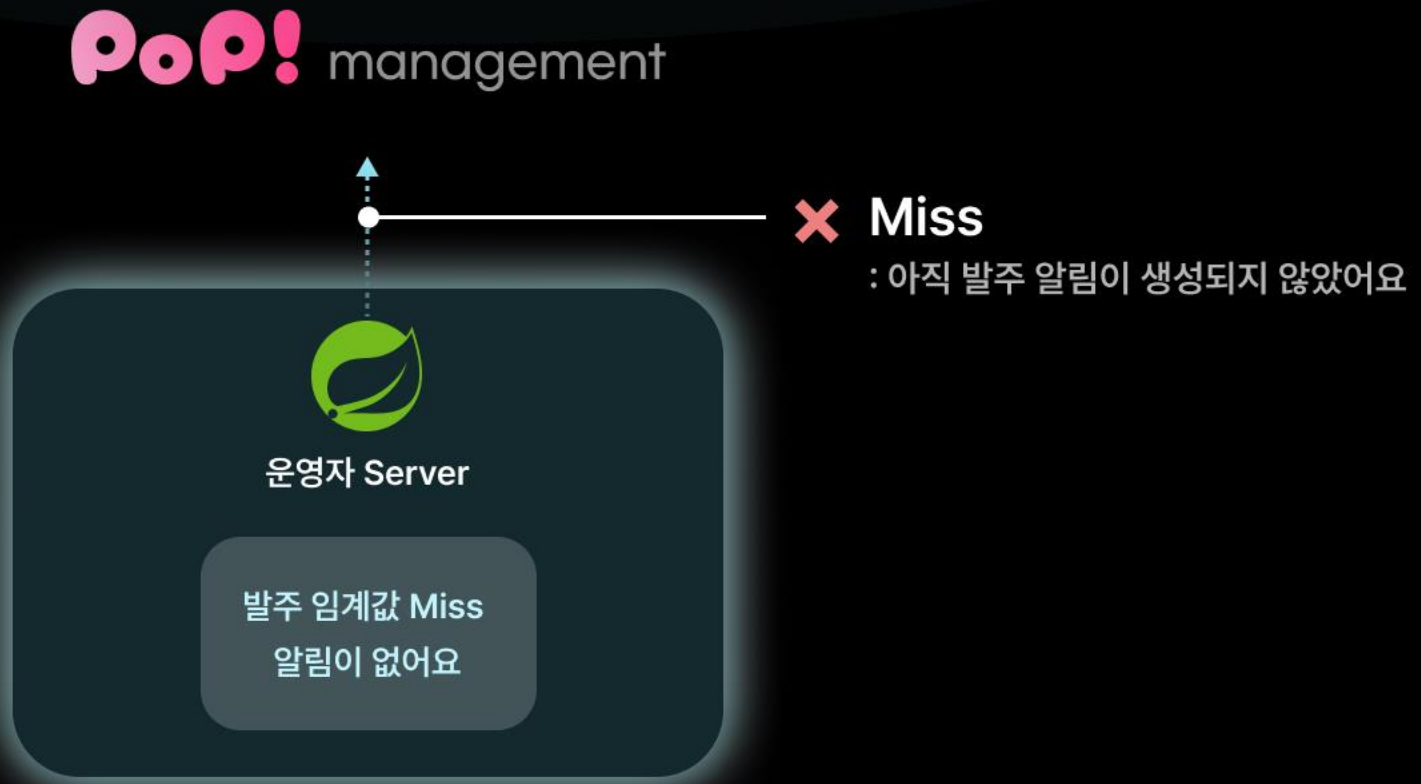
운영자 웹
: POPI management는 웹사이트로 운영돼요

✓ **Polling**
: 일정한 주기로 서버와 응답을 주고 받는 방식이에요

✓ **10분**에 한 번씩 polling 방식으로 API를 요청해요

AS-IS

기존 발주 알림은 어떻게 진행됐나요?



03 성능 & UX 개선

AS-IS

기존 발주 알림은 어떻게 진행됐나요?

사용자 어플에서 결제를 진행해요



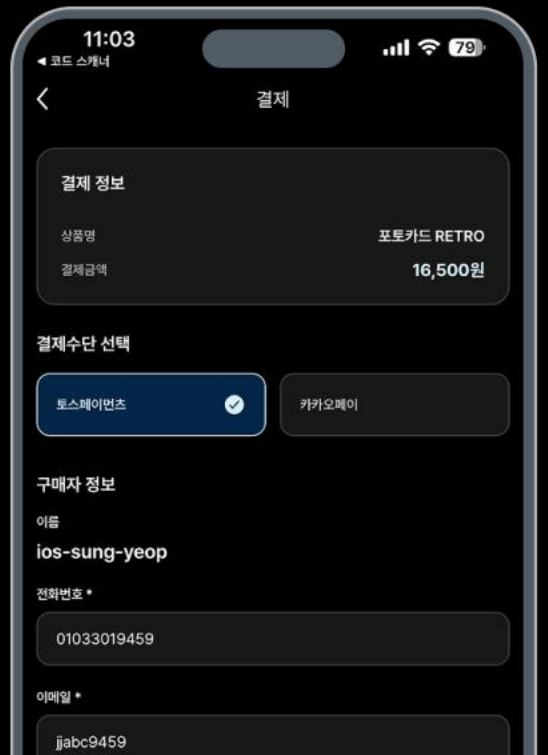
PoP! management

운영자 Server

AS-IS

기존 발주 알림은 어떻게 진행됐나요?

PoP! management



운영자 Server

발주 임계값 Hit
알림이 생성되었어요

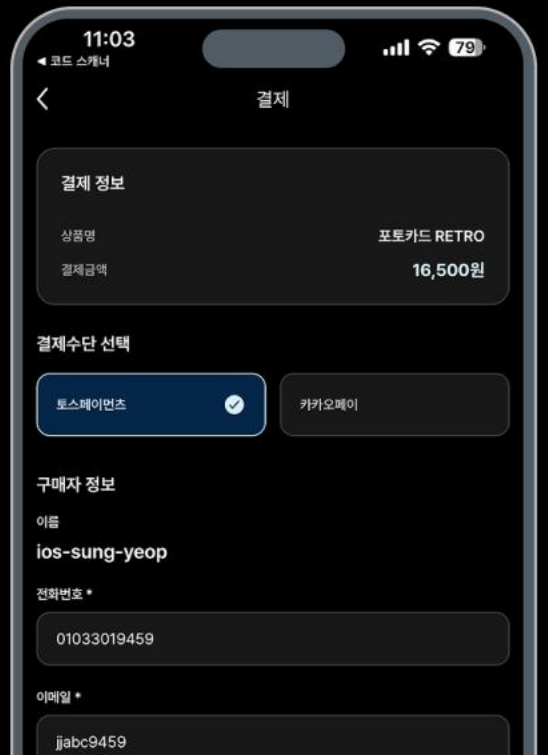
✔ **Hit**
: 발주 알림이 생성되었어요

✖ **No API**
: 아직 클라이언트에서 API 요청이 오지 않았어요

03 성능 & UX 개선

AS-IS

기존 발주 알림은 어떻게 진행됐나요?



PoP! management

운영자 Server

발주 임계값 Hit
알림이 생성되었어요

문제 인식

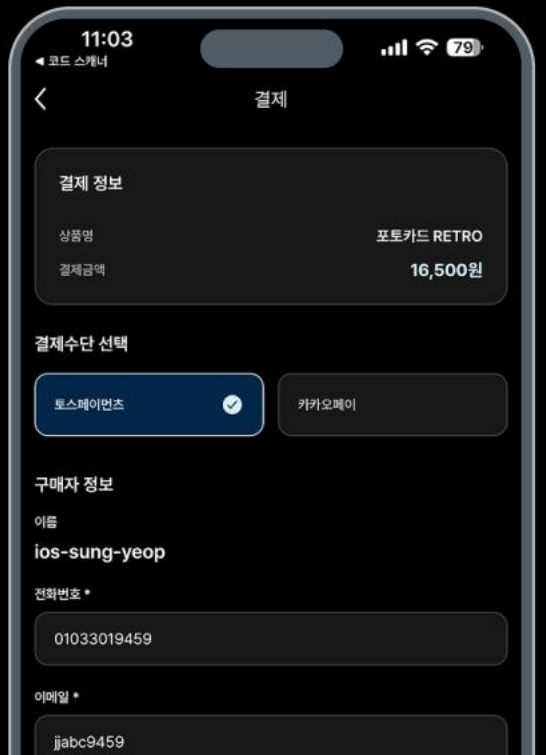
실시간성을 반영하지 못해요

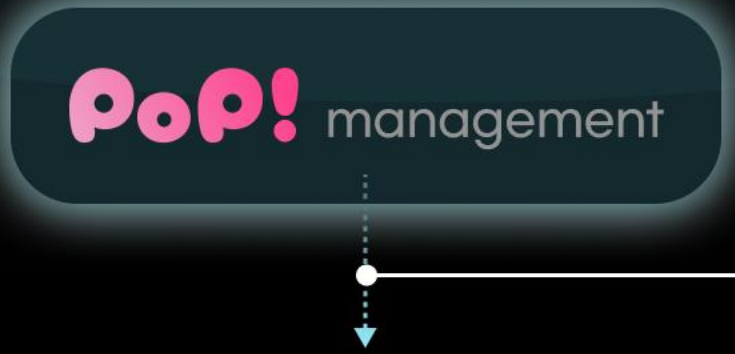
사용자 경험이 저하돼요

03 성능 & UX 개선

TO-BE

API 요청 + Web Socket 도입





운영자 Server

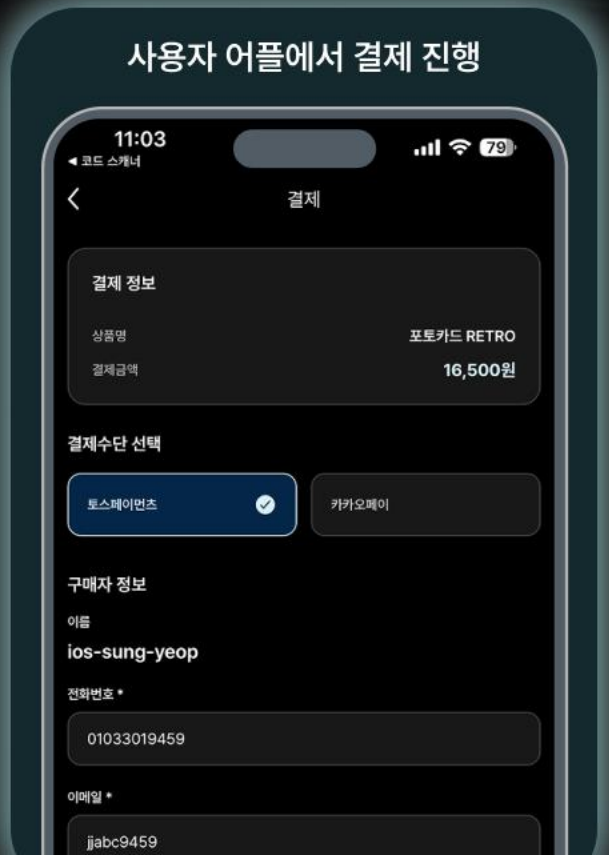
✓ **알림 API 호출**

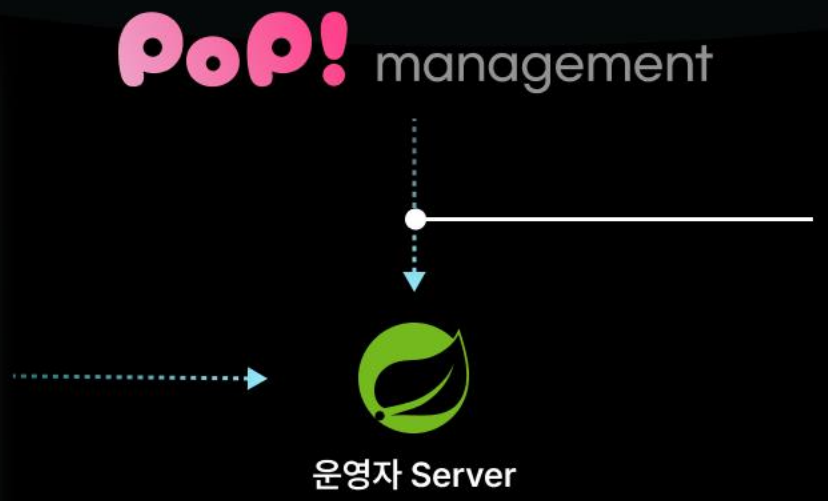
: 기존에 누적된 알림을 API 요청을 통해 제공해요

03 성능 & UX 개선

TO-BE

API 요청 + Web Socket 도입





✓ **알림 API 호출**

: 기존에 누적된 알림을 API 요청을 통해 제공해요

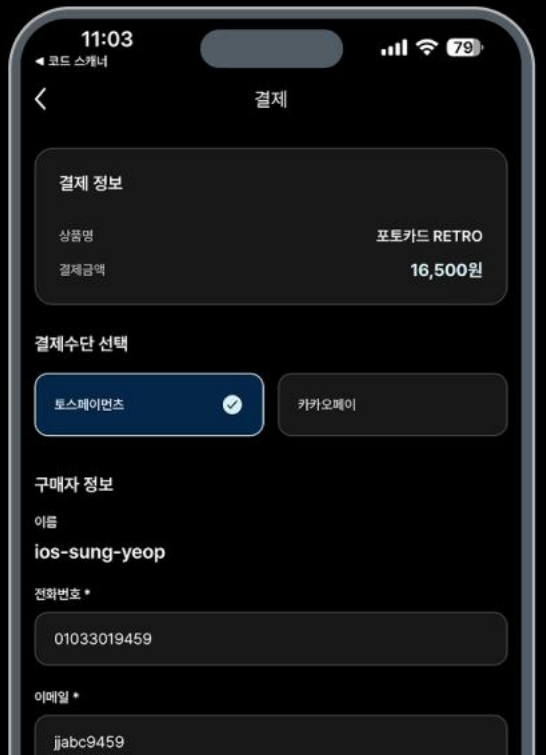
✓ **웹소켓 도입**

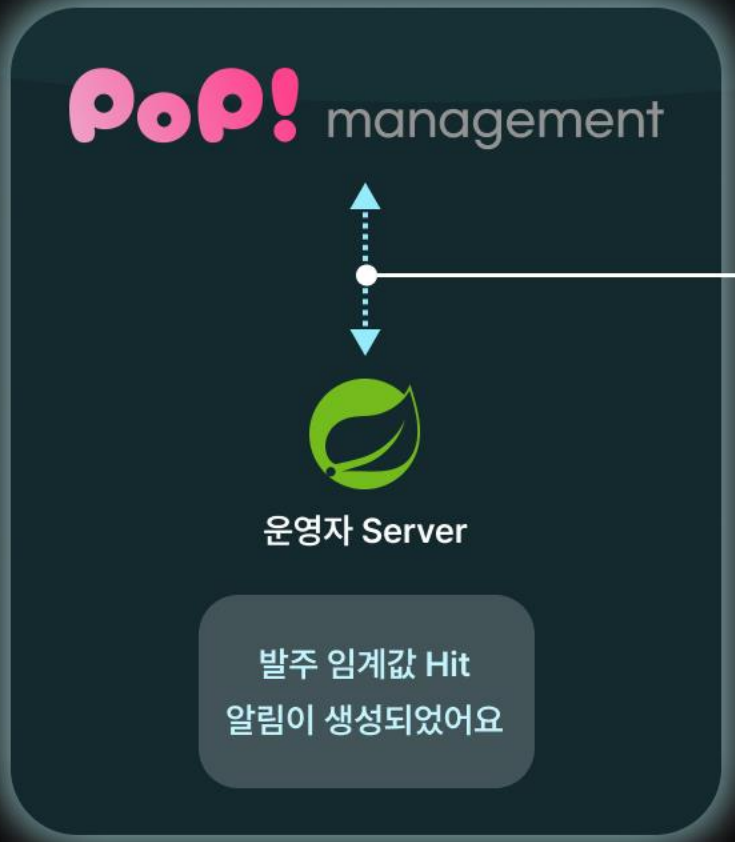
: 새로운 알림이 발생되면 실시간으로 확인할 수 있어요

03 성능 & UX 개선

TO-BE

API 요청 + Web Socket 도입



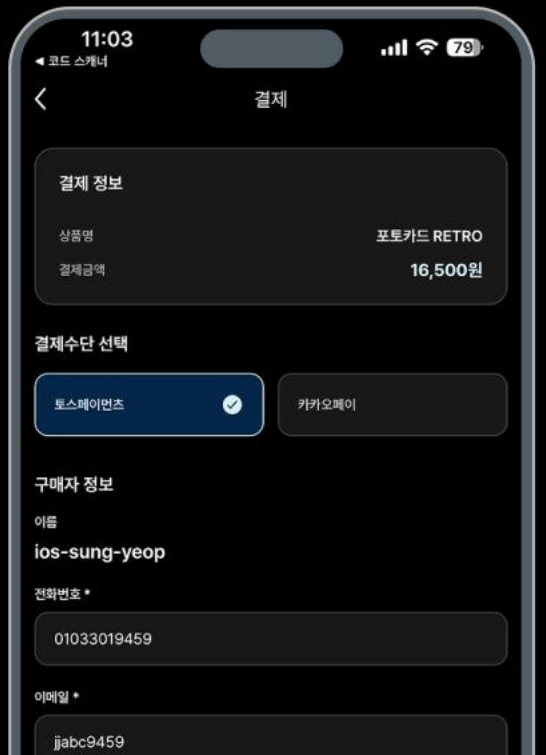


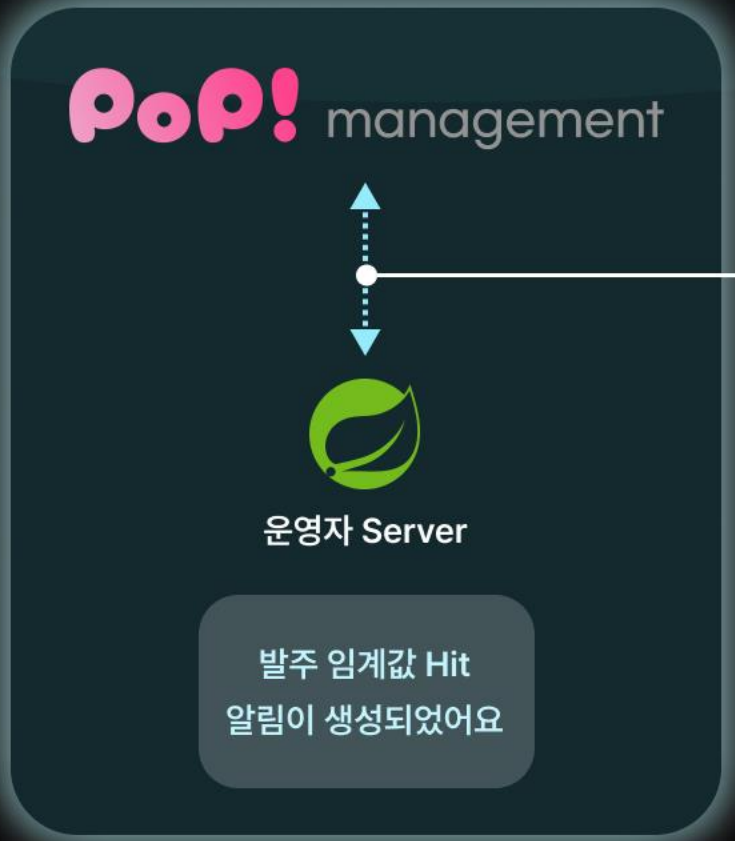
- **알림 API 호출**
  : 기존에 누적된 알림을 API 요청을 통해 제공해요
- **웹소켓 도입 (양방향 통신)**
  : 새로운 알림이 발생되면 실시간으로 확인할 수 있어요

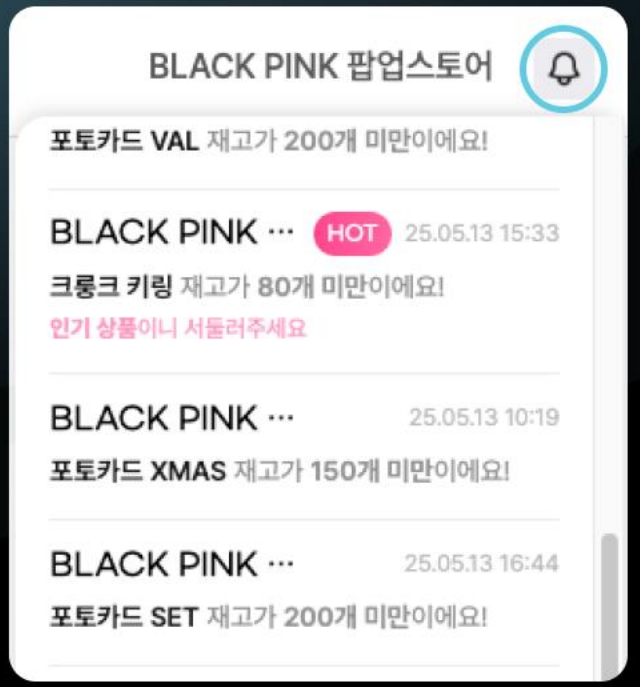
03 성능 & UX 개선

TO-BE

API 요청 + Web Socket 도입







- **알림 API 호출**
  : 기존에 누적된 알림을 API 요청을 통해 제공해요
- **웹소켓 도입 (양방향 통신)**
  : 새로운 알림이 발생되면 실시간으로 확인할 수 있어요

04 E2E Test

How to Test?

Playwright 도입

Test Target

데이터가 저장되어있는 개발 서버

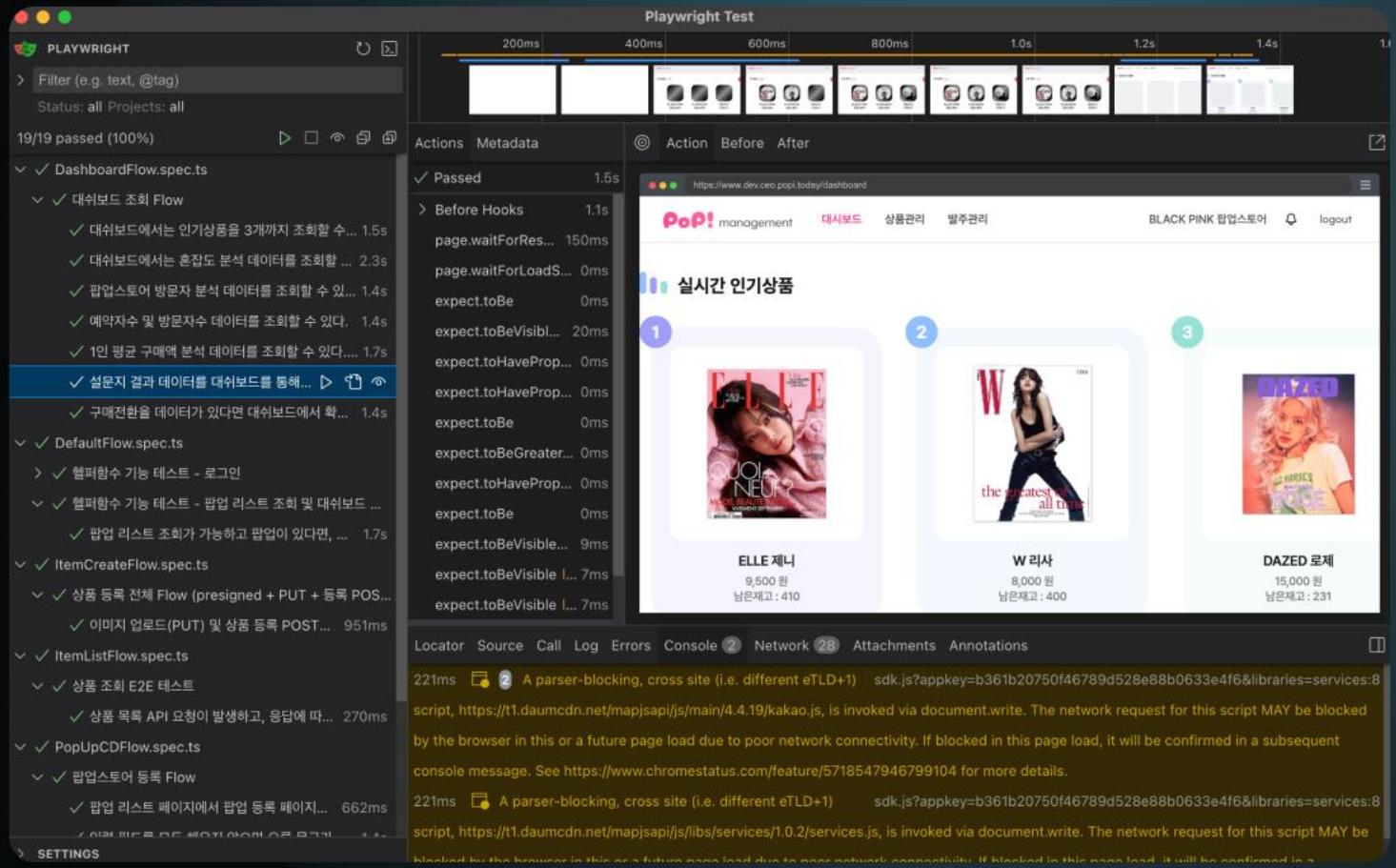
↓

실제 환경과 최대한 비슷한 환경에서 테스트 진행

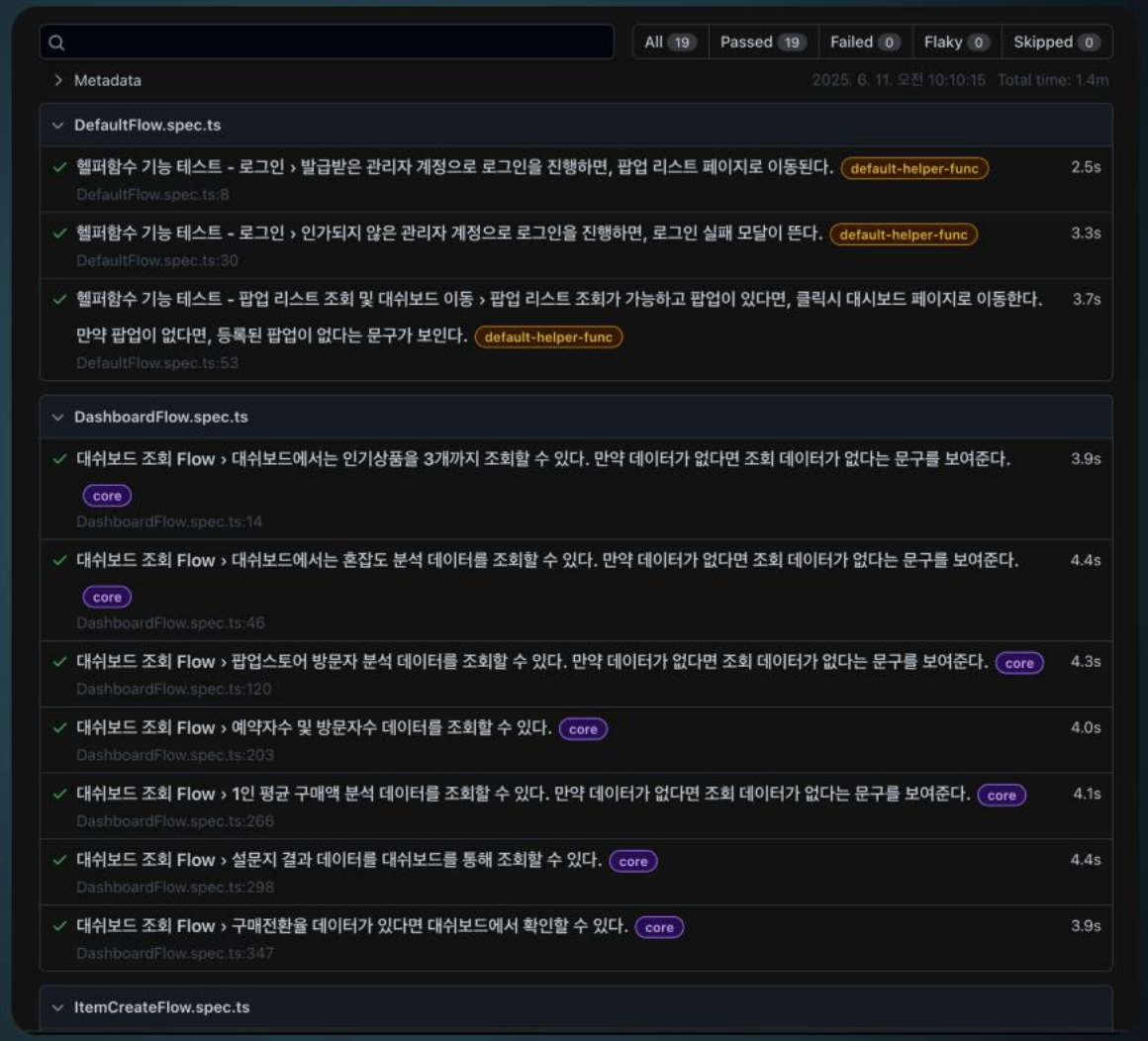
Test Senario

- 로그인
- 팝업 리스트 선택
- 팝업 등록 & 삭제
- 상품 조회
- 상품 등록

총 5개의 운영자 메인 시나리오에 대한 E2E 테스트 진행

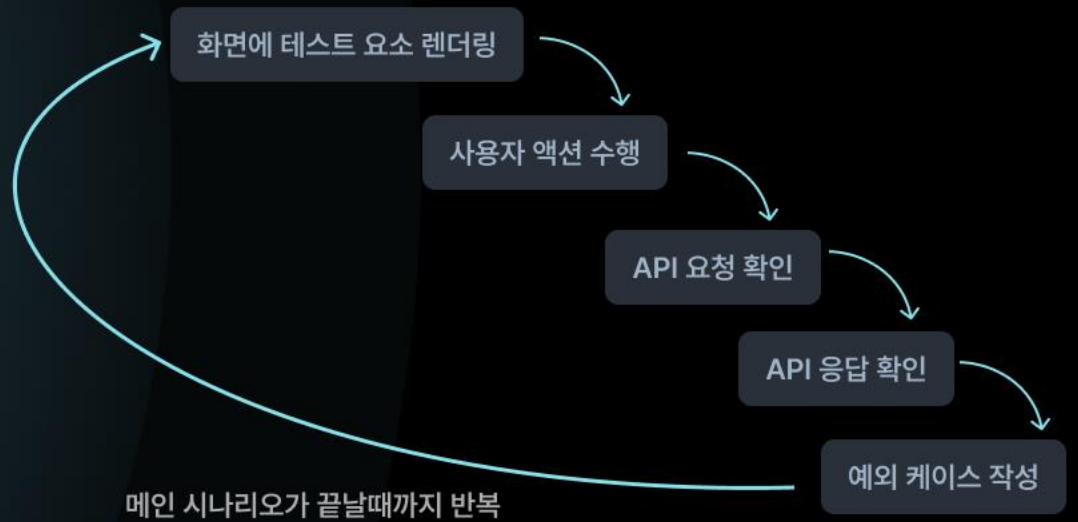


04 E2E Test



E2E 테스트 구성

각각의 시나리오는 다음과 같은 순서로 테스트를 진행해요



04

백엔드

01 서비스 & 인프라

Cloud


EKS


EC2


ELB


RDS


S3


IoT Core


Elastic Cache

Test & Code Quality



Jacoco



SonarQube


JUnit5



Mockito

Framework


Spring Boot

ORM


QueryDSL


JPA


Spring Batch

Database Migration


Flyway

Build Tool


Gradle

CI / CD


Jenkins


ArgoCD

Database


My SQL


DynamoDB

Cache & Message Queue


Redis


kafka

Authentication


Spring Security



JWT

Telementry


Prometheus


Grafana


Loki



K6

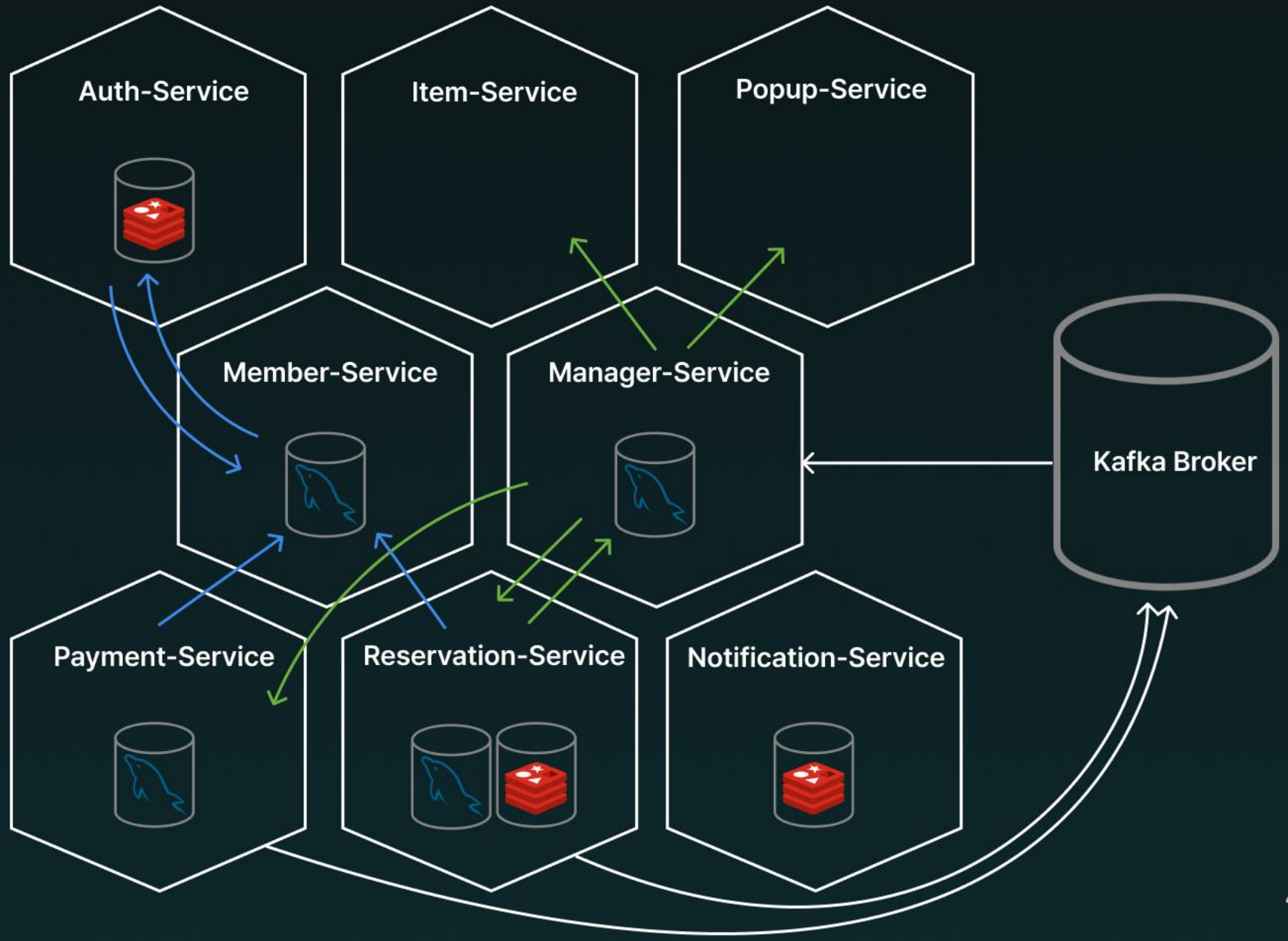

InfluxDB

01 서비스 & 인프라

Data Architecture

서비스별 데이터 아키텍처

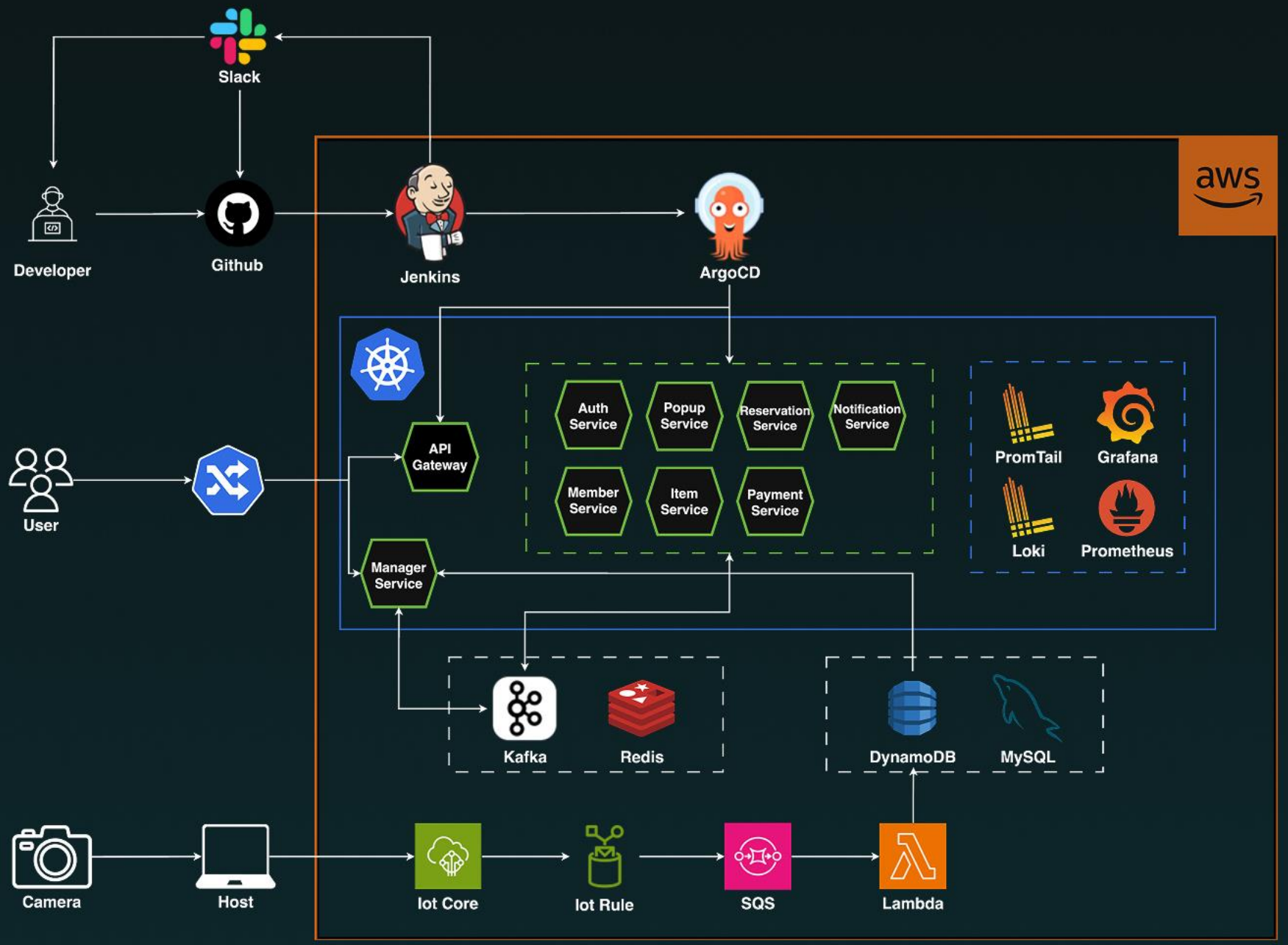




01 서비스 & 인프라

System Architecture

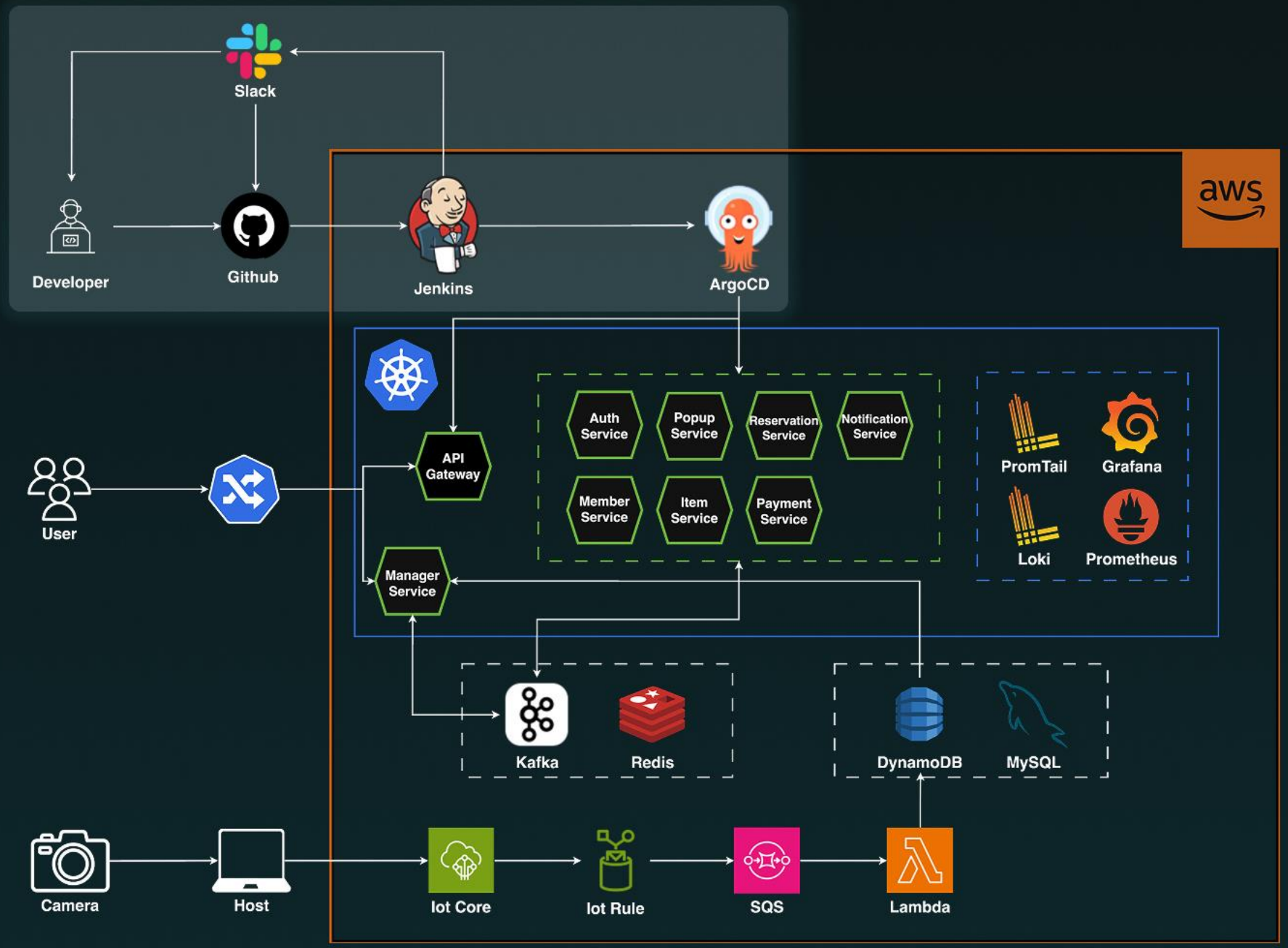
시스템 아키텍처



01 서비스 & 인프라

System Architecture

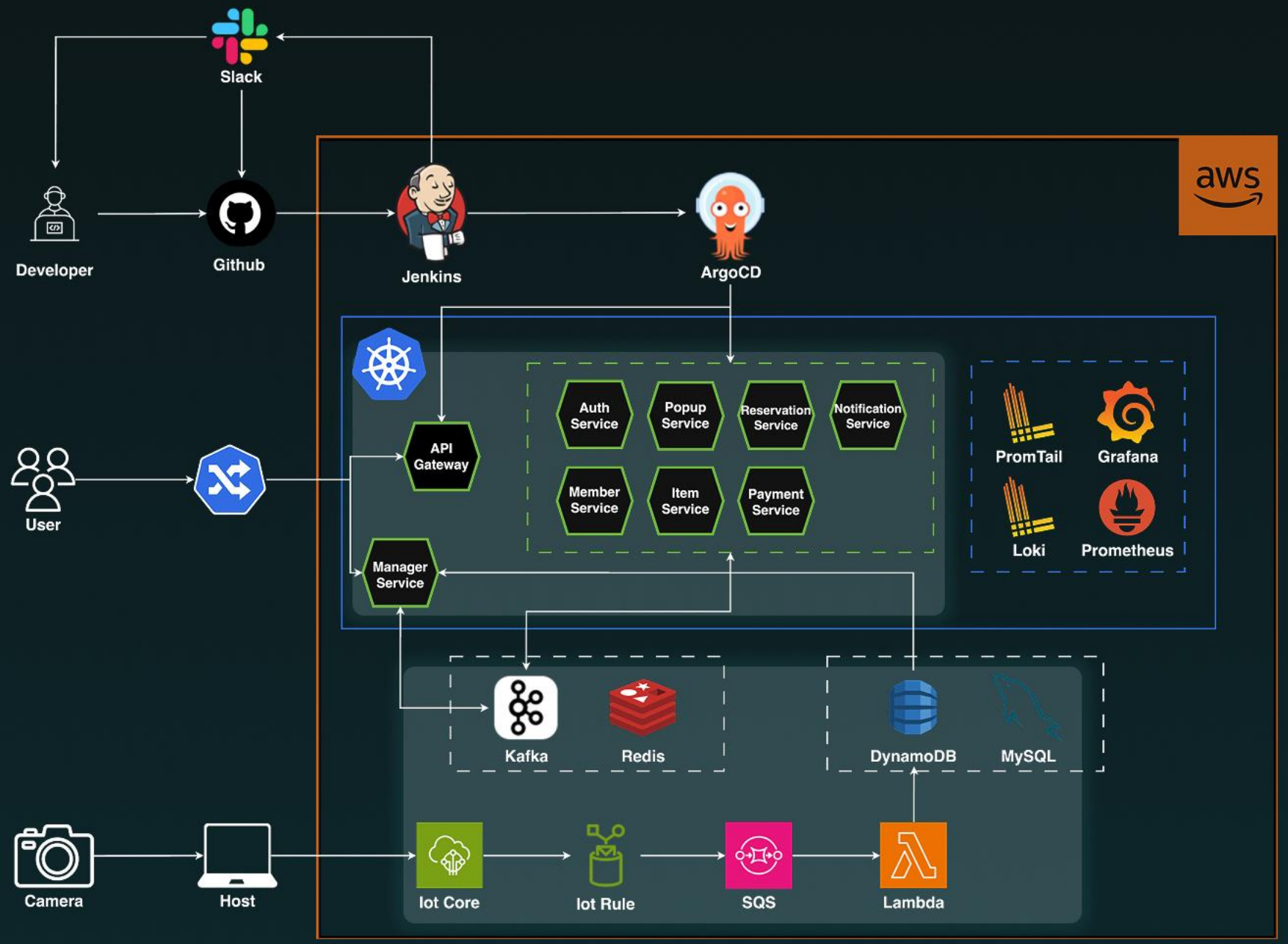
시스템 아키텍처



01 서비스 & 인프라

System Architecture

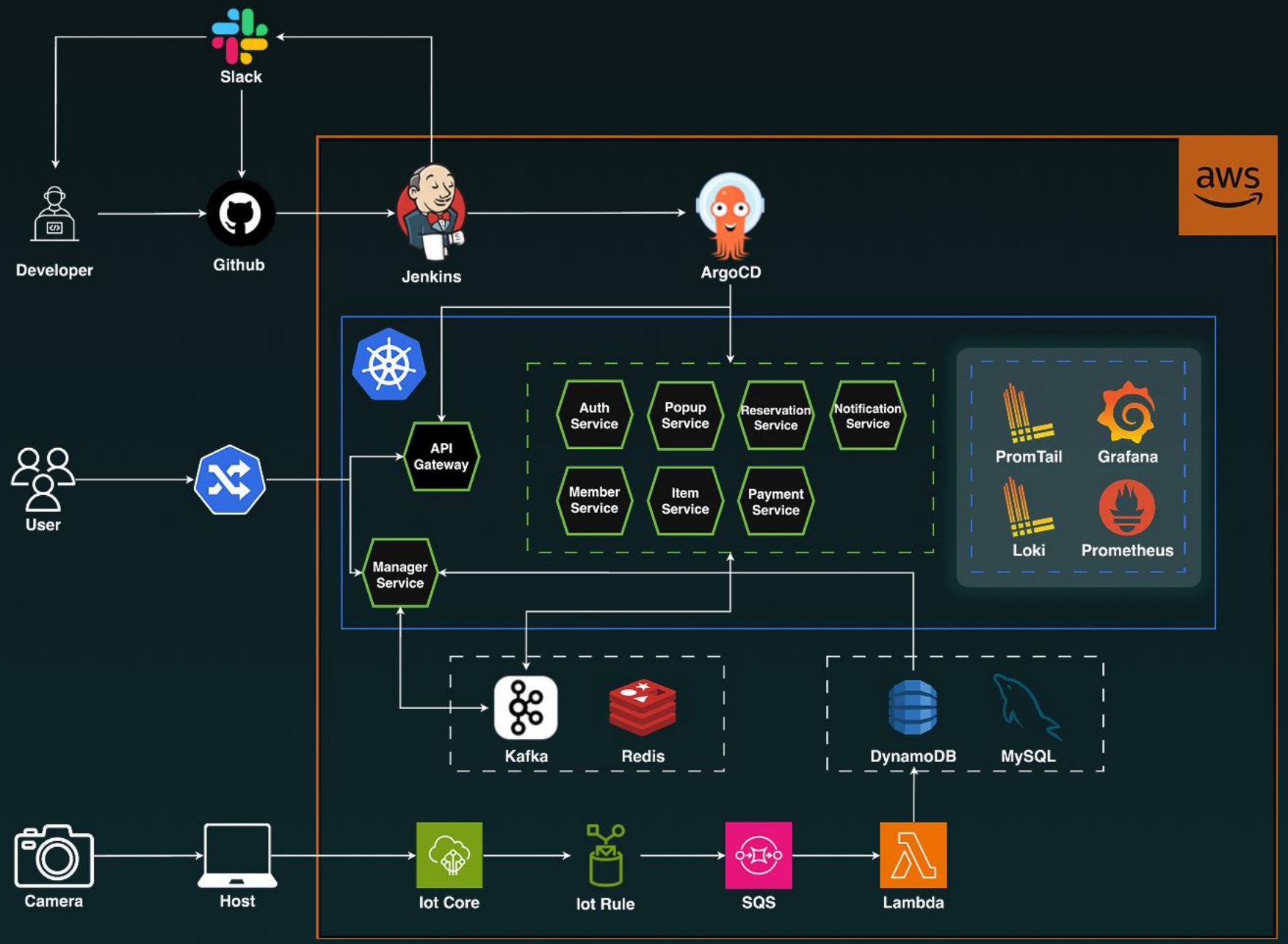
시스템 아키텍처



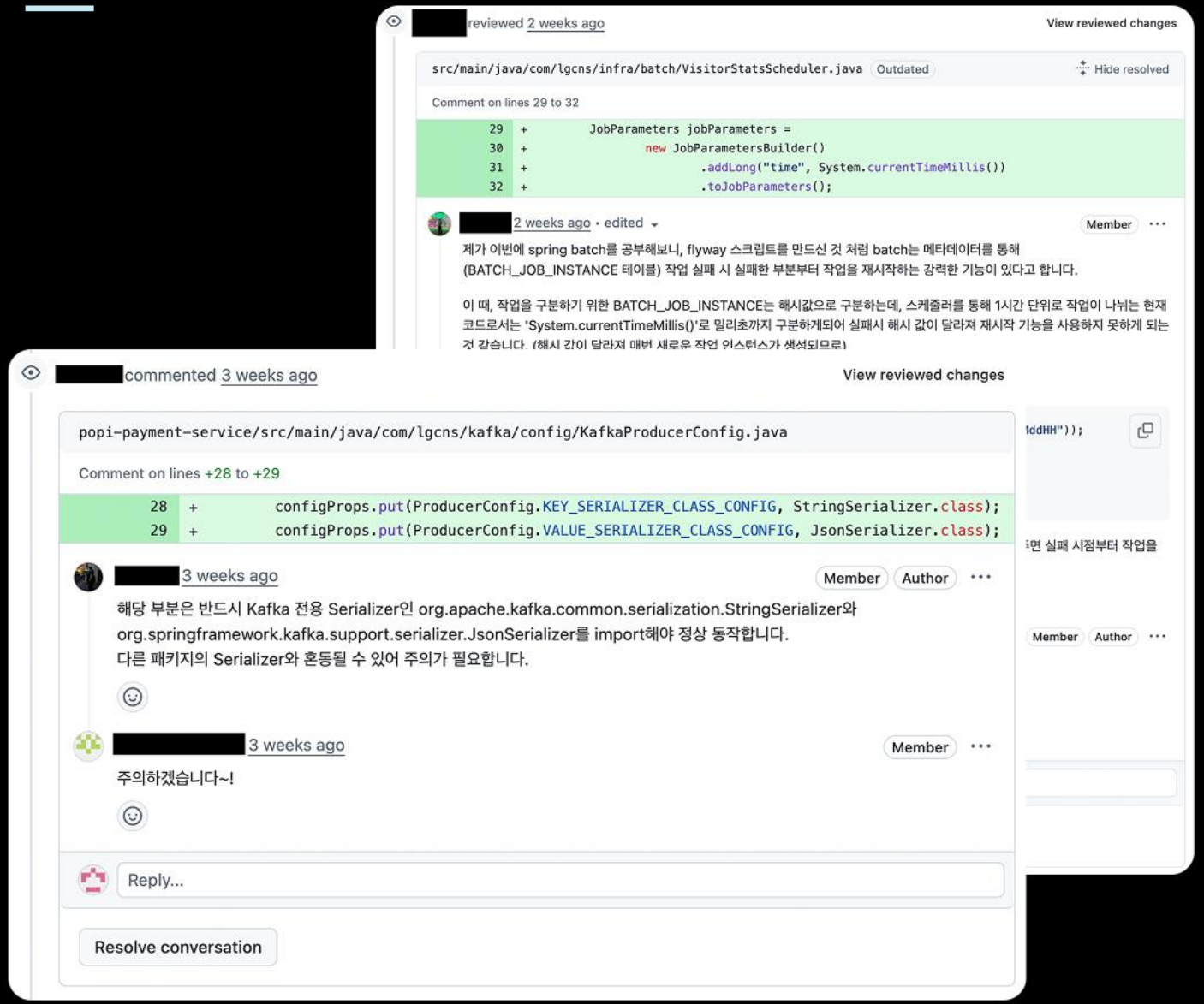
01 서비스 & 인프라

System Architecture

시스템 아키텍처



01 서비스 & 인프라



팀원 작업 효율 상승을 위한

컨벤션 룰 지정 및 코드 리뷰

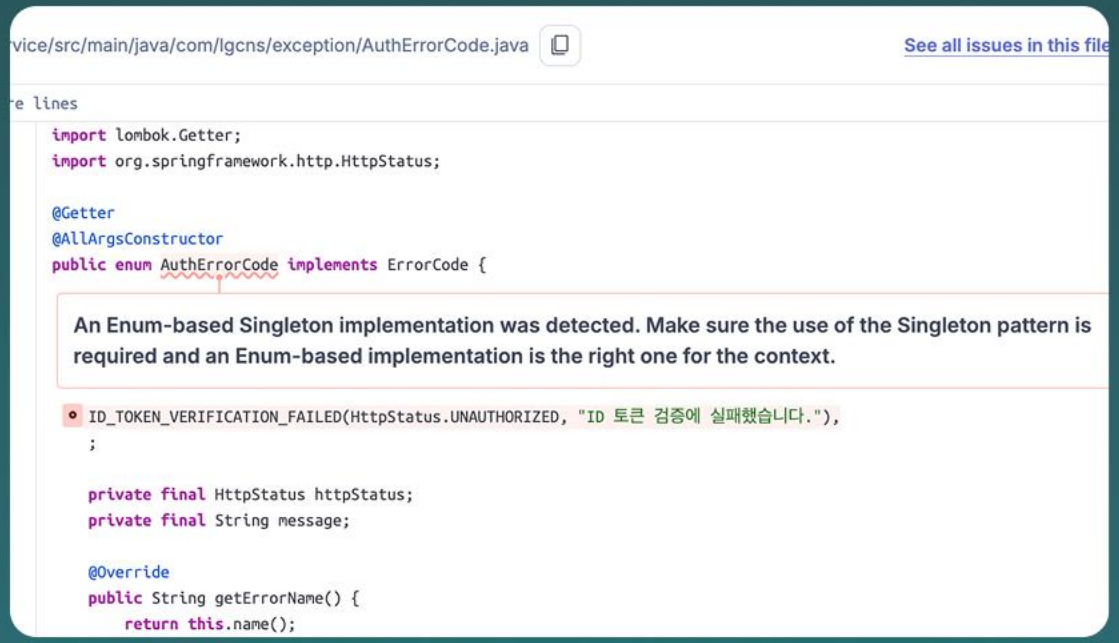
- Controller
- Service
- DTO
- 어노테이션 순서
- 정적 팩토리 메서드

01 서비스 & 인프라

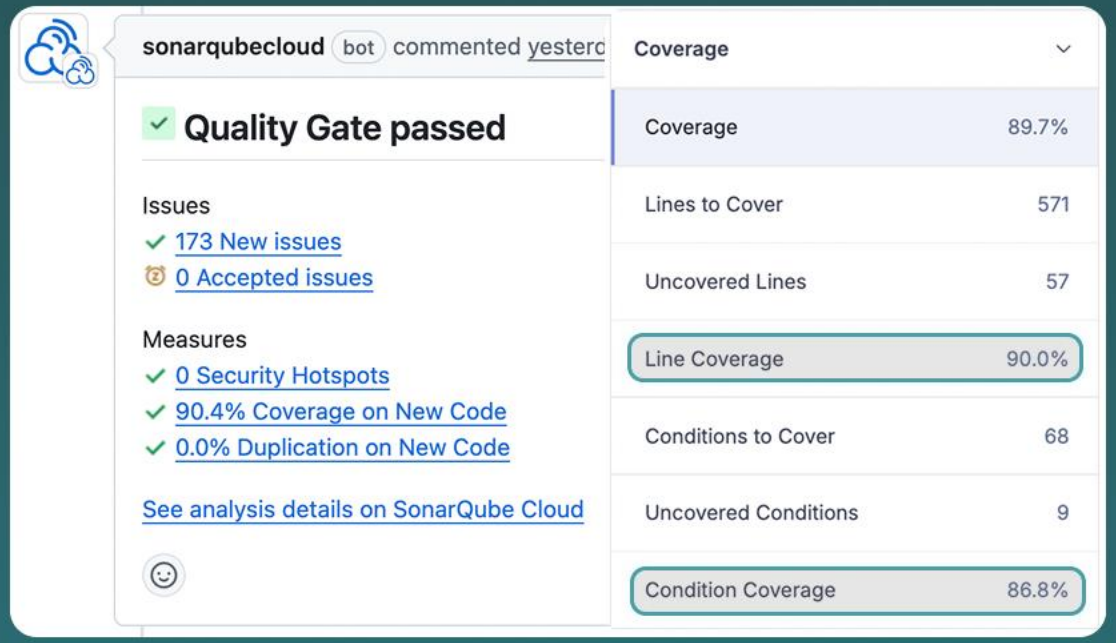
Integration

CI

Sonarqube 정적 분석



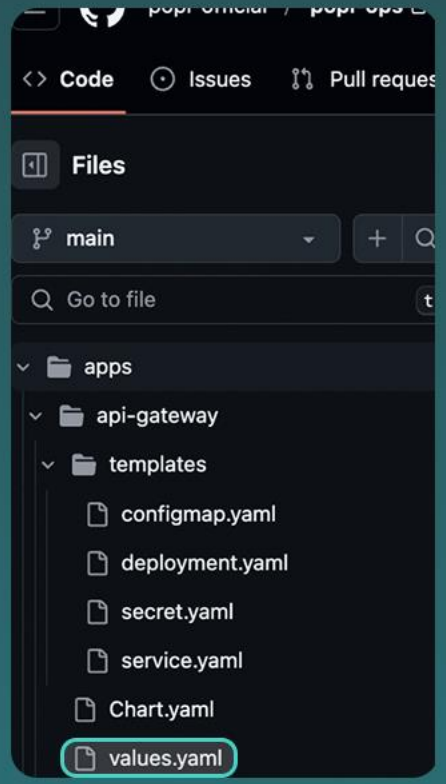
코드 커버리지 측정

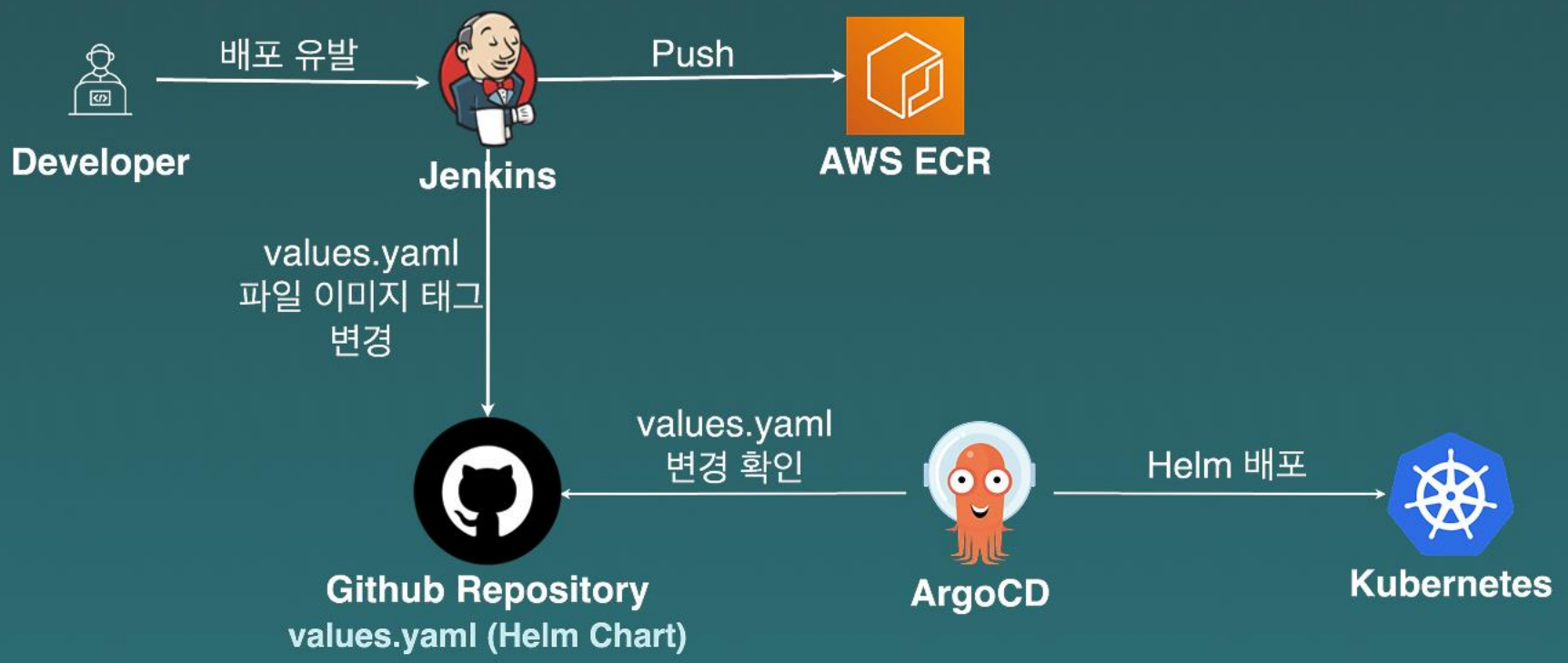


01 서비스 & 인프라

Deployment

CD



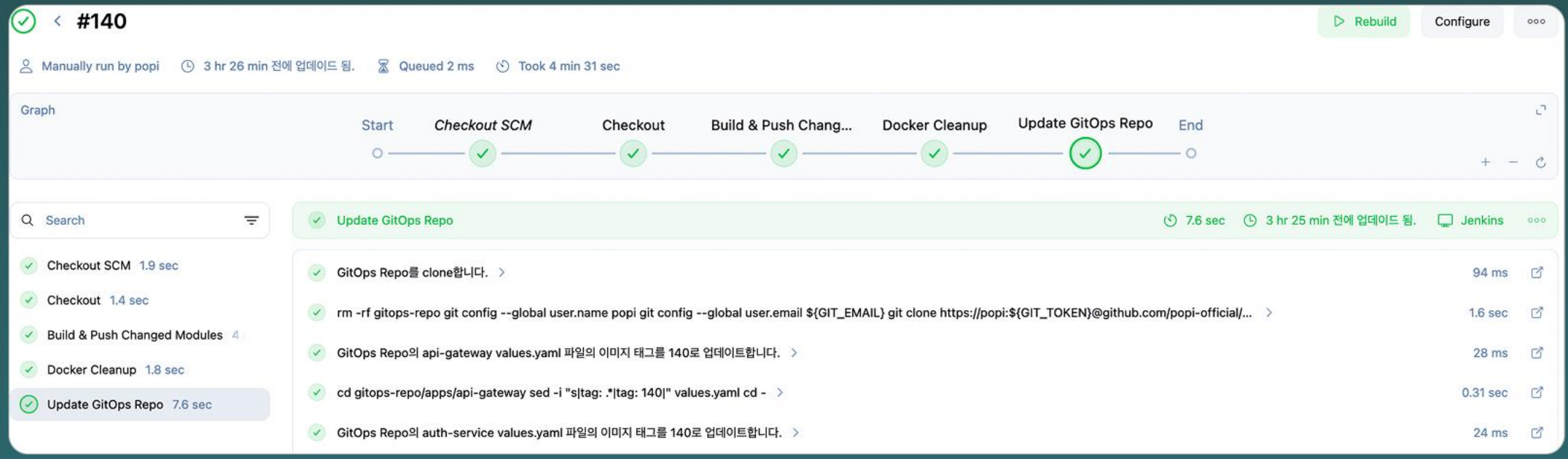


01 서비스 & 인프라

Deployment

CD

Deployment Pipeline



이미지 태그	아티팩트 유형	푸시 위치	크기 (MB)	이미지 URI	다이제스트	마지막으로 기록된 풀타임
140	Image	2025년 6월 17일, 10:07:36 (UTC+09)	299.83	URI 복사	sha256:970f5c507354abc...	2025년 6월 17일, 10:12:38 (UTC+09)

01 서비스 & 인프라

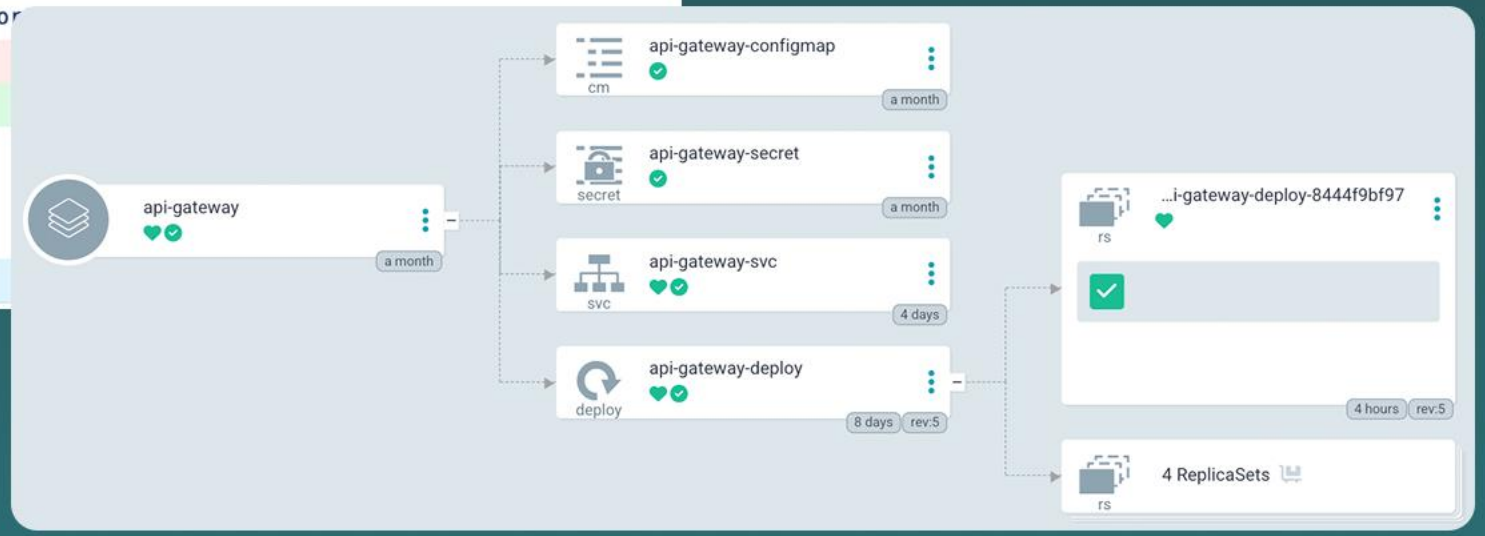
Deployment

CD

GitOps + ArgoCD

apps/api-gateway/values.yaml

```
@@ -6,7 +6,7 @@ replicaCount: 1
6  6   container:
7  7     image:
8  8       name: 446305434496.dkr.ecr.ap-nor
9     -     tag: 138
   9  +     tag: 140
10 10       pullPolicy: Always
11 11
12 12   service:
```

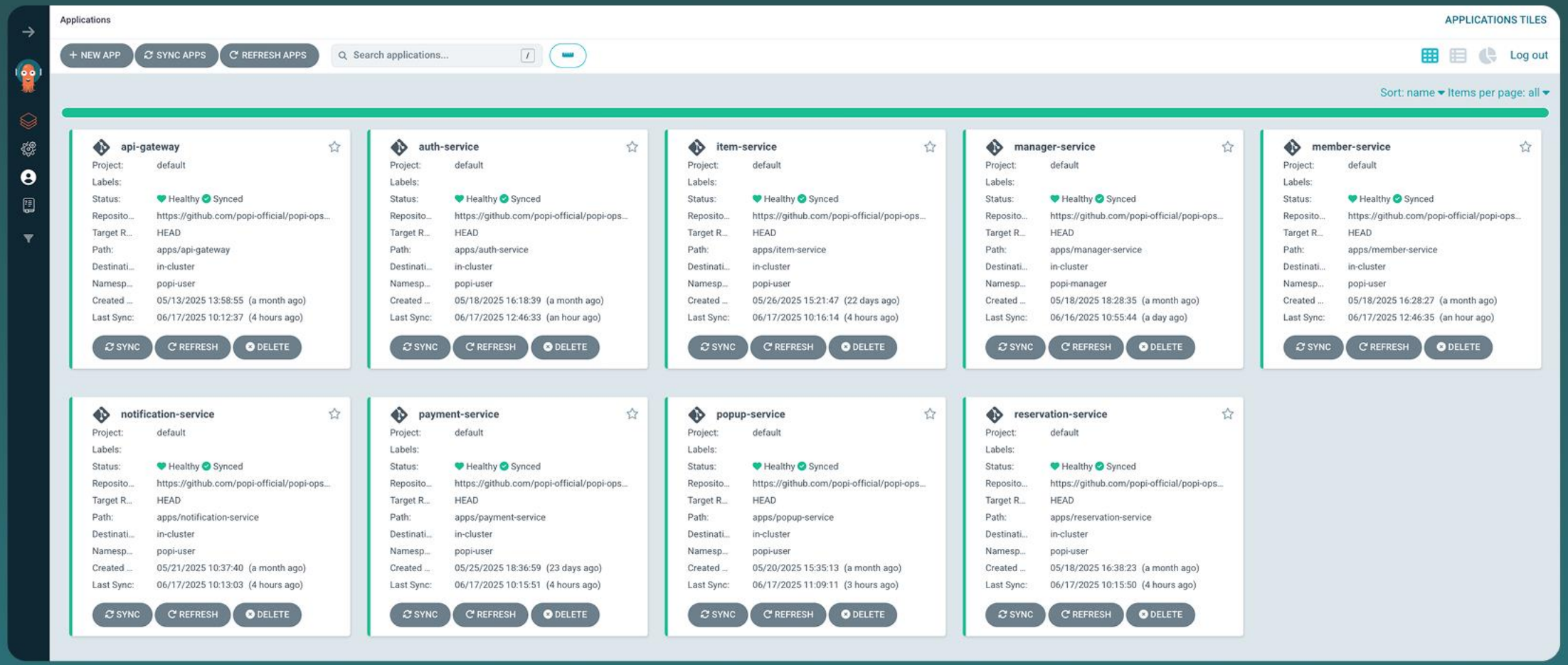


01 서비스 & 인프라

Deployment

CD

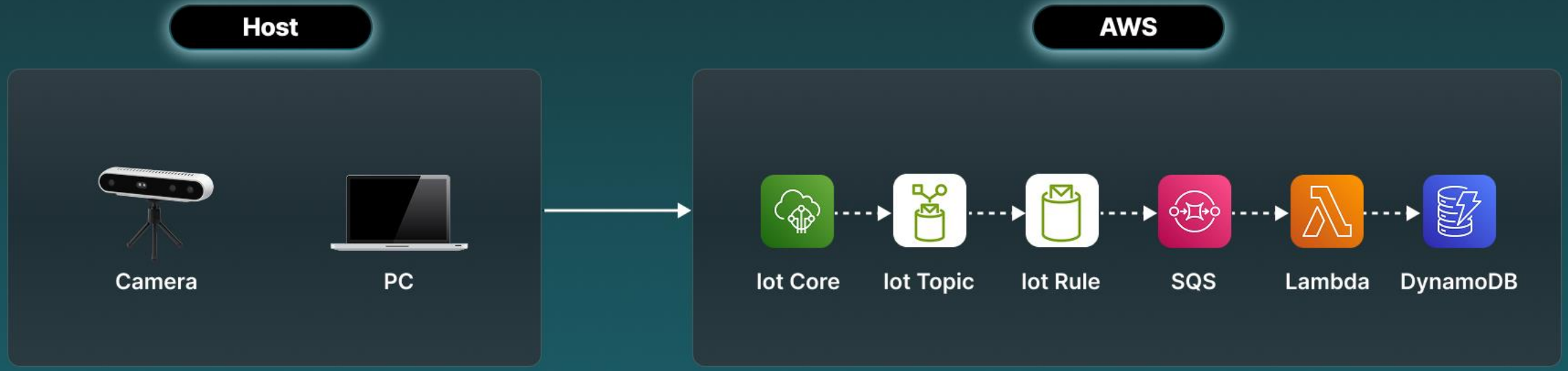
ArgoCD Dashboard



01 서비스 & 인프라

RealSense D415 Camera

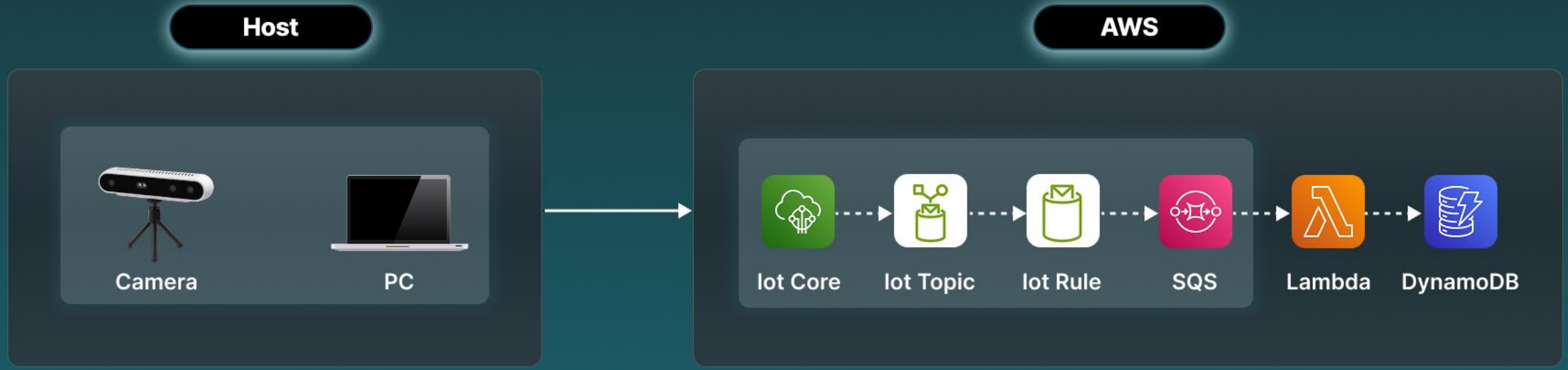
카메라 아키텍처



01 서비스 & 인프라

RealSense D415 Camera

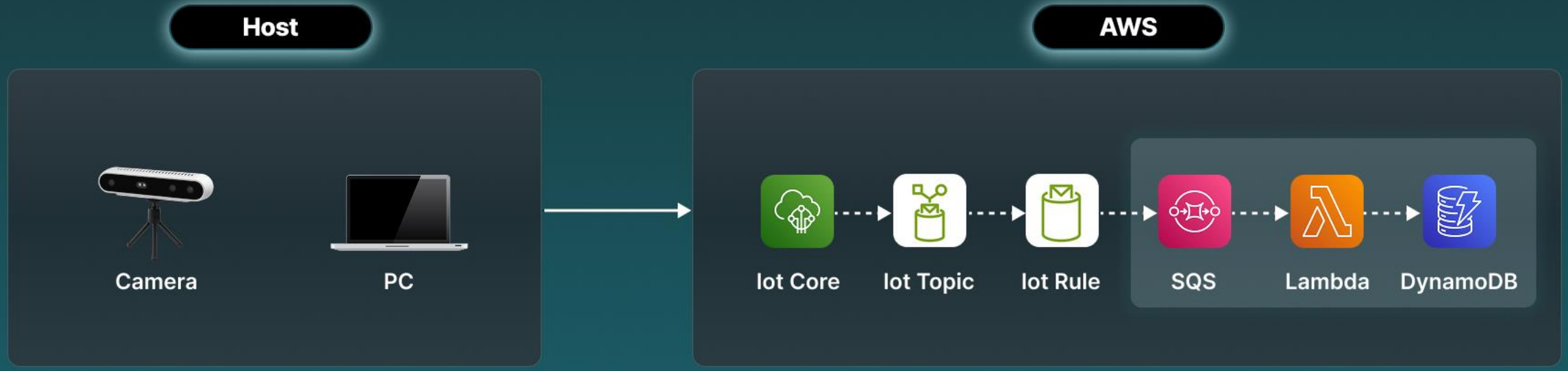
카메라 아키텍처



01 서비스 & 인프라

RealSense D415 Camera

카메라 아키텍처



How to use Data?

데이터를 어떻게 활용할까?

Proxemics 개념을 채택하여 관심도 분석 기준을 세웠어요

거리 기반 관심도 분석 테이블

영역	거리 범위	해석
Public Zone	3m 이상	우연한 관심 / 지나가는 중
Social Zone	1.2 ~ 3m	실제 상품 비교 및 고려
Personal Zone	0.5 ~ 1.2m	구매 가능성 존재
Intimate Zone	0.5m 이하	진지한 관심, 감정적 반응 가능

체류 시간 기반 관심도 분석 테이블

관심 정도	시간	해석
주의	0 - 2초	우연한 관심 / 지나가는 중
의도적 관심	3 - 5초	실제 상품 비교 및 고려
심화 관심	5초 이상	구매 가능성 존재

01 서비스 & 인프라

How to use Data?

거리 + 체류 시간 기반으로 관심도 점수를 구성했어요



관심도 점수 테이블

거리 범위	체류 시간	점수
0.5m 이하	5초 이상	10
0.5m 이하	3 - 5초	8
0.5 ~ 0.8m	5초 이상	6
0.5 ~ 0.8m	3 - 5초	4
0.8 ~ 1.2m	3 - 5초	2
0.8 ~ 1.2m	3 - 5초	0

DynamoDB

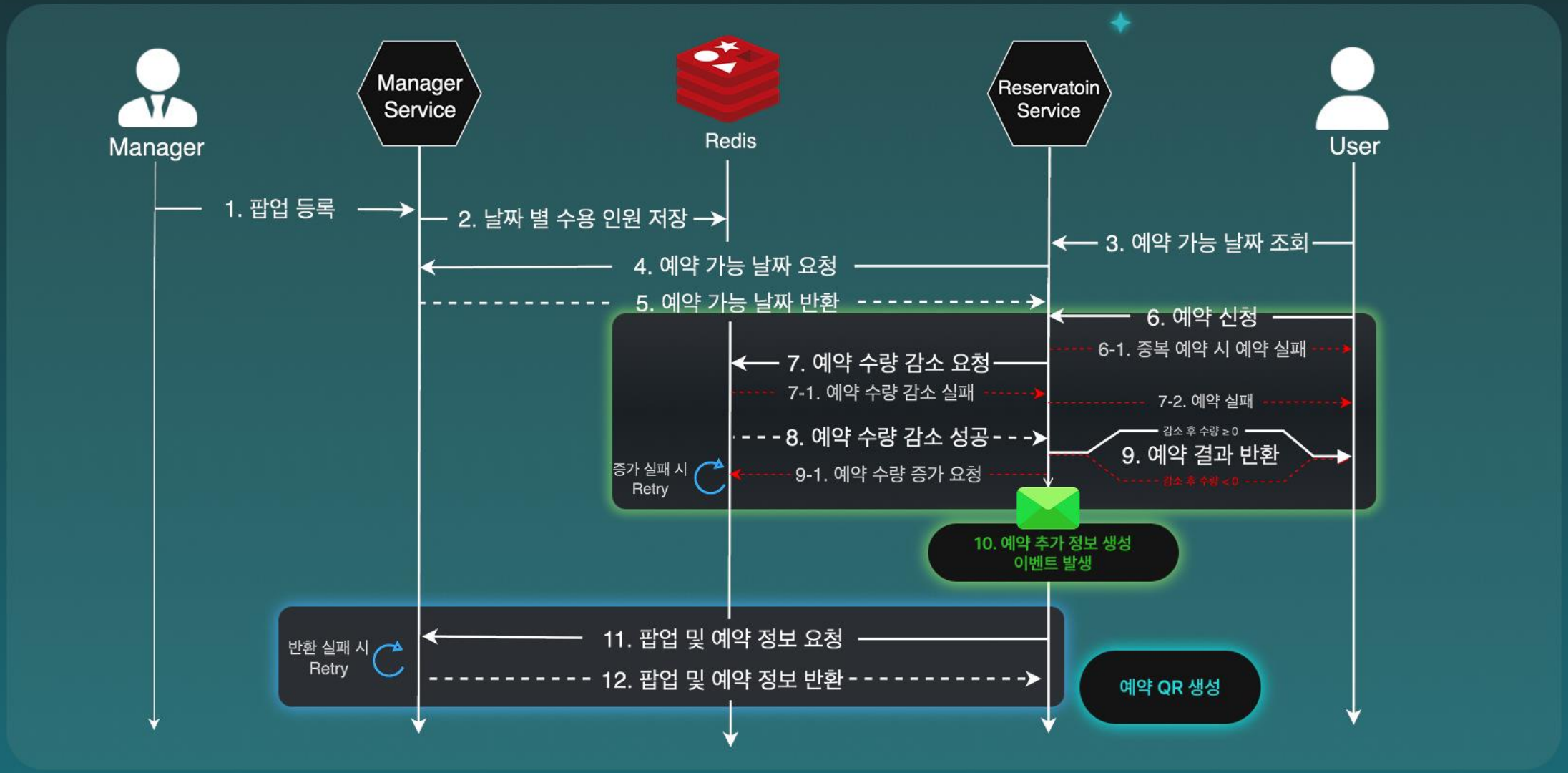
데이터는 이렇게 저장해요

항목	예시값	설명
popup_id	1	팝업스토어 ID
event_key(SK)	2025-05-12T08:00:00Z#stay#19	이벤트 구분용 복합 키
score	10	관심도 점수 (0 ~ 10)
event_type	stay	이벤트 유형
location	e3	매대
item_id	19	관심 상품 ID
distance	0.45	고객과 상품 사이 거리 (m)
duration	5.07	체류 시간 (s)
timestamp	2025-05-12T08:00:00Z	이벤트 발생 시각
device_id	REALSENSE_D415	카메라 또는 센서 ID

02 핵심 로직

Reservation

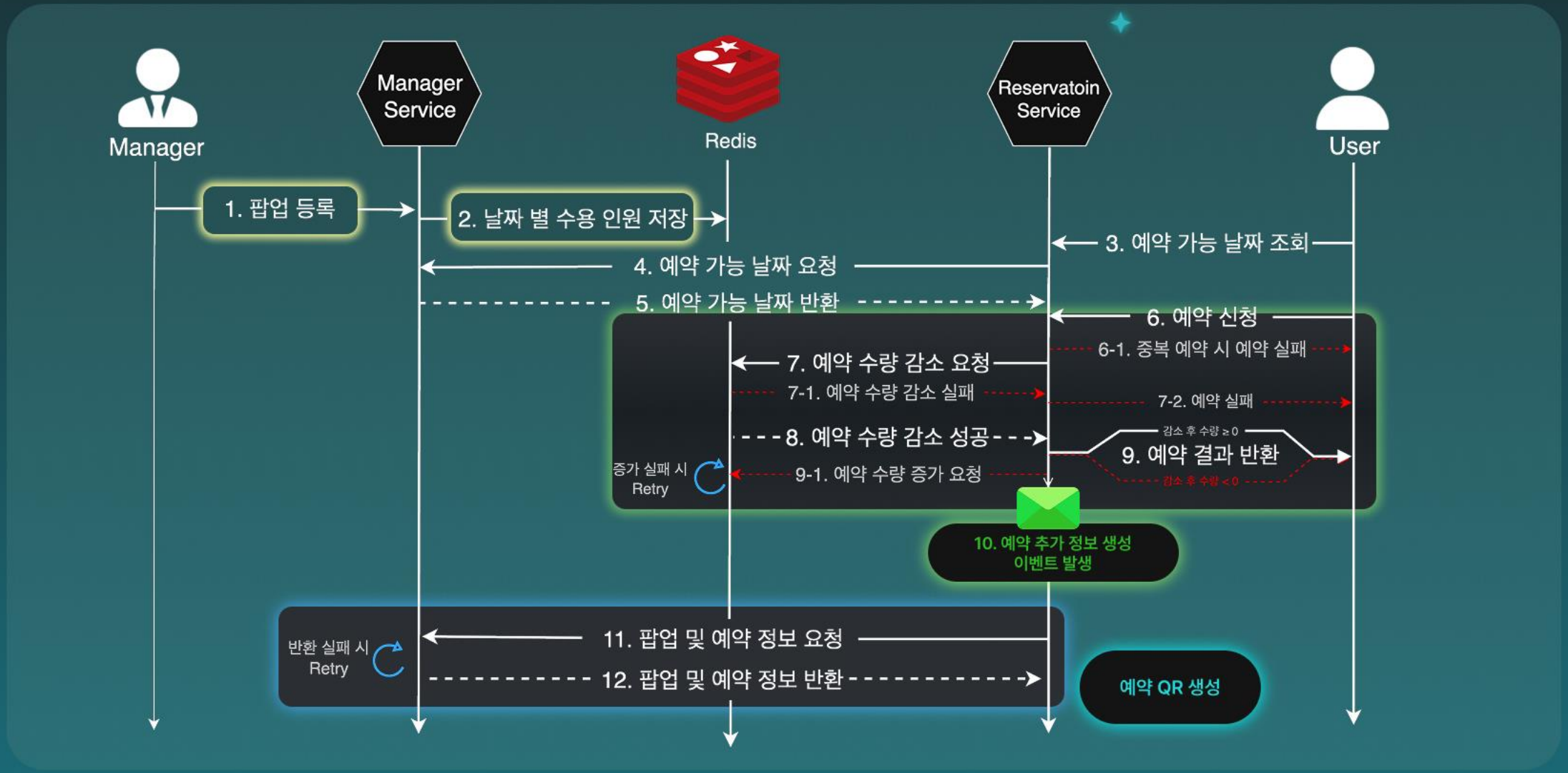
팝업 예약



02 핵심 로직

Reservation

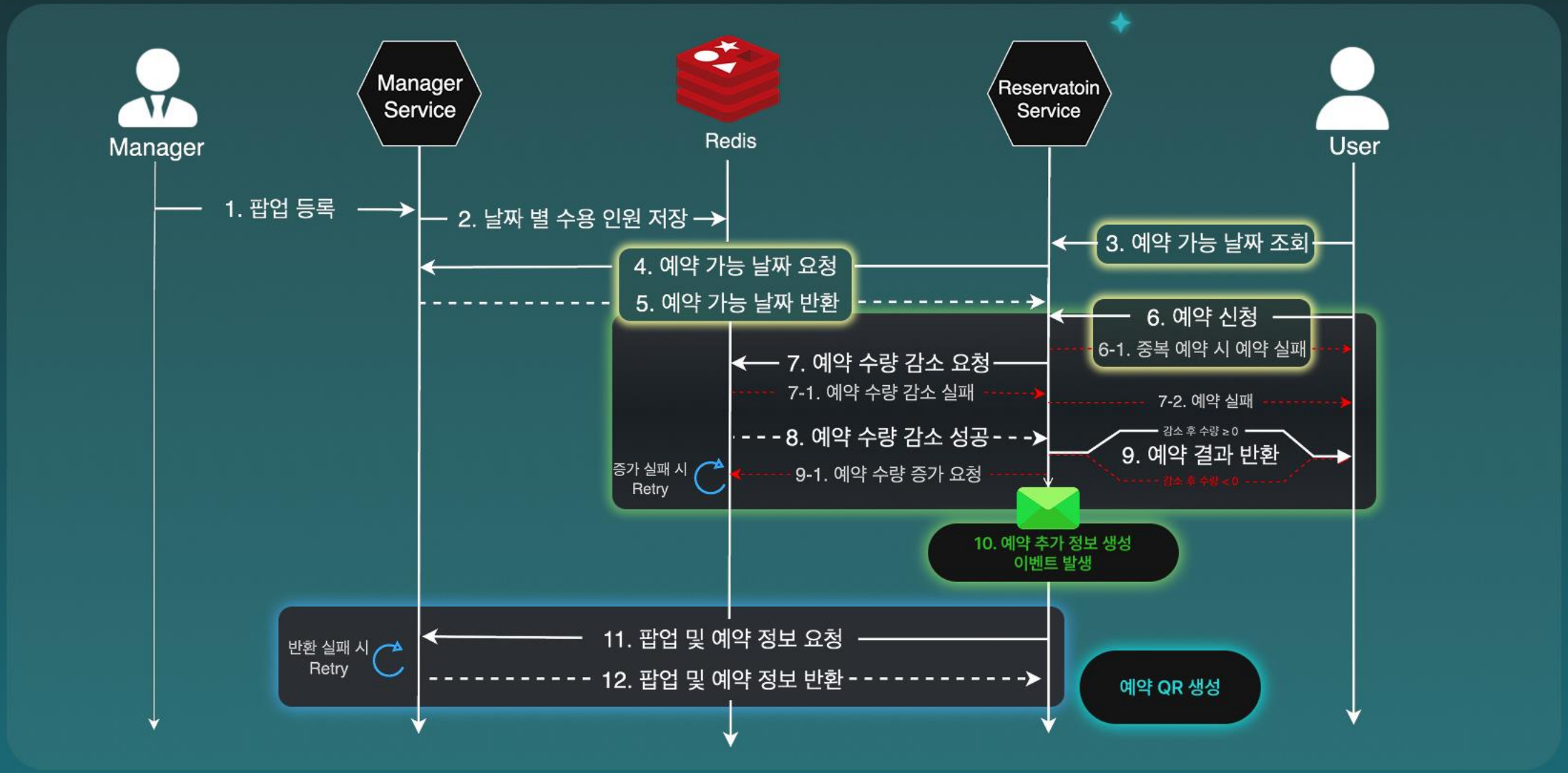
팝업 예약



02 핵심 로직

Reservation

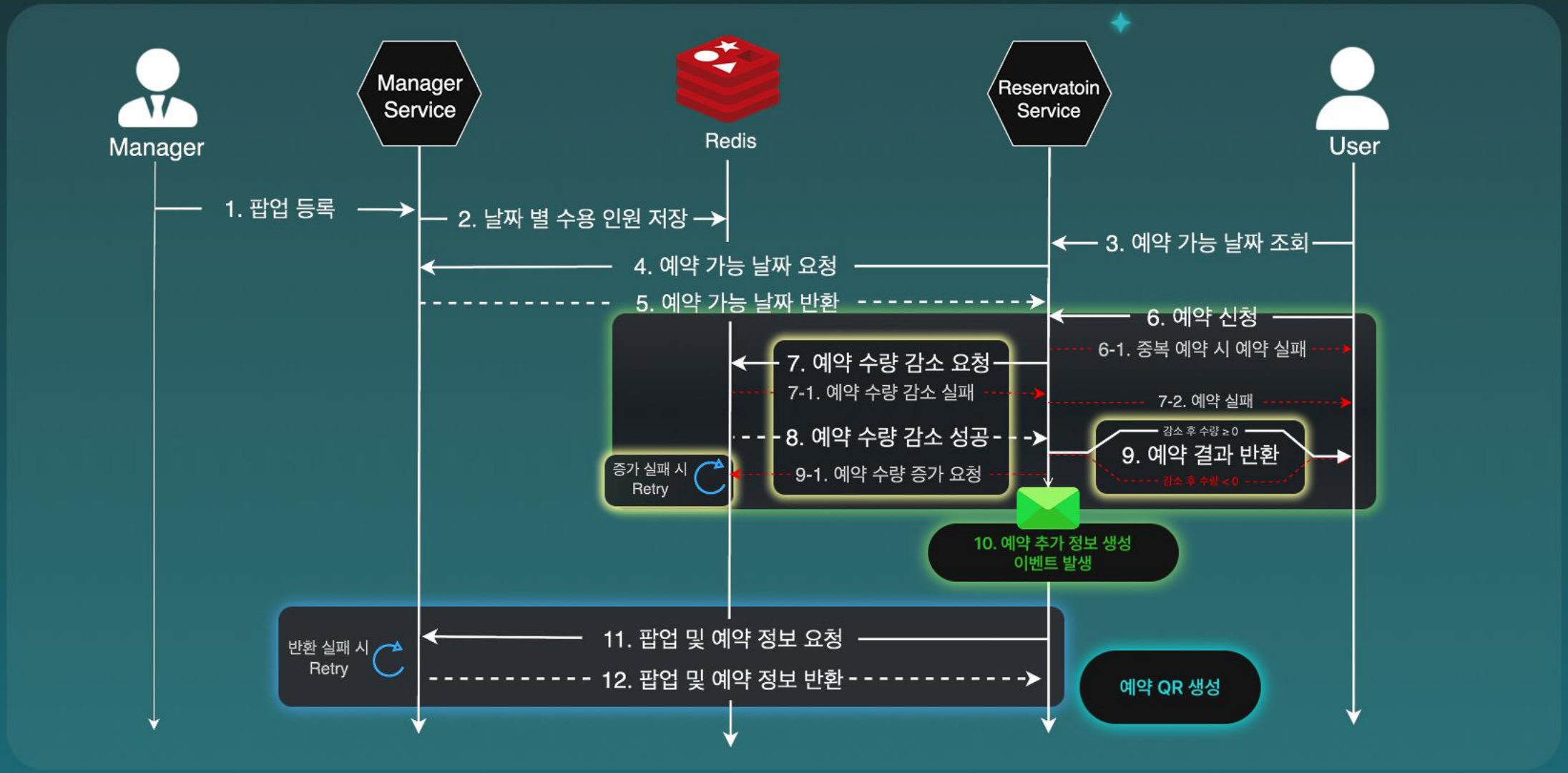
팝업 예약



02 핵심 로직

Reservation

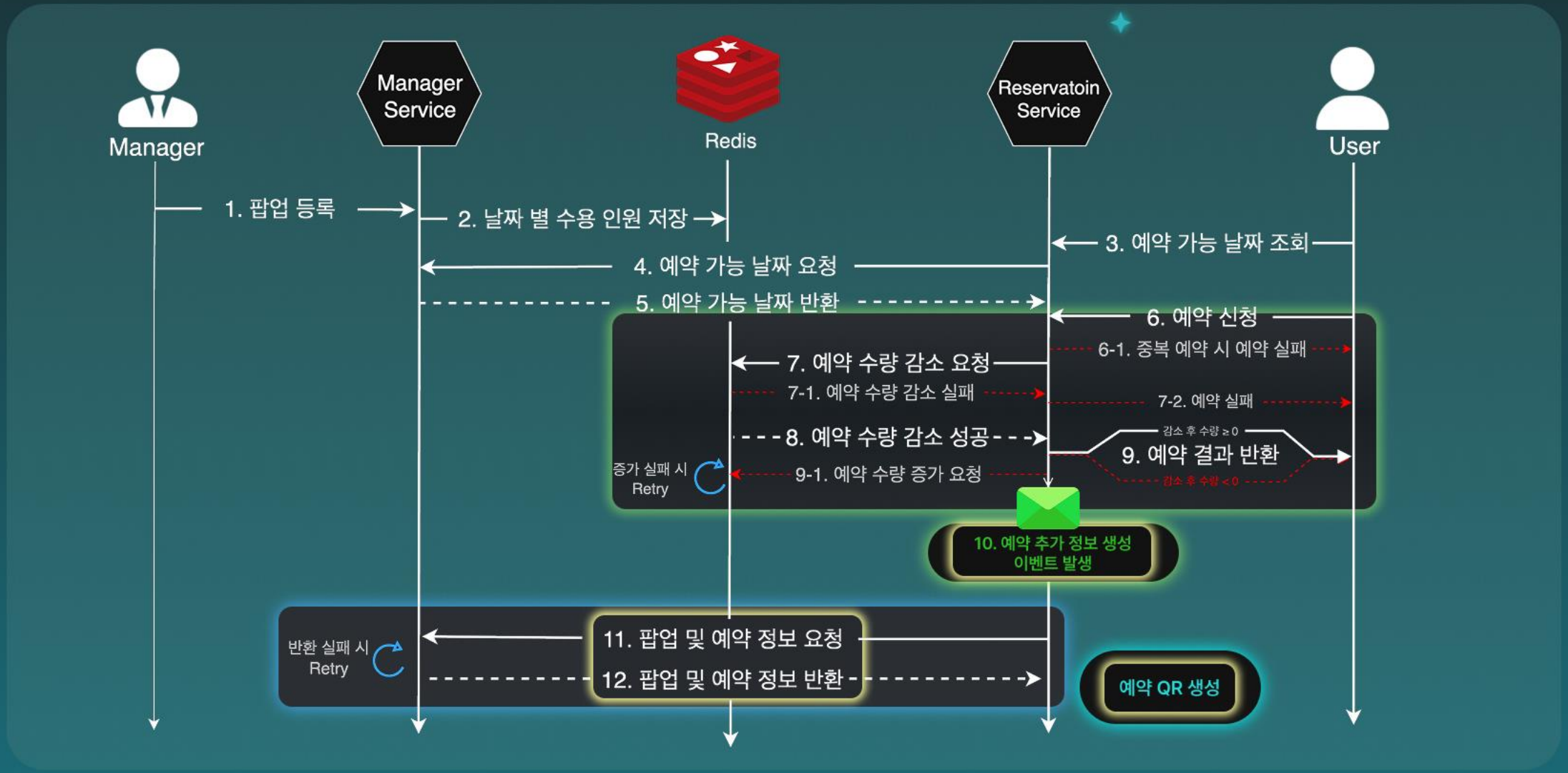
팝업 예약



02 핵심 로직

Reservation

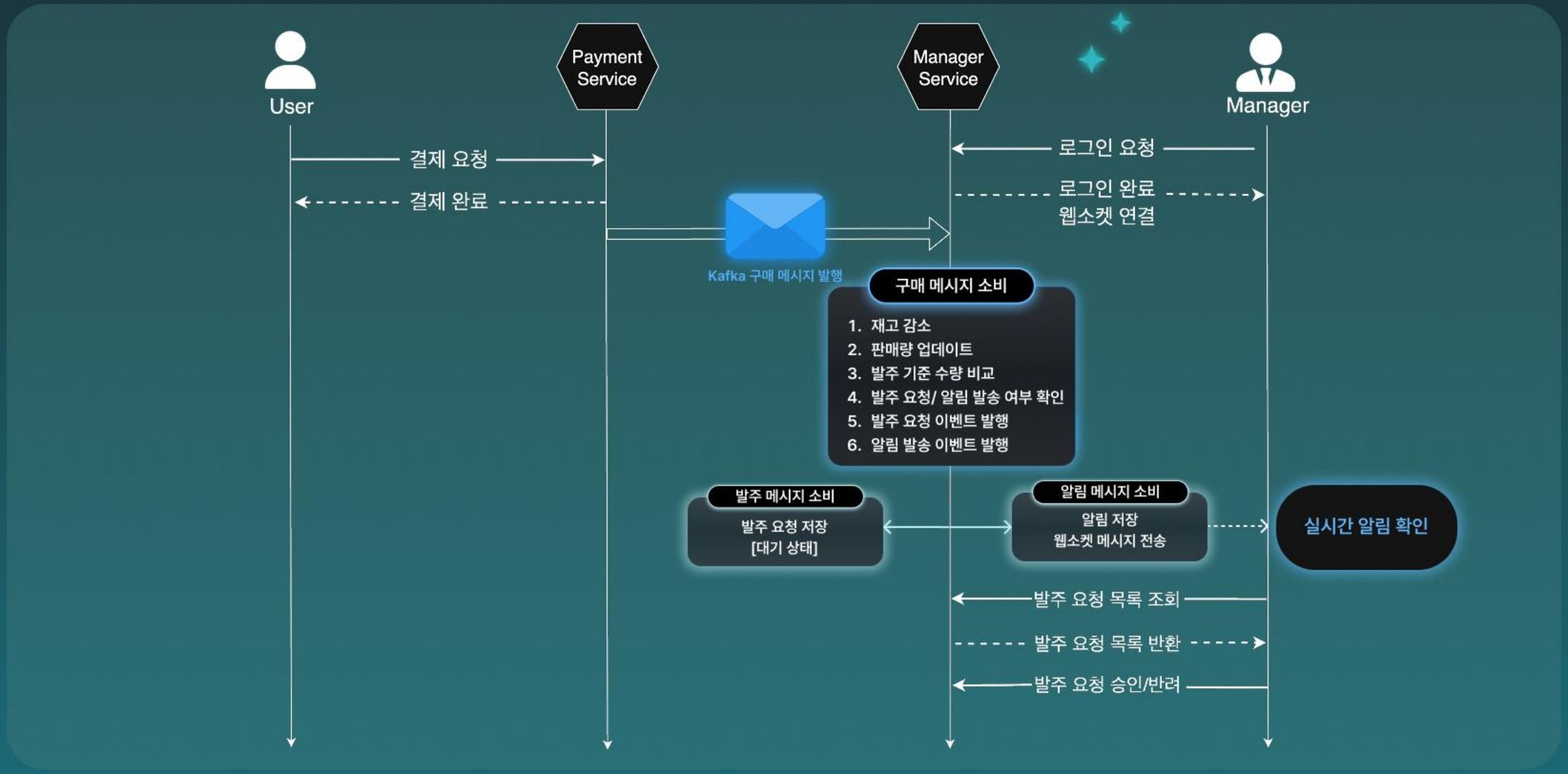
팝업 예약



02 핵심 로직

Payment & Ordering

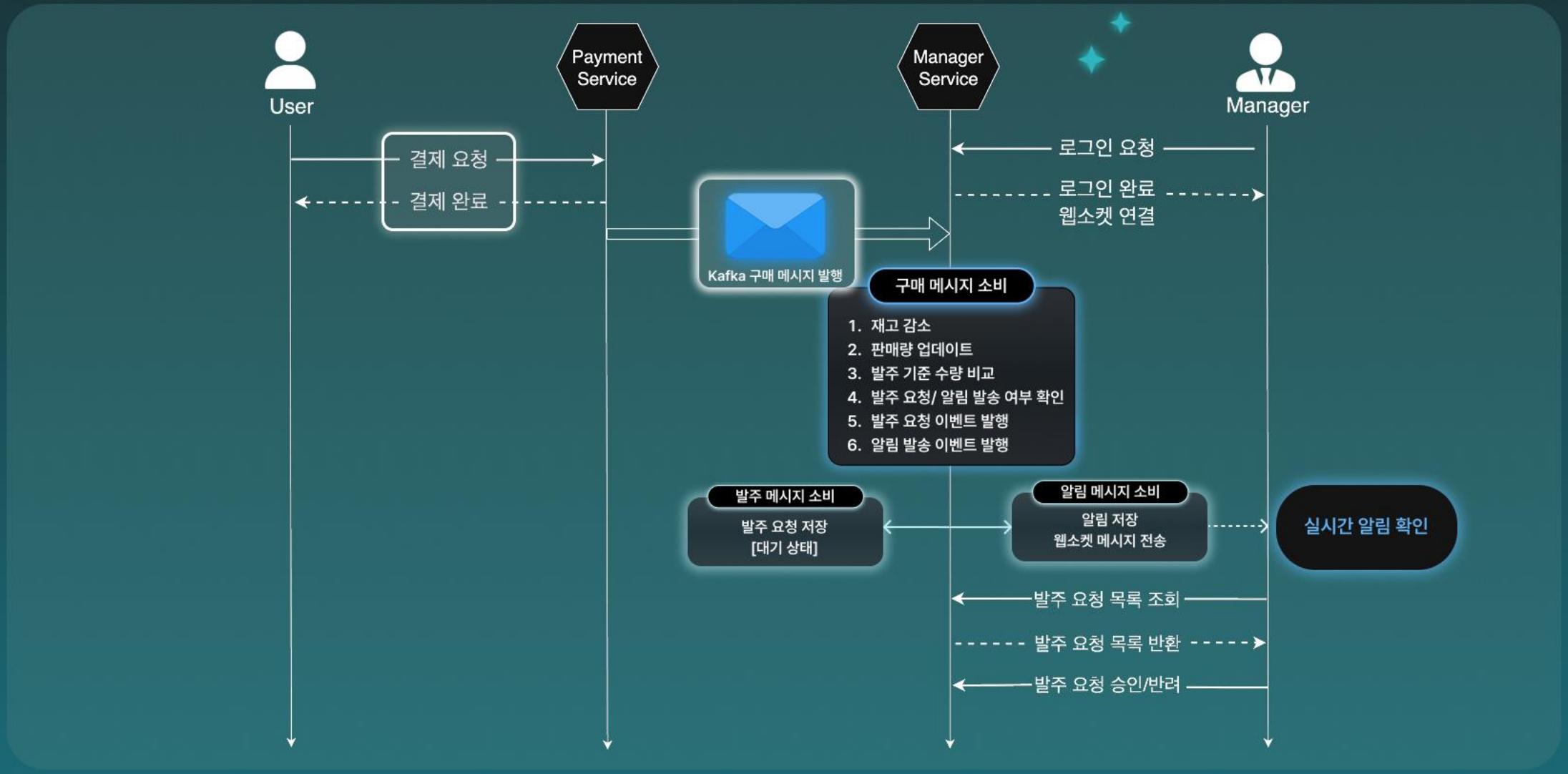
결제 및 발주 요청



02 핵심 로직

Payment & Ordering

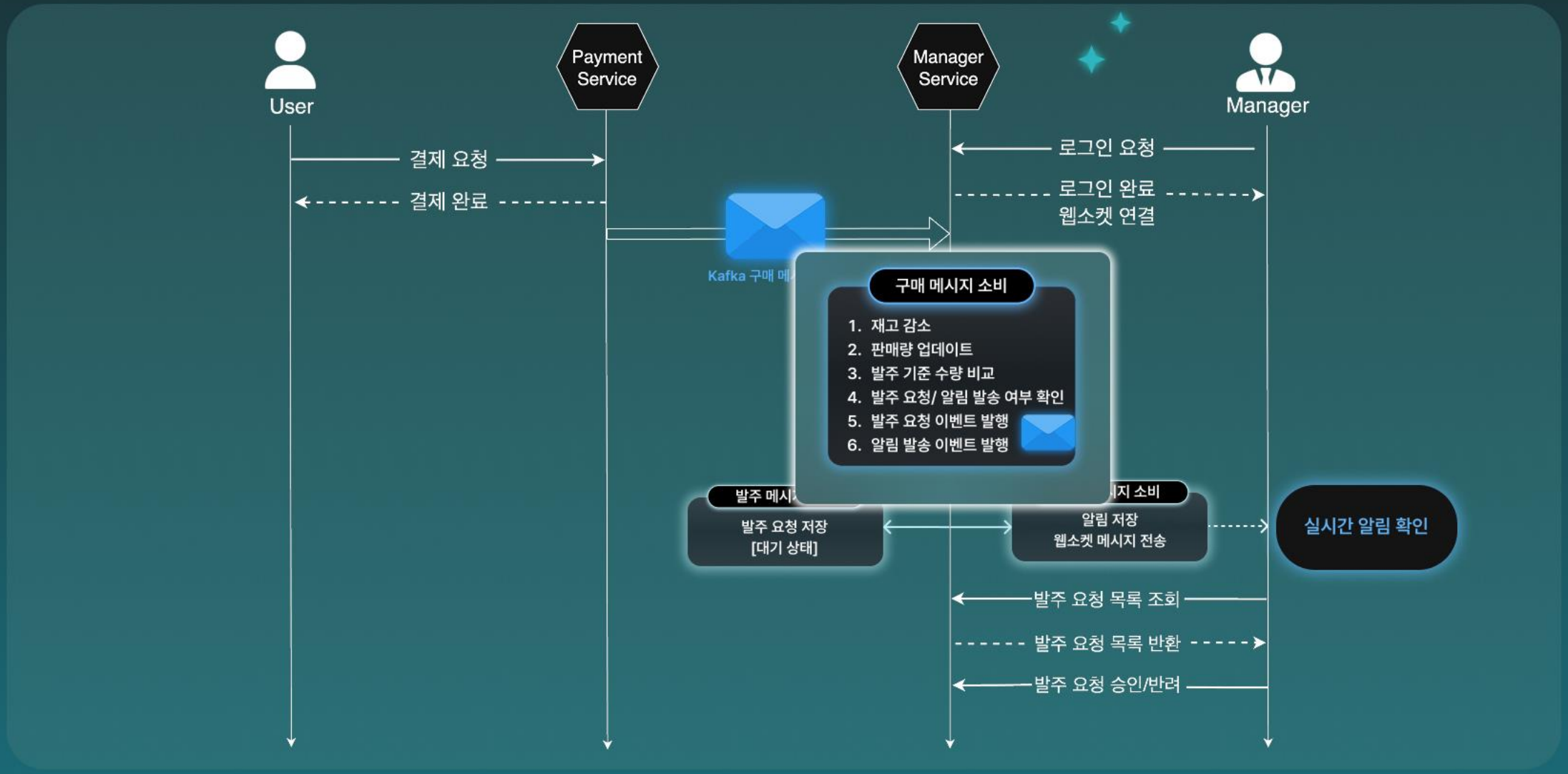
결제 및 발주 요청



02 핵심 로직

Payment & Ordering

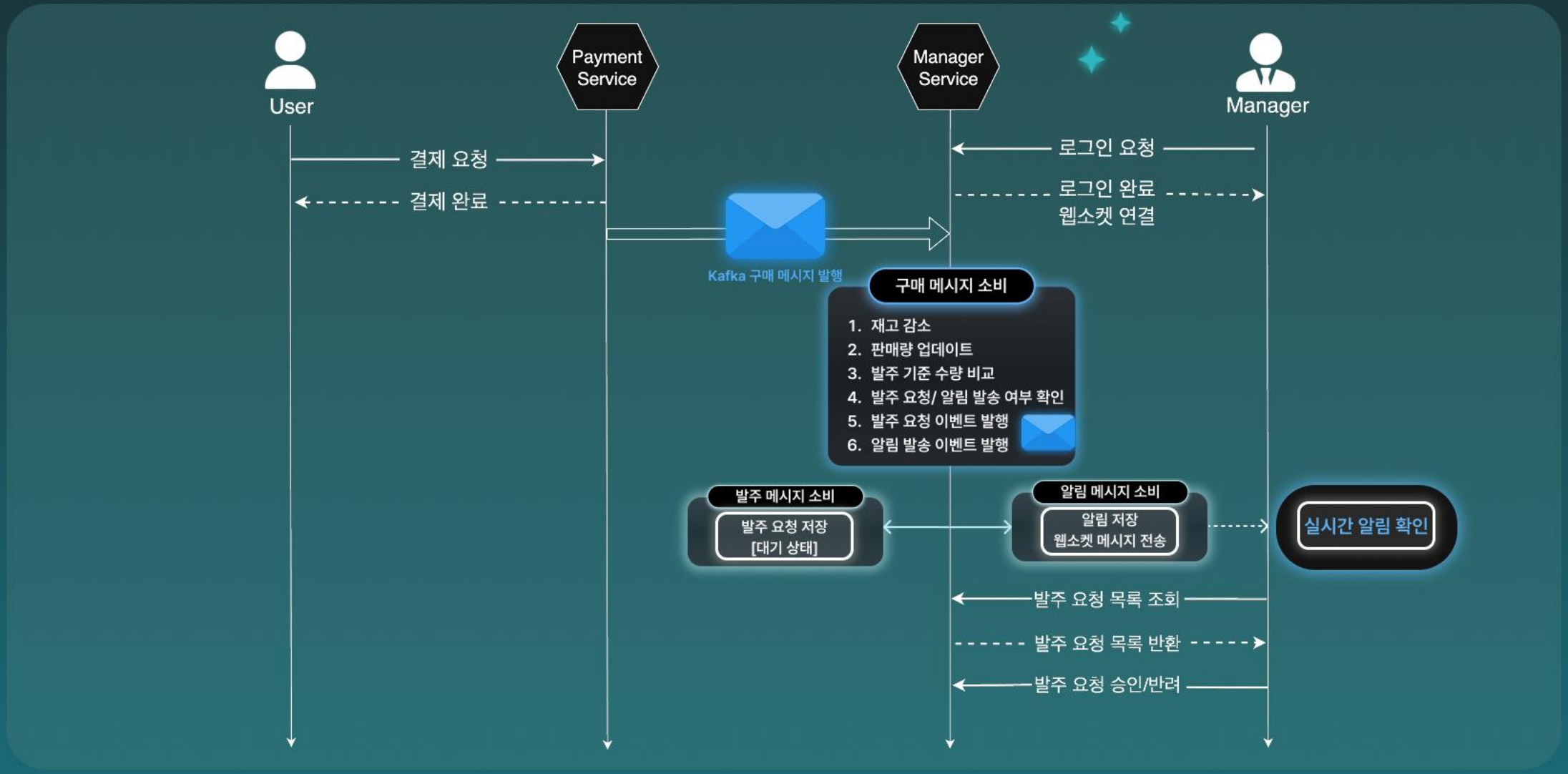
결제 및 발주 요청



02 핵심 로직

Payment & Ordering

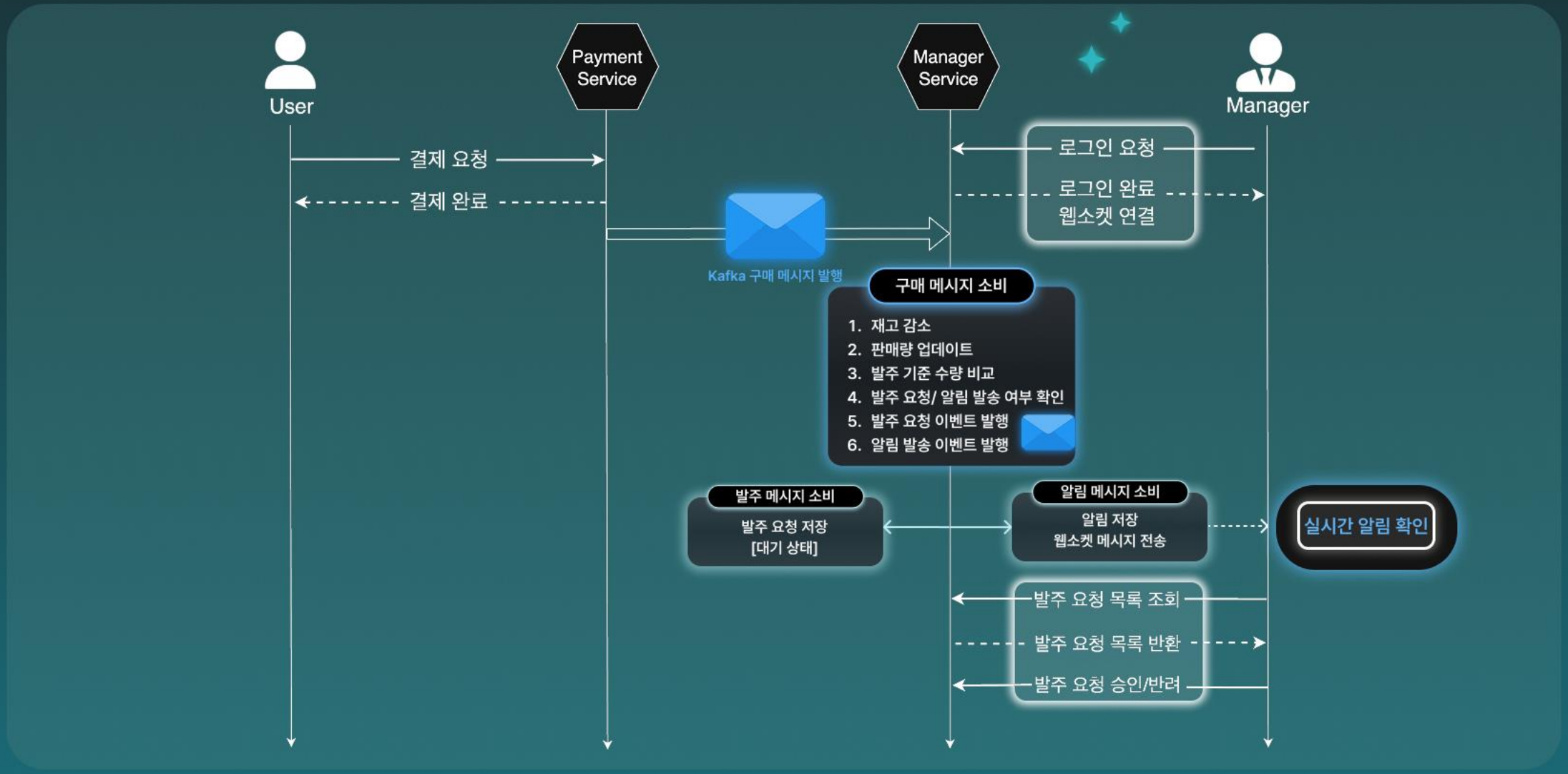
결제 및 발주 요청



02 핵심 로직

Payment & Ordering

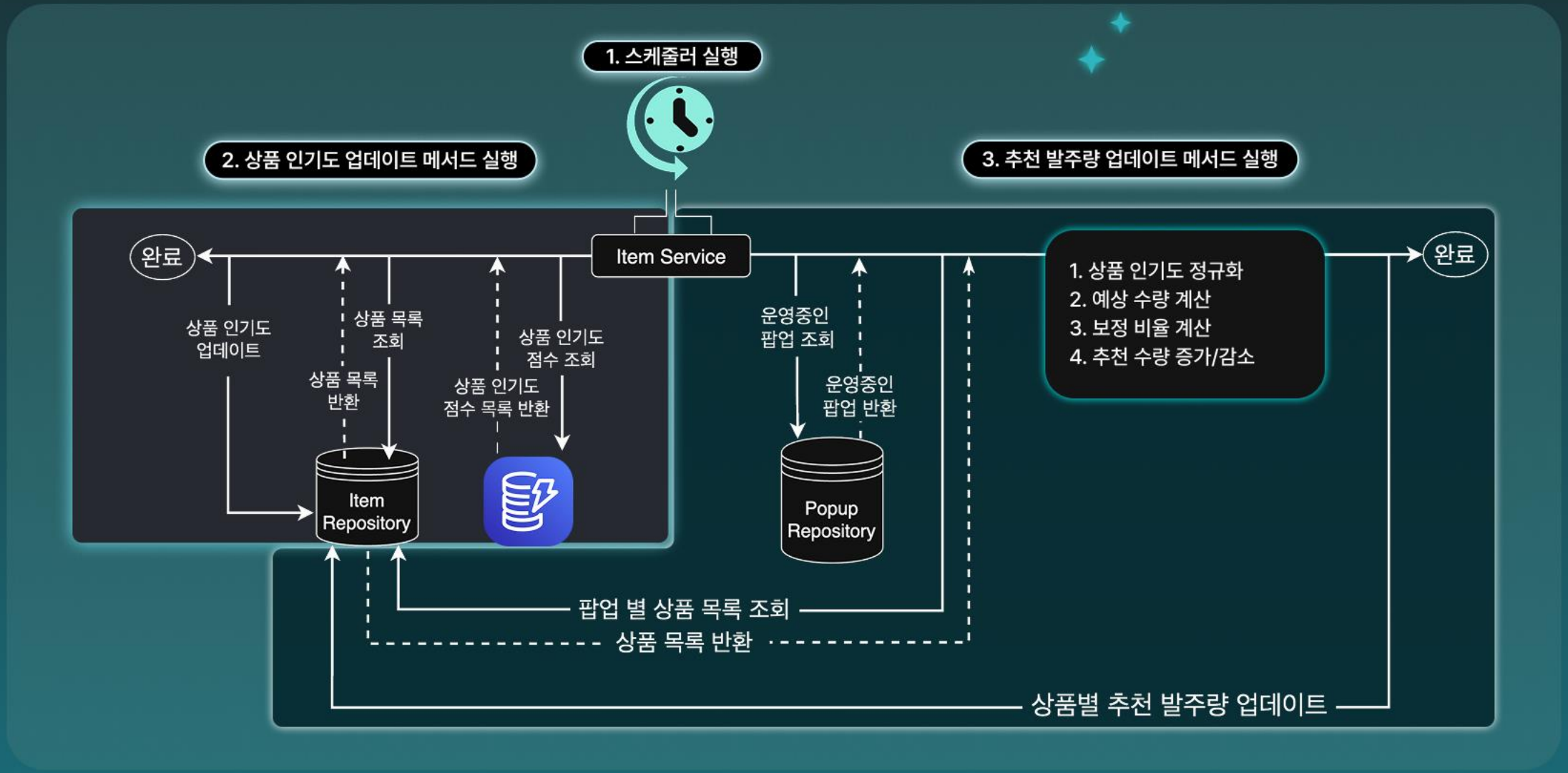
결제 및 발주 요청



02 핵심 로직

Recommendation

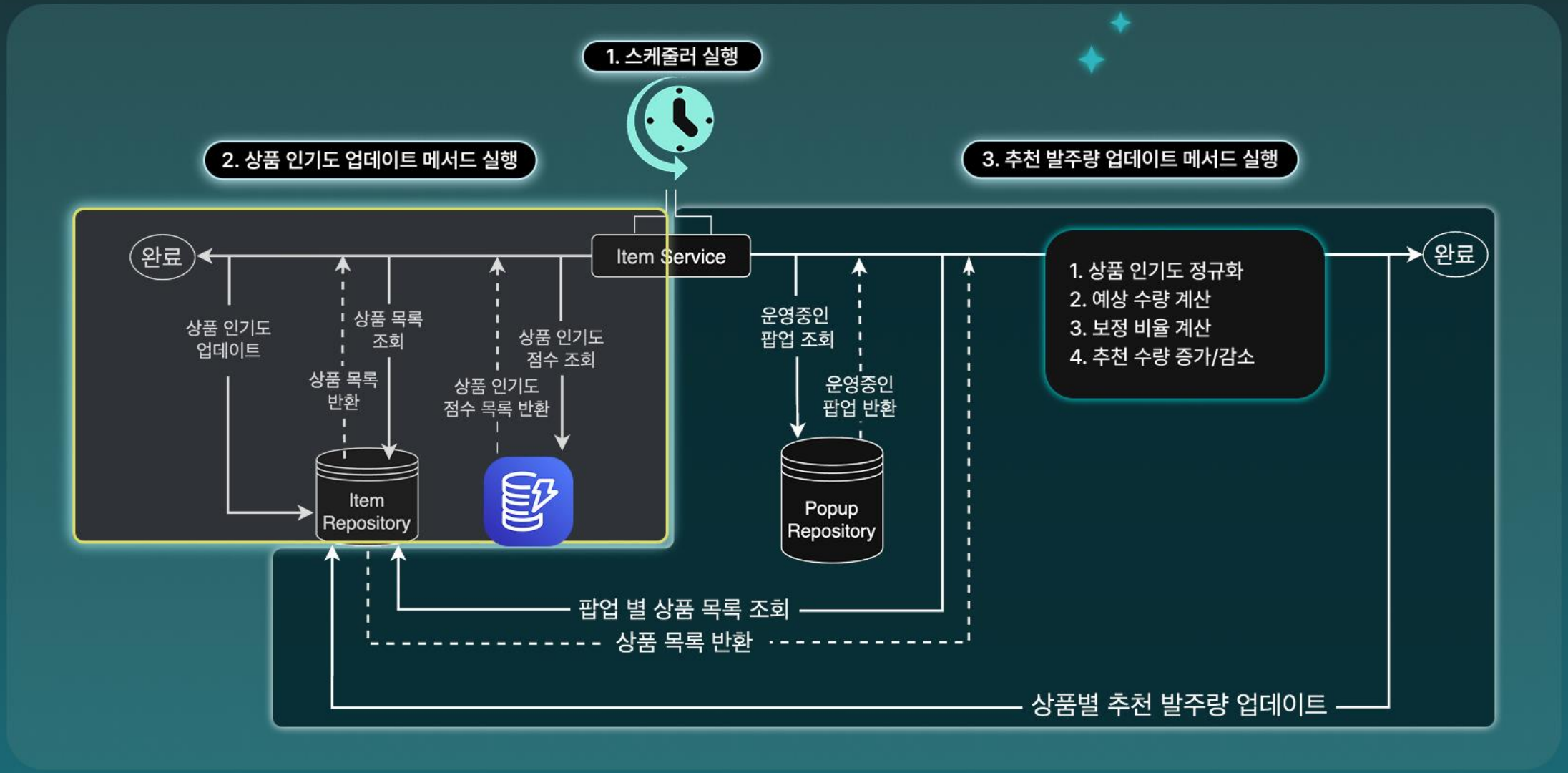
추천 발주량 갱신



02 핵심 로직

Recommendation

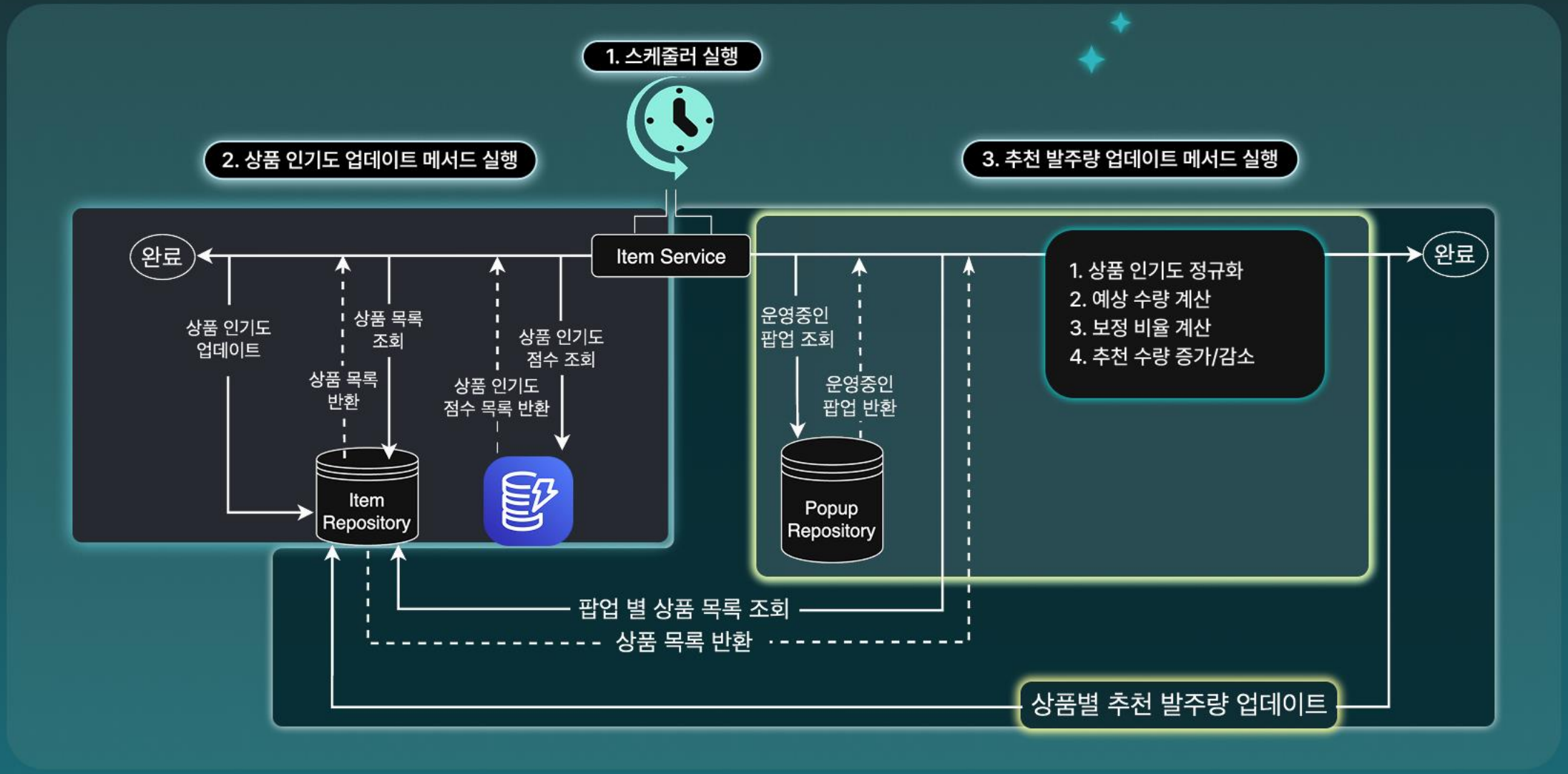
추천 발주량 갱신



02 핵심 로직

Recommendation

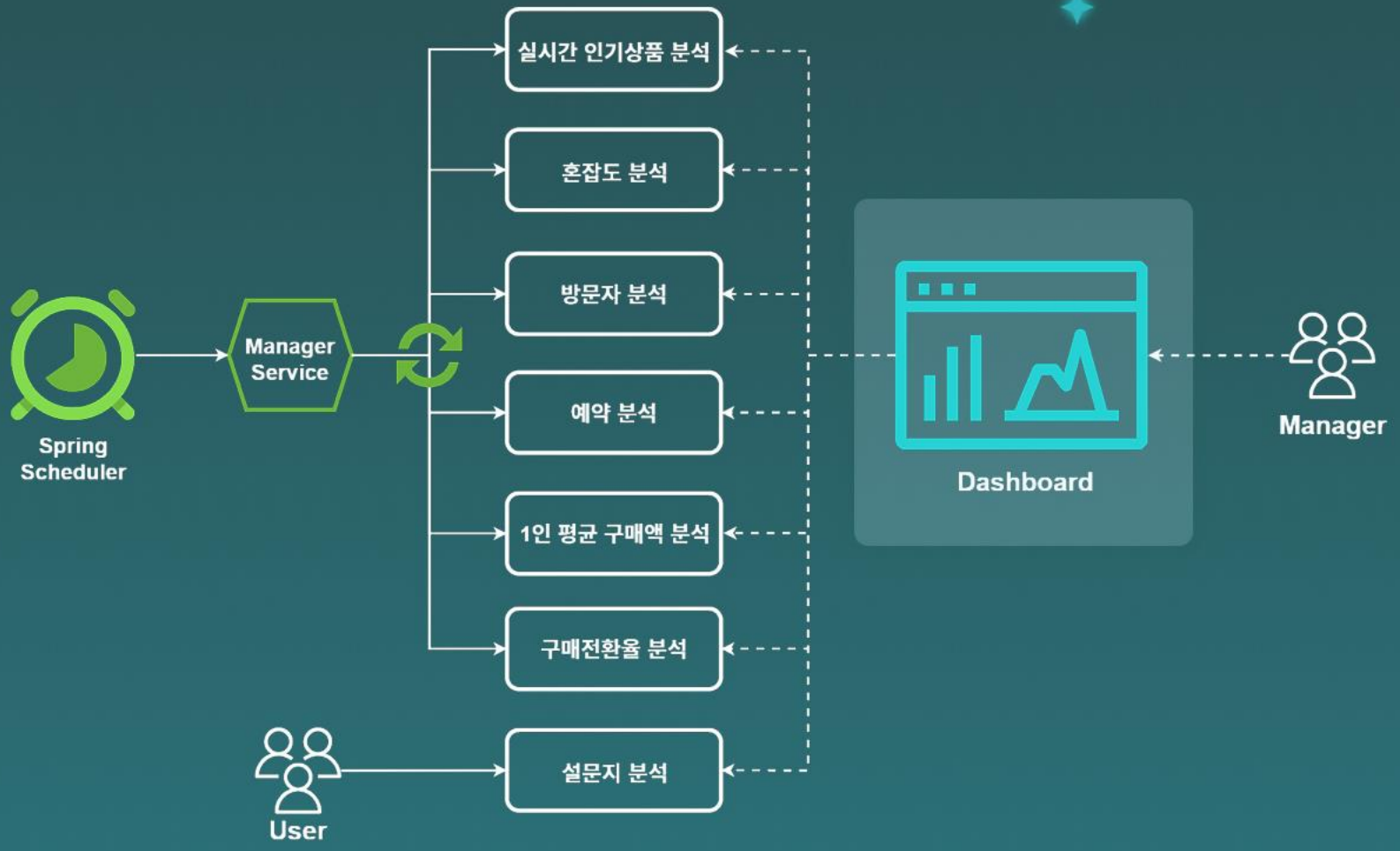
추천 발주량 갱신



02 핵심 로직

Scheduling

대시보드 데이터 갱신



03 성능 개선

Improve Performance

Bulk 연산 & Native Query

대용량 데이터의 INSERT 쿼리 성능 개선을 위해
JPA method 대신 Bulk 연산과 Native Query를 사용했어요

JPA insert reservations took 189 ms

```
2025-06-18T10:20:59.423+09:00  INFO 67135 --- [managers] [nio-8080-exec-2] c.l.d.popup.service.PopupServiceImpl     : JPA insert reservations took 189 ms
2025-06-18T10:20:59.461+09:00 DEBUG 67135 --- [managers] [nio-8080-exec-2] org.hibernate.SQL                        : select r1_0.reservation_id,r1_0.popup_id,r1_0
.possible_count,r1_0.reservation_close_date_time,r1_0.reservation_date,r1_0.reservation_open_date_time,r1_0.reservation_time from reservation r1_0 left join popup
p1_0 on p1_0.popup_id=r1_0.popup_id where p1_0.popup_id=?
Hibernate: select r1_0.reservation_id,r1_0.popup_id,r1_0.possible_count,r1_0.reservation_close_date_time,r1_0.reservation_date,r1_0.reservation_open_date_time,r1_0
.reservation_time from reservation r1_0 left join popup p1_0 on p1_0.popup_id=r1_0.popup_id where p1_0.popup_id=?
2025-06-18T10:20:59.520+09:00  INFO 67135 --- [managers] [nio-8080-exec-2] c.l.g.aop.aspect.ServiceLoggingAspect    : [SERVICE] Method: PopupServiceImpl.createPopup,
 Duration: 389ms
2025-06-18T10:20:59.547+09:00  INFO 67135 --- [managers] [nio-8080-exec-2] c.l.g.logging.filter.HttpLoggingFilter   :
{"traceId":"47cd77a0-ec60-4af4-a597-e5ed919e6d3b","responseBody":"{"success":true,"status":201,"data":{"popupId":6},"timestamp":"2025-06-18T10:20:59.543822"}",
"responseStatus":201}
```

Bulk insert reservations took 53 ms

```
2025-06-18T10:19:57.398+09:00  INFO 66939 --- [managers] [nio-8080-exec-2] c.l.d.popup.service.PopupServiceImpl     : Bulk insert reservations took 53 ms
2025-06-18T10:19:57.428+09:00 DEBUG 66939 --- [managers] [nio-8080-exec-2] org.hibernate.SQL                        : select r1_0.reservation_id,r1_0.popup_id,r1_0
.possible_count,r1_0.reservation_close_date_time,r1_0.reservation_date,r1_0.reservation_open_date_time,r1_0.reservation_time from reservation r1_0 left join popup
p1_0 on p1_0.popup_id=r1_0.popup_id where p1_0.popup_id=?
Hibernate: select r1_0.reservation_id,r1_0.popup_id,r1_0.possible_count,r1_0.reservation_close_date_time,r1_0.reservation_date,r1_0.reservation_open_date_time,r1_0
.reservation_time from reservation r1_0 left join popup p1_0 on p1_0.popup_id=r1_0.popup_id where p1_0.popup_id=?
2025-06-18T10:19:57.507+09:00  INFO 66939 --- [managers] [nio-8080-exec-2] c.l.g.aop.aspect.ServiceLoggingAspect    : [SERVICE] Method: PopupServiceImpl.createPopup,
 Duration: 263ms
2025-06-18T10:19:57.524+09:00  INFO 66939 --- [managers] [nio-8080-exec-2] c.l.g.logging.filter.HttpLoggingFilter   :
{"traceId":"07df34fb-7acc-4ad0-8dbd-26ad73ea863f","responseBody":"{"success":true,"status":201,"data":{"popupId":5},"timestamp":"2025-06-18T10:19:57.521389"}",
"responseStatus":201}
```

03 성능 개선

Improve Performance

gRPC vs OpenFeign

OpenFeign

```
TOTAL RESULTS

  checks_total.....................: 7369    724.409872/s
  checks_succeeded.................: 100.00% 7369 out of 7369
  checks_failed....................: 0.00%   0 out of 7369

  ✓ REST status is 200

  HTTP
  http_req_duration..............................................: avg=67.9
p(95)=156.5ms
    { expected_response:true }...................................: avg=67.9
p(95)=156.5ms
  http_req_failed................................................: 0.00%  0
  http_reqs......................................................: 7369   7

  EXECUTION
  iteration_duration.............................................: avg=68.4
p(95)=157.32ms
  iterations.....................................................: 7369   7
  vus............................................................: 50     m
  vus_max........................................................: 50     m
```

gRPC

```
TOTAL RESULTS

  checks_total.....................: 500     47.892578/s
  checks_succeeded.................: 100.00% 500 out of 500
  checks_failed....................: 0.00%   0 out of 500

  ✓ status is OK

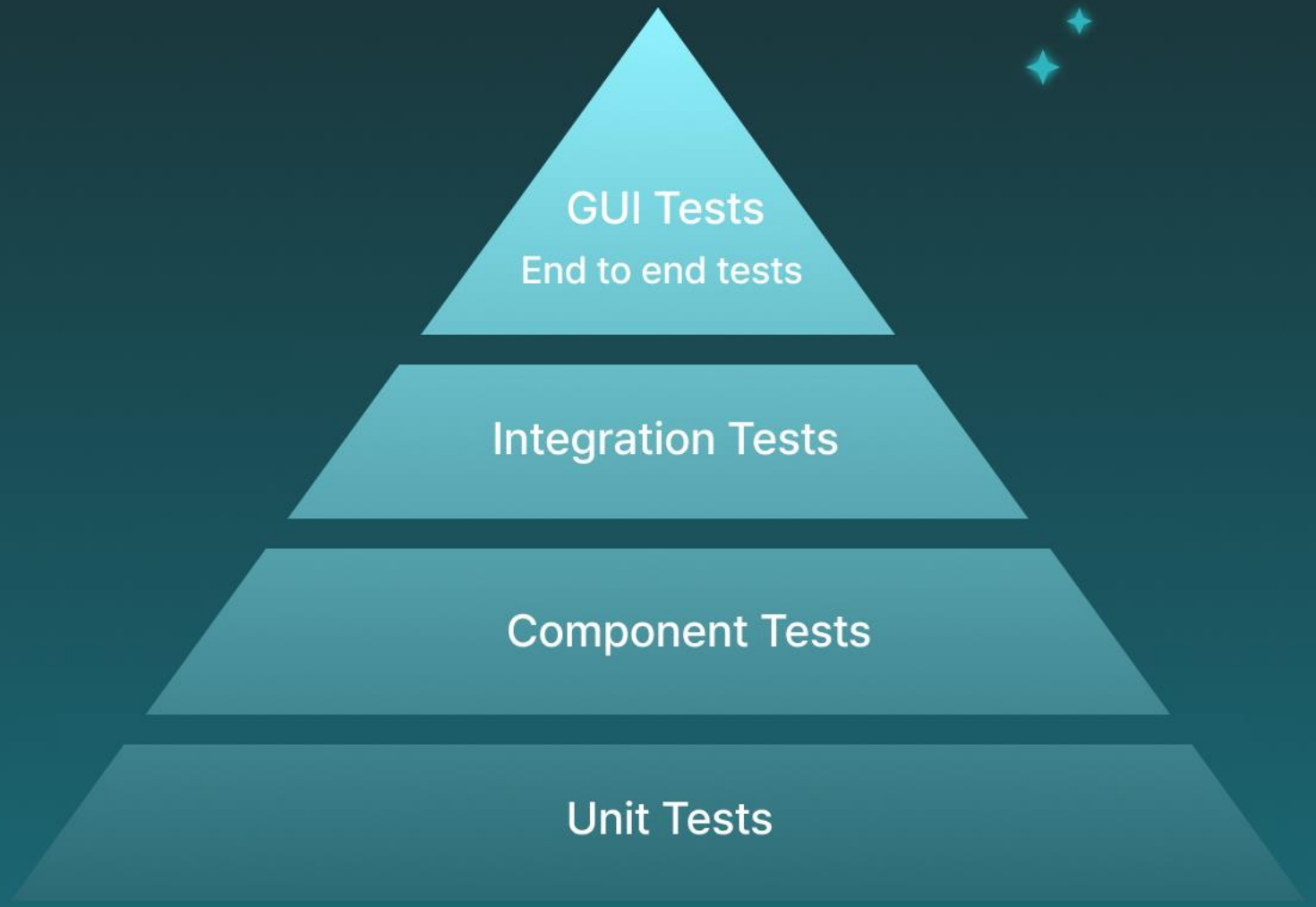
  EXECUTION
  iteration_duration...............: avg=1.04s   min=1s     med=1.03s
  iterations.......................: 500     47.892578/s
  vus..............................: 50      min=50       max=50
  vus_max..........................: 50      min=50       max=50

  NETWORK
  data_received....................: 100 kB 9.5 kB/s
  data_sent........................: 139 kB 13 kB/s

  GRPC
  grpc_req_duration................: avg=30.77ms min=4.96ms med=27.82ms
```

Test Strategy

테스트 전략



Unit & Integration Test

단위 & 통합 테스트

테스트 환경

Unit test	@Mock
Integration test	@SpringBootTest
Batch	@SpringBatchTest
Open Feign	WireMockServer
gRPC	InProcessServer
Kafka	@EmbeddedKafka

테스트 시나리오

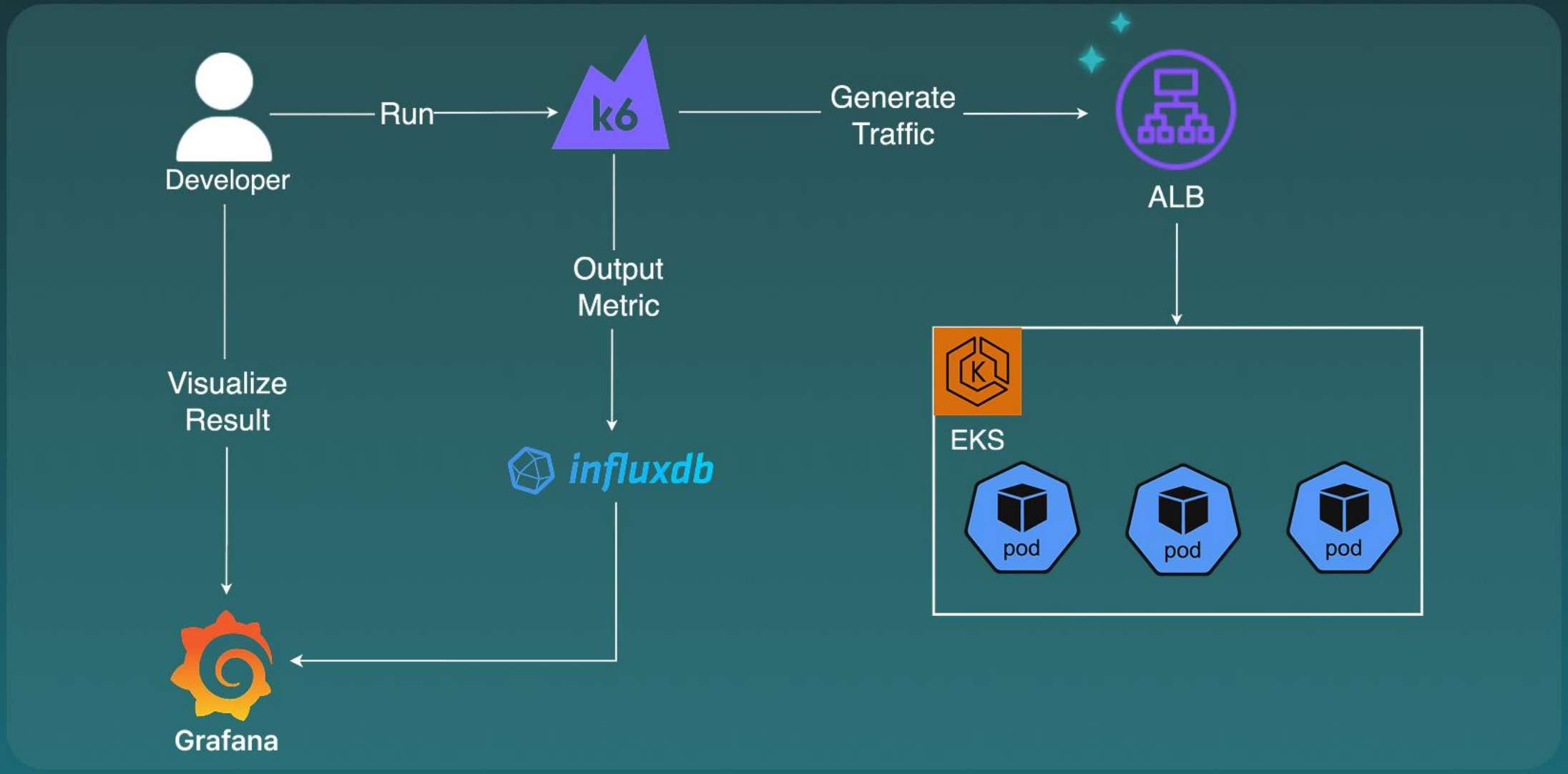
- ✓ 전체 팝업 평균 구매액을 저장할 때
  - ✓ 운영 중인 팝업만 통계 데이터가 저장된다
- ✓ 평균 구매액을 조회할 때
  - ✓ 존재하지 않는 팝업 ID로 조회하면 예외가 발생한다
  - ✓ 데이터가 존재하지 않으면 0을 반환한다
  - ✓ 정상적으로 평균 구매액 조회에 성공한다
  - ✓ 다른 매니저의 팝업에 접근하면 예외가 발생한다

PaymentStatsServiceImpl

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Me
createPaymentStats()		84%		75%	1	3	4	26	0	
findLatestAverageAmount(Long)		100%		n/a	0	1	0	4	0	
validatePopupOwnership(Manager, Popup)		100%		100%	0	2	0	3	0	
findPopupById(Long)		100%		n/a	0	1	0	3	0	
lambda$findPopupById$0()		100%		n/a	0	1	0	1	0	
static {...}		100%		n/a	0	1	0	1	0	
Total	11 of 116	90%	1 of 6	83%	1	9	4	37	0	

Load Test

부하 테스트



Why PLG?

왜 PLG를 사용했나요?

EFK Stack

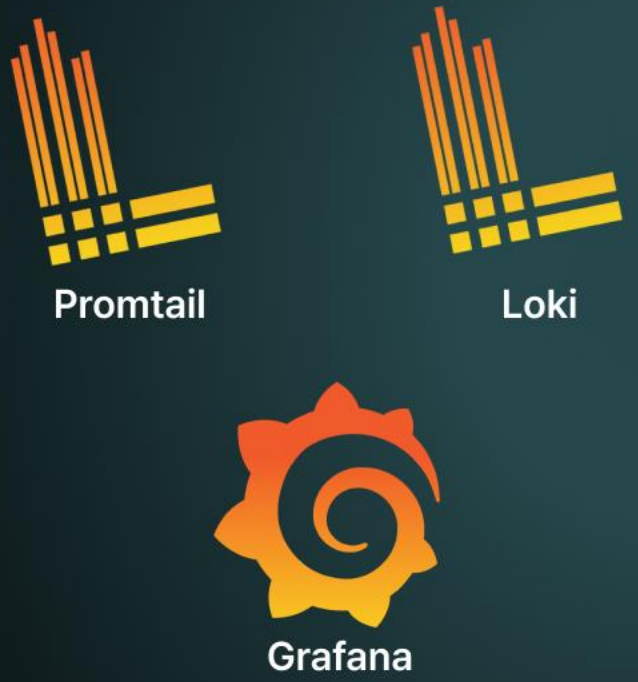
AWS OpenSearch 비용적인 문제

Fluent Bit가 응답 지연 시 Probe 오류 발생

PLG Stack

로그의 메타데이터를 인덱싱하도록 설계되어 비용 효율적

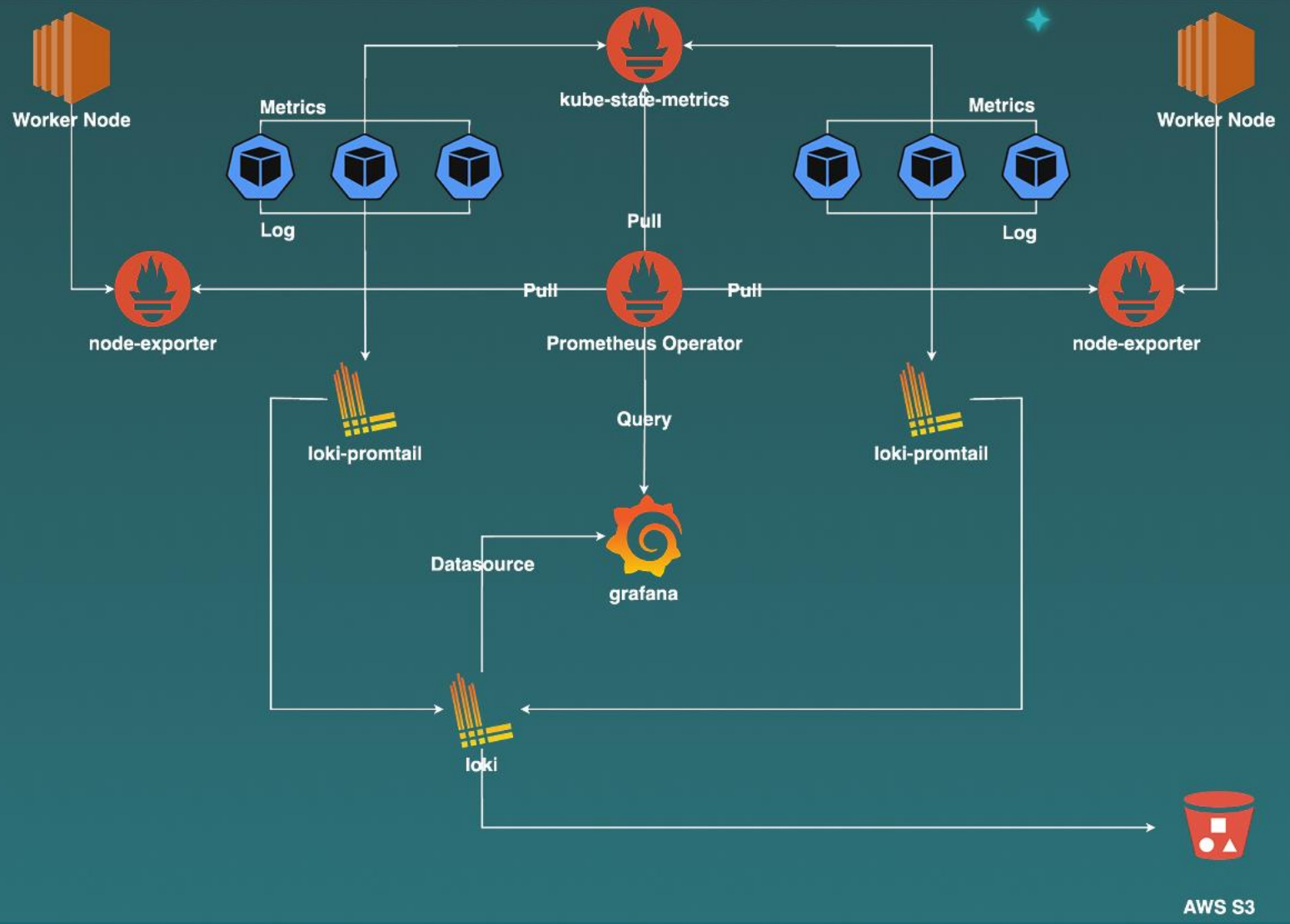
데이터 조회/저장 컴포넌트를 별개로 조정 가능하여 확장에 유리



05 로깅 & 모니터링

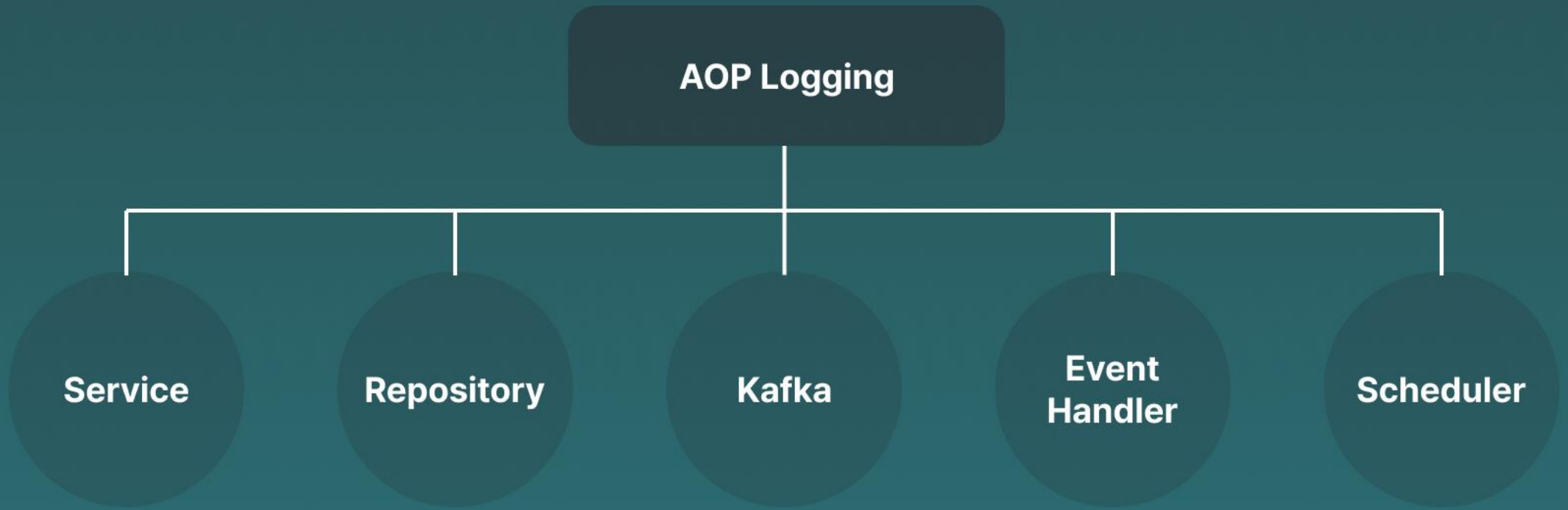
Logging & Monitoring

로그 및 Metrix 수집



Logging & Monitoring

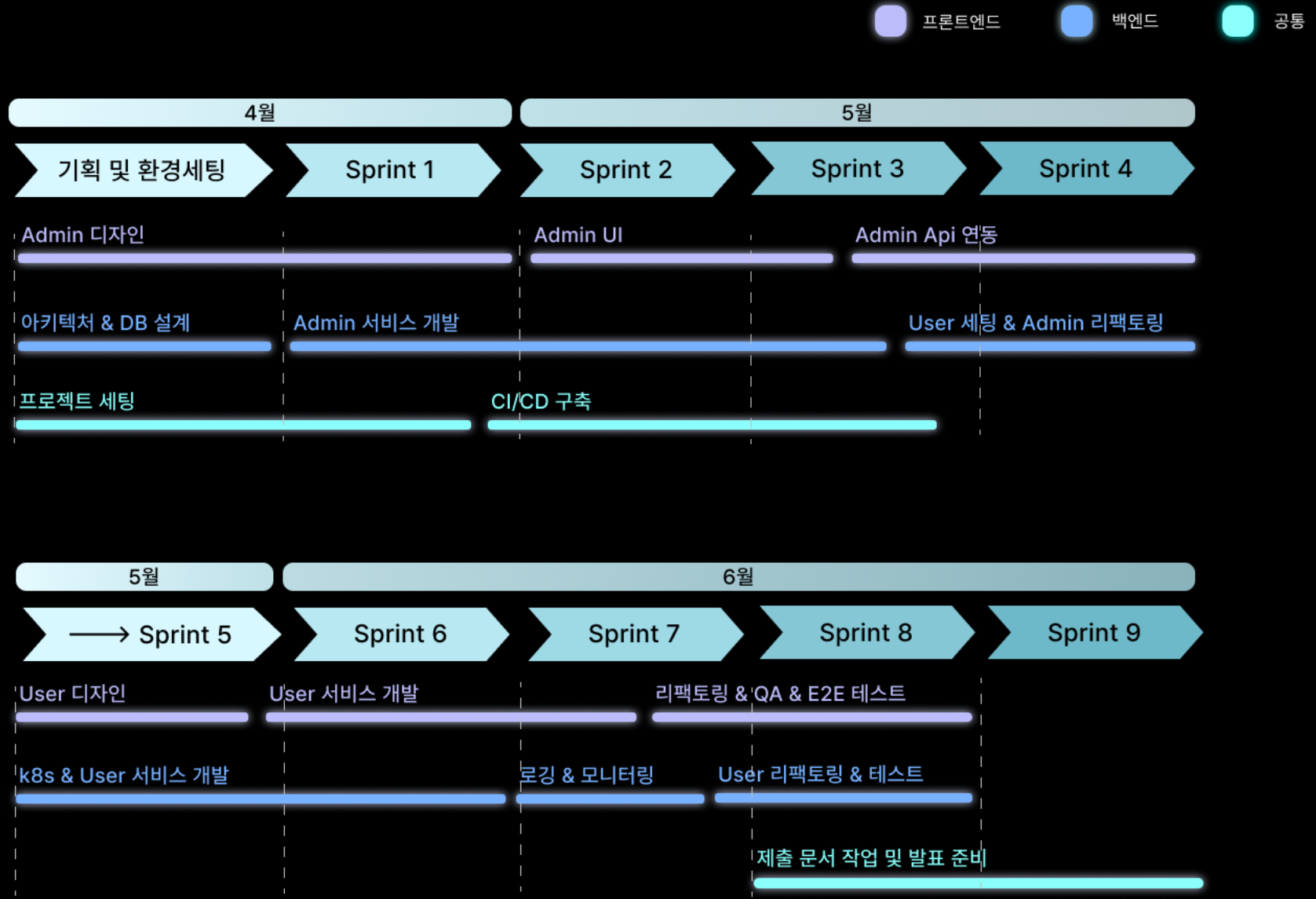
로그 및 Metrix 수집



05

마무리

01 협업 과정


프론트엔드
백엔드
공통
4월
5월
기획 및 환경세팅
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Admin 디자인
Admin UI
Admin Api 연동
아키텍처 & DB 설계
Admin 서비스 개발
User 세팅 & Admin 리팩토링
프로젝트 세팅
CI/CD 구축
5월
6월
Sprint 5
Sprint 6
Sprint 7
Sprint 8
Sprint 9
User 디자인
User 서비스 개발
리팩토링 & QA & E2E 테스트
k8s & User 서비스 개발
로깅 & 모니터링
User 리팩토링 & 테스트
제출 문서 작업 및 발표 준비



인력과 운영비 절감을 위한

무인 스마트 팝업스토어 관리 서비스

PoP!

LG CNS AM INSPIRE 1기