

<http://www.mapzip.shop>

Map.Zip

스케줄 기반 맛집 추천 서비스



INDEX

1. 팀원 역할분담
2. 프로젝트 개요
3. 아키텍처
4. 기술적인 도전
5. 페이지별 기능 소개
6. ERD
7. 테이블 정의서
8. NoSQL
9. API 명세서
10. CI/CD 파이프라인
11. 모니터링
12. K6 테스트
13. 트러블슈팅
14. 결과
15. 회고

팀장

박시윤

- EKS 내부 프로그램 설치, 멀티클러스터, Istio 설정
- 모니터링 및 분산 추적 - Gateway 서버 구축
- K6 테스트
- 협업 툴 연동 및 프론트엔드, 백엔드 CI/CD 구축

부팀장

한동연

- EKS 클러스터 인프라 구축
- Spring Authorization 서버 기반 인증 서버 프론트, 백엔드 구현
- Redis 기반 토큰 관리

팀원

서예은

- Route53,,https,ECR, s2s vpn 인프라 구축
- 추천 서버 구축
- msk, bedrock 연동
- 프론트엔드 초안 작성

팀원

양정모

- Aurora DB RDS 클러스터 인프라 구축
- 정적 웹사이트 호스팅 S3 리소스 구성
- 스케줄 서버 CRUD 작성 및 Tmap 로직 기여

팀원

조성욱

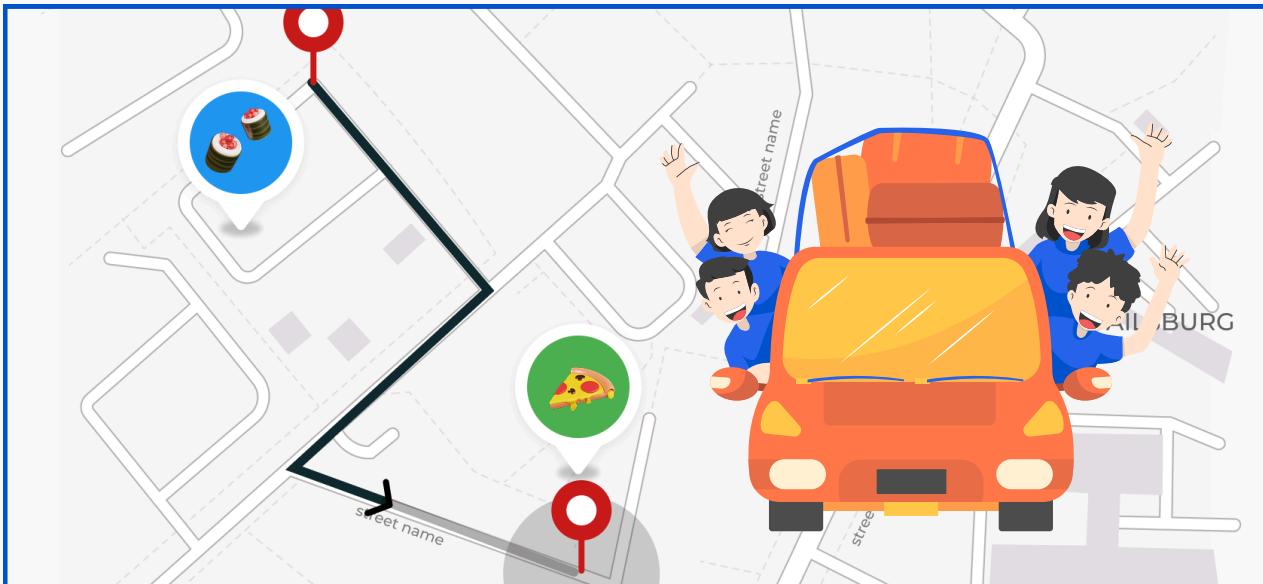
- EKS 내부 프로그램 설치, 멀티클러스터, Istio 설정- 네트워크(VPC, 서브넷, IGW) 인프라 구축
- CloudWatch 알람, SNS 토픽, Lambda 함수 기반 실시간 Slack 알람 시스템 개발
- Config Server 구축, Config CI/CD 구축

팀원

조성민

- msk, DynamoDB, elasticache 인프라 구축
- review 서버 백엔드 ,프론트 엔드 , 모니터링 구축
- google VISION api 연동
- k6 테스트

개요



이동 경로 기반 AI 맛집 추천

장거리 이동시 이동 스케줄과 사용자 정의 요구사항, 식사시간, 목표 도착 시간, 식사 반경 등을 입력하면 원하는 식사시간에 사용자가 위치하게 될 장소를 예측하여 해당 지역의 맛집을 추천해주는 서비스

기획의도



계획된 여행, 사용자에게 최적화된 식사 추천

단순 지역 맛집을 추천하는것이 아니라 사용자 입력 데이터와 AI, 경로탐색 API를 통해 장거리 이동의 모든 과정을 예측하고 최적화하여 사용자에게 계획된 즐거움을 제공하는 플랫폼

스케줄 기반 식사 예정 지점 도출

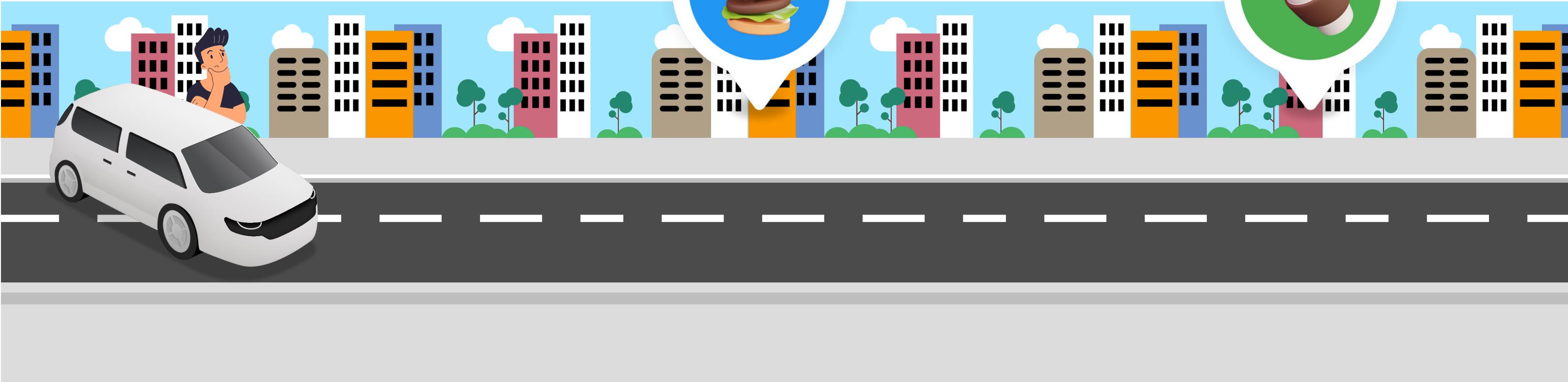
- 출발지 ~ 도착지 경로 중 식사 시간 기준 위치 계산
- 사용자가 설정한 식사반경과 도출된 좌표를 기반으로 반경 내 음식점 리스트 확보

AI 사용자 맞춤 반경 내 맛집 추천

- 사용자가 입력한 요구사항(식사/간식, 동승자 정보, 이동목적)과 리뷰 데이터를 기반
- AWS bedrock(claude-3-sonnet)에서 식당 3개씩 추천

영수증 OCR 리뷰 작성

- 영수증 OCR 검증을 통해 해당 식당 방문자만 리뷰 작성 가능



Backend

-  Spring Boot
-  PostgreSQL
-  Valkey

AWS

-  CloudFront
-  ElastiCache
-  DynamoDB
-  Lambda
-  Client VPN
-  MSK
-  VPC
-  EKS
-  S3
-  S2S VPN
-  AuroraDB
-  Parameter Store
-  Bedrock
-  SNS
-  ECR

Frontend

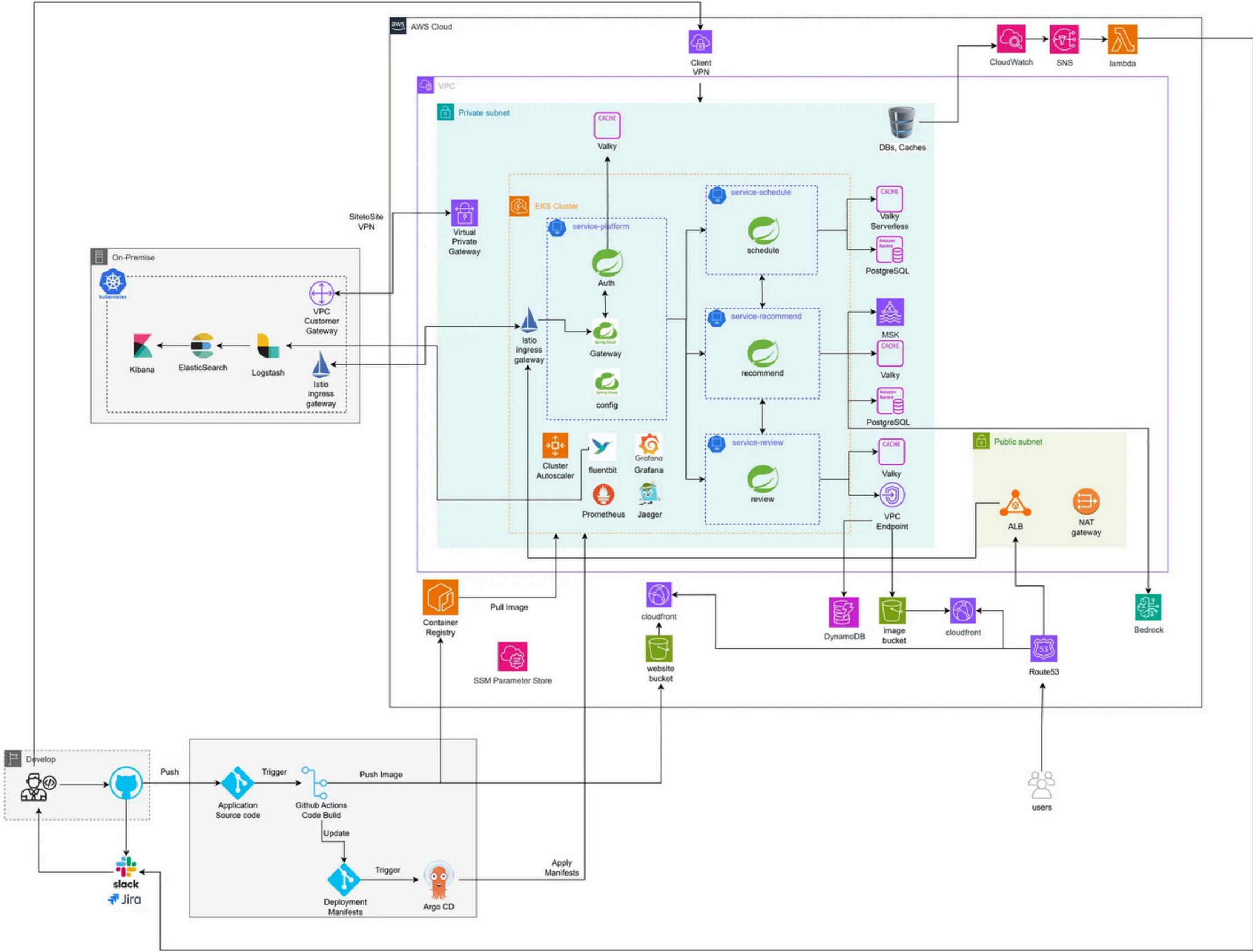
-  React
-  TypeScript
-  Next.js

IaC, CI/CD

-  Terraform
-  Github Actions
-  ArgoCD

협업

-  GitHub
-  Slack
-  Drow.io
-  Jira
-  Notion
-  Figma
-  Terraform Cloud



Frontend & Backend

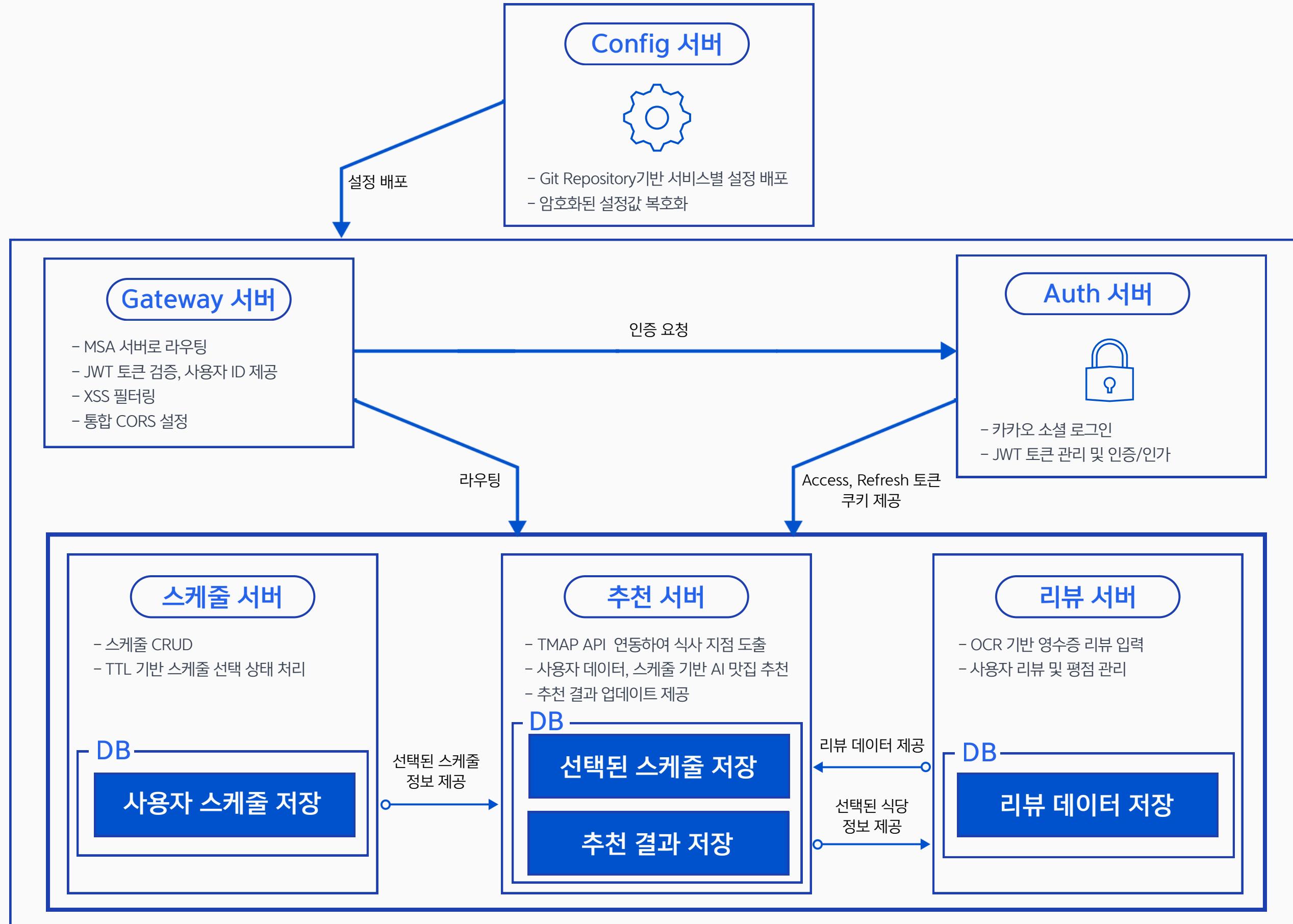
- Frontend: S3 + CloudFront 호스팅
- Backend: EKS 기반 MSA 아키텍처

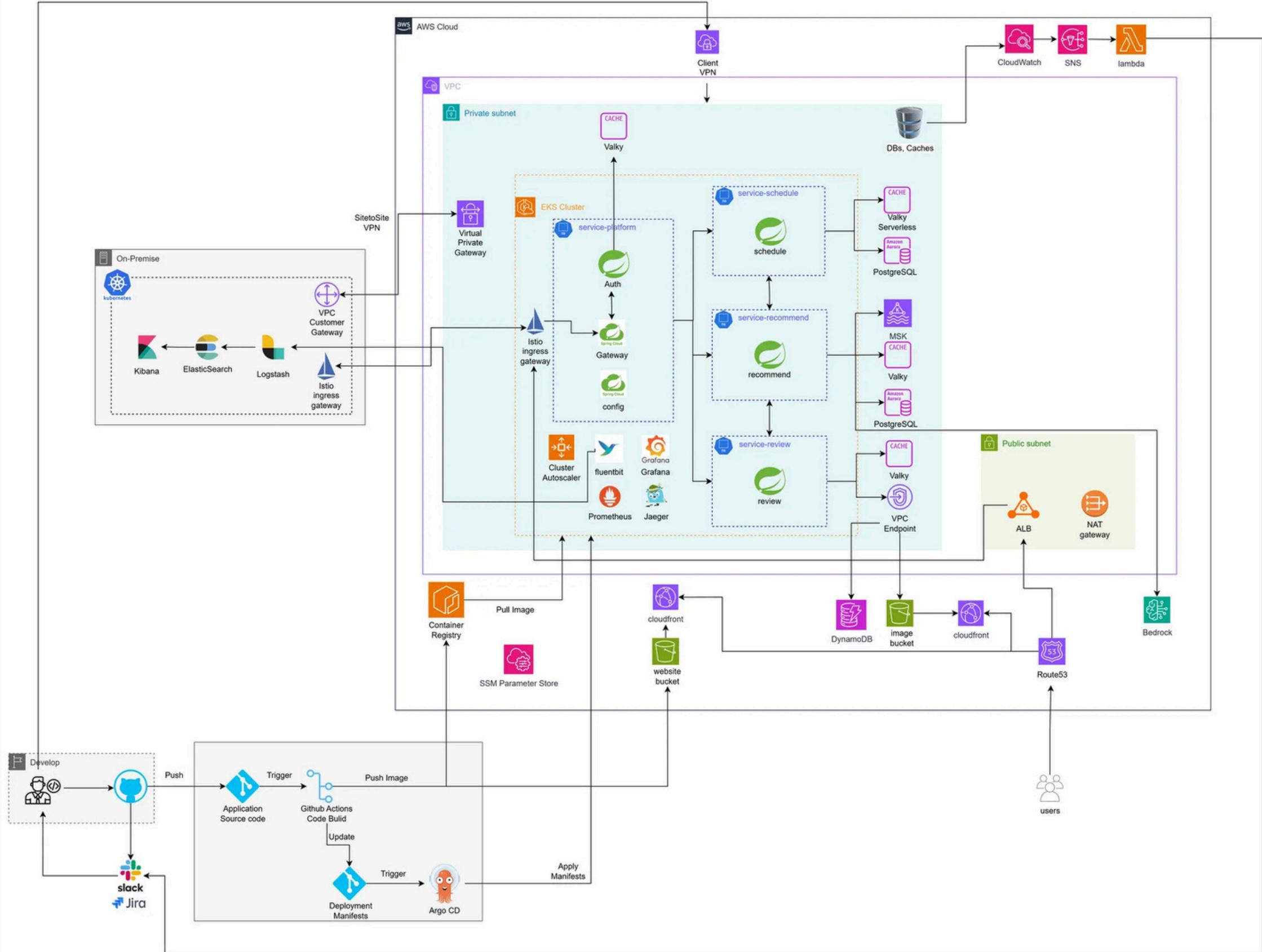
안정적인 인프라

- Cluster Autoscaler, HPA로 리소스 스케일링
- Rolling Update, 헬스체크 설정으로 무중단 배포
- Multi-AZ Private/Public 서브넷 구성으로 고가용성 보장

MSA Architecture

- Spring Cloud Gateway, Config 서버로 중앙화된 관리
- 6개 서비스 분리로 독립성과 확장성 확보
- Istio 서비스메시로 네트워킹, 분산추적, 필터링 관리
- gRPC를 통한 고성능 내부 서비스 통신





Monitoring

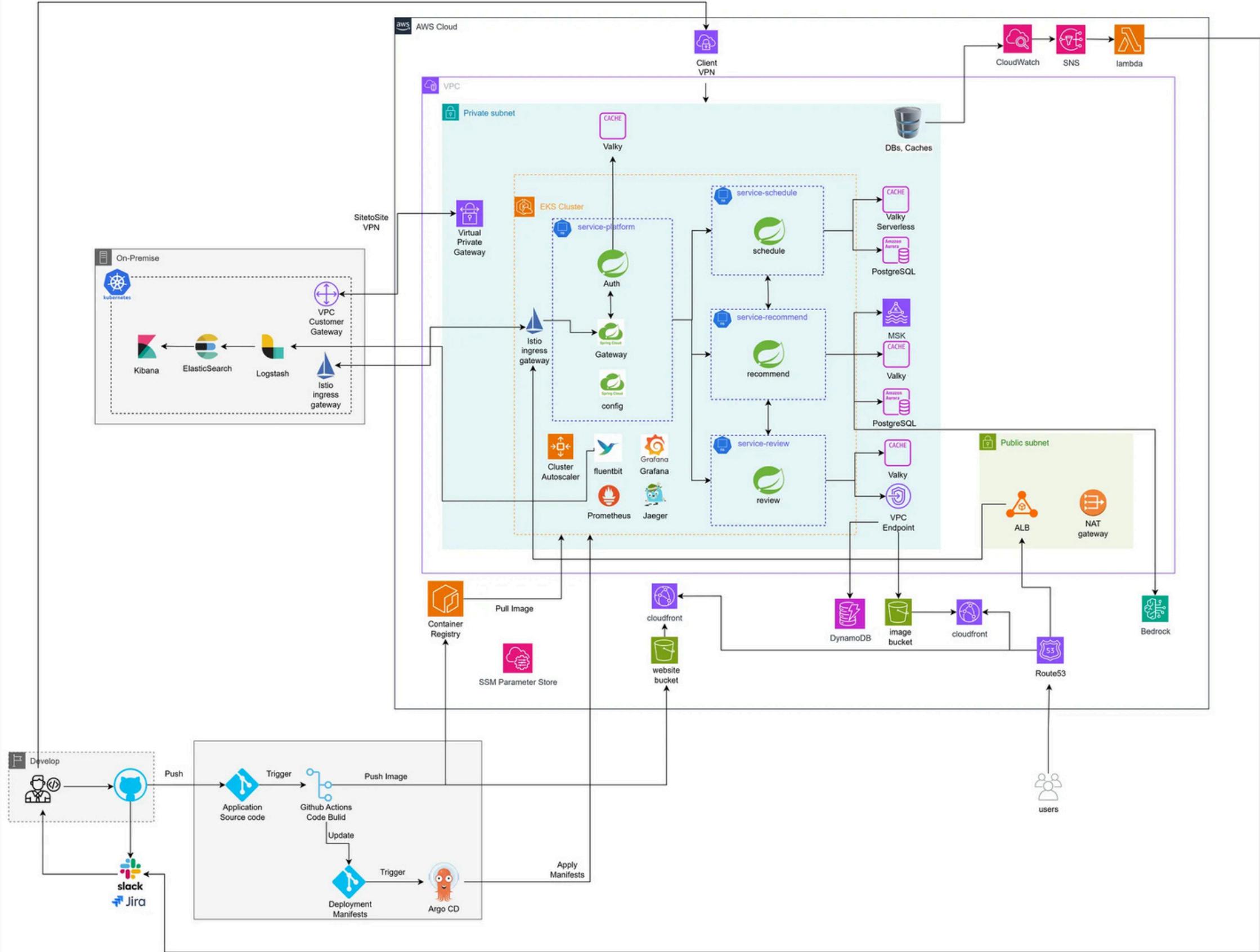
- Prometheus + Grafana 대시보드
- DB별 슬랙 알람, EFK 로그 모니터링

Database

- Valky : TTL 기능이 필요하거나 빠른 조회를 위해 사용. 저장 할 데이터 양이 적고 자주 사용되지 않는 서비스에서는 Valky Serverless 사용
- PostgreSQL : 추후 AI 서비스와 벡터 DB로 RAG시스템 구축 고려하여 선택
- DynamoDB : NoSQL로 높은 확장성
- MSK (Kafka) : 오래 걸리는 API 비동기 처리를 위해 사용
- 장애 대응 : DB별 자동 백업 , Point-in-Time 복구 설정, 읽기 복제본으로 부하 분산

Hybrid Cloud

- 온프레미스 K8S 로깅 서버 구축
- S2S VPN 연결, Istio 멀티클러스터 설정



보안

- 백엔드, 프론트엔드에 HTTPS 적용
- Client VPN 사용하여 내부 리소스 접근
- SSM Parameter Store로 스크립트 민감정보 관리
- Spring 설정 서버 암호화 기능 사용
- Terraform cloud sensitive 변수 설정, Github Actions Secret 변수 설정하여 인프라, 리소스 배포에 사용
- Sealed Secret으로 EKS Secret GitOps 관리
- AWS 리소스별 최소 권한 부여, Github Actions용 OIDC 설정
- 백엔드 gateway에서 XSS 필터링 적용

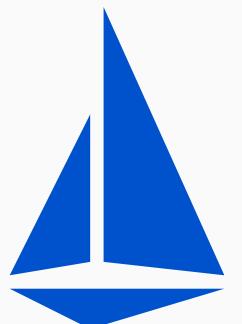
환경 분리

- Terraform workspace 분리 (prod/staging/dev) 후 환경별 변수 설정 및 조건 분기
- GitHub branch 연동



MSA 아키텍처

- **선택 이유:** 서비스 단위를 독립적으로 배포·확장할 수 있어 장애 격리와 유연한 기능 추가가 가능하다.
- **도전 과제:** 서비스 간 통신 복잡성 증가, 데이터 일관성 보장, 장애 추적이 어려워 운영 난도가 높다.



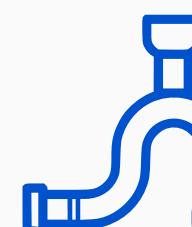
서비스 메시

- **선택 이유:** MSA 환경에서 서비스 간 보안·트래픽 관리·모니터링을 일관 적용하여 운영 가시성이 향상된다.
- **도전 과제:** 높은 러닝커브와 리소스 소모로 복잡한 운영 구조를 안정적으로 구축·운영하기 어렵다.



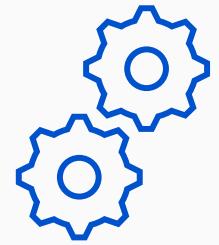
gRPC 통신

- **선택 이유:** REST 대비 빠르고 효율적이며, 스트리밍·양방향 통신으로 고성능 분산 시스템 구현이 가능하다
- **도전 과제:** 프로토콜 버퍼 정의, 변환 처리 등 학습과 운영 난이도가 존재한다



인프라 환경 분리

- **선택 이유:** 개발/스테이징/운영 분리로 안정적 배포와 품질 확보가 가능하다.
- **도전 과제:** 환경별 설정 관리와 CI/CD 파이프라인 복잡성이 증가한다.



외부 API 연동

생성형 AI, OCR, 경로 탐색, 지도

- **선택 이유:** 직접 구현 대신 외부 API 활용으로 빠른 서비스 확장과 풍부한 기능을 제공한다.
- **도전 과제:** API별 상이한 포맷/속도/정책으로 인한 표준화가 필요하고, 비용·호출 제한 관리가 어렵다



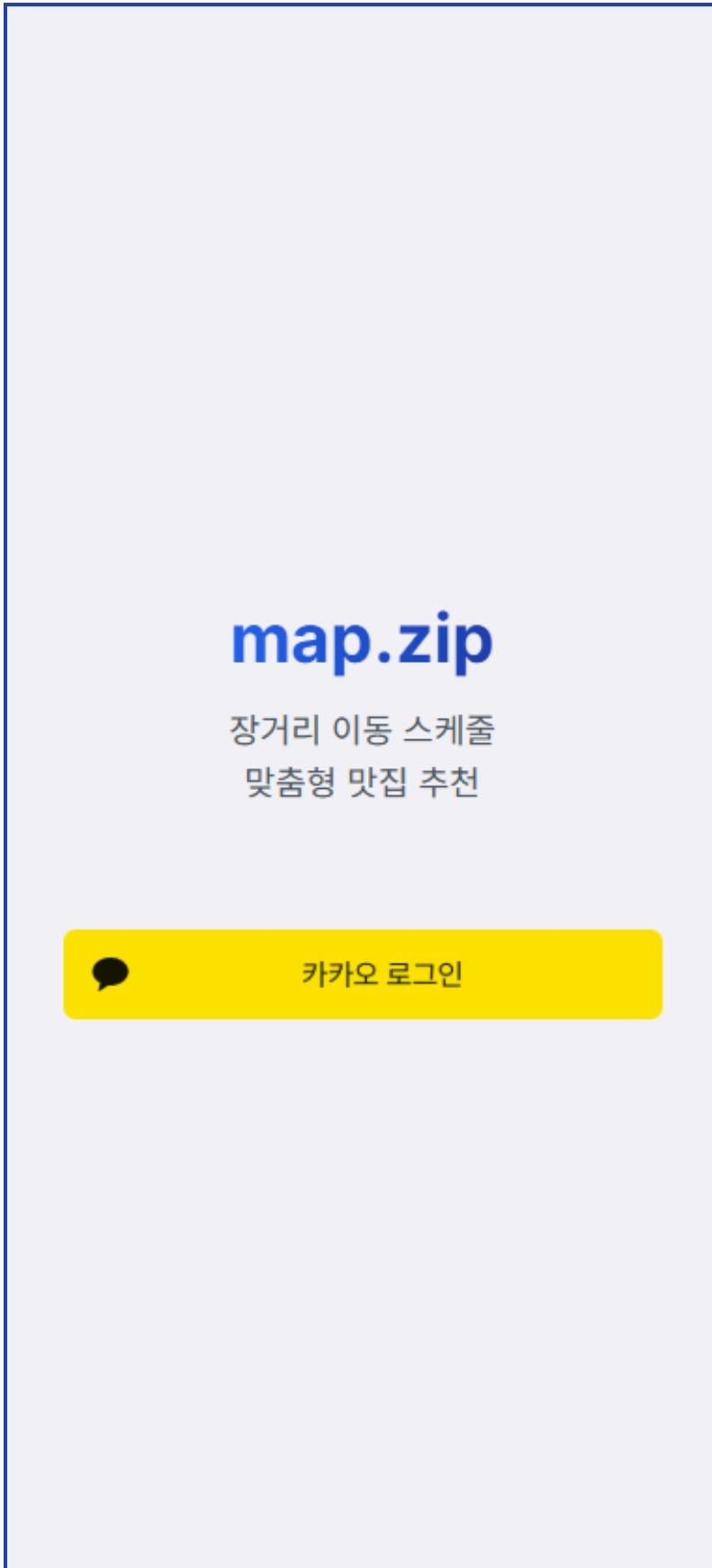
협업 툴 연동

- **선택 이유:** Slack, Jira 등 연동으로 알림·자동화 워크플로우를 제공하여 팀 생산성이 향상된다.
- **도전 과제:** 각 툴 API 이해와 통합, 팀 협업 방식에 맞는 커스터마이징이 필요하다.



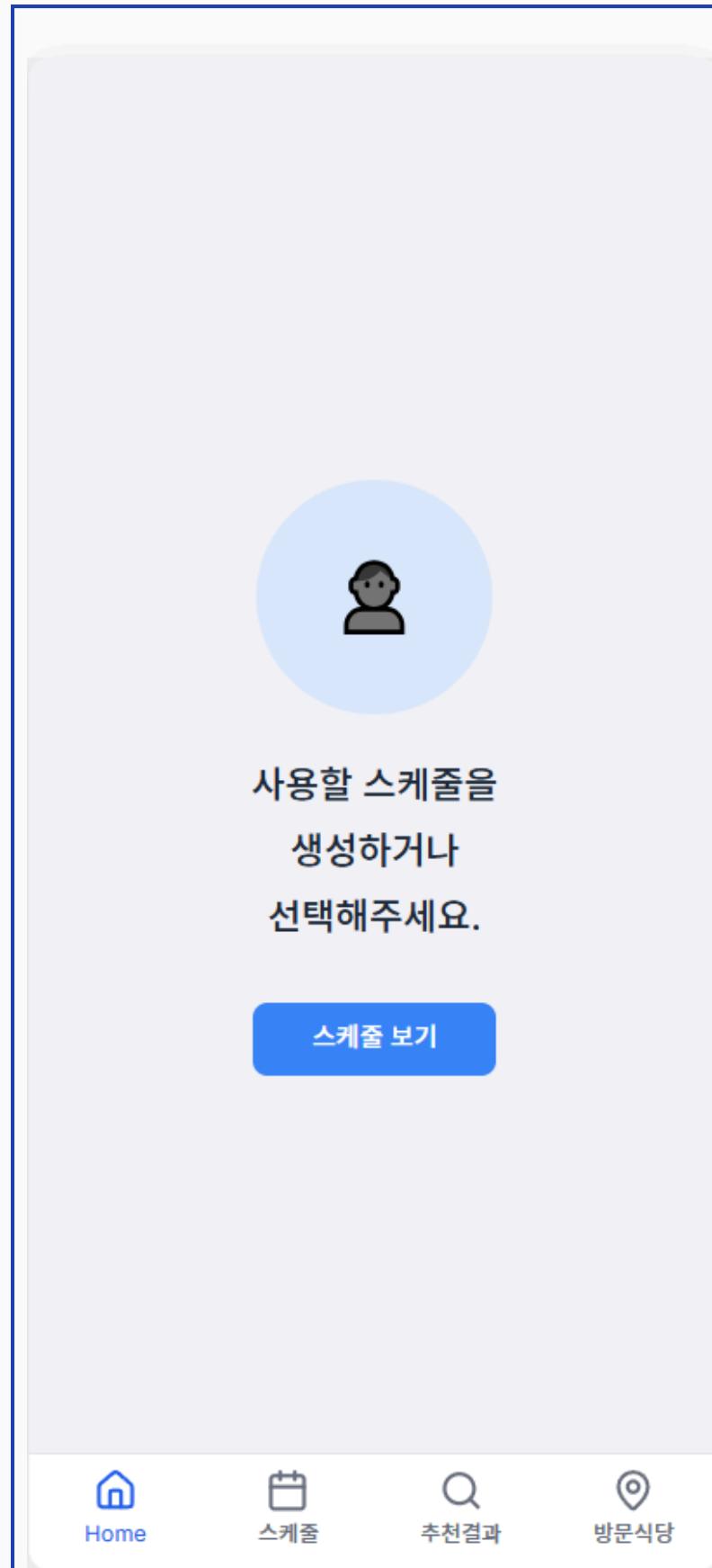
하이브리드 클라우드

- **선택 이유:** 퍼블릭 클라우드 확장성과 온프레미스 보안·비용 효율성을 동시에 활용할 수 있다
- **도전 과제:** 복잡한 네트워크·보안 구성과 워크로드 배치 설계 역량이 필요하다



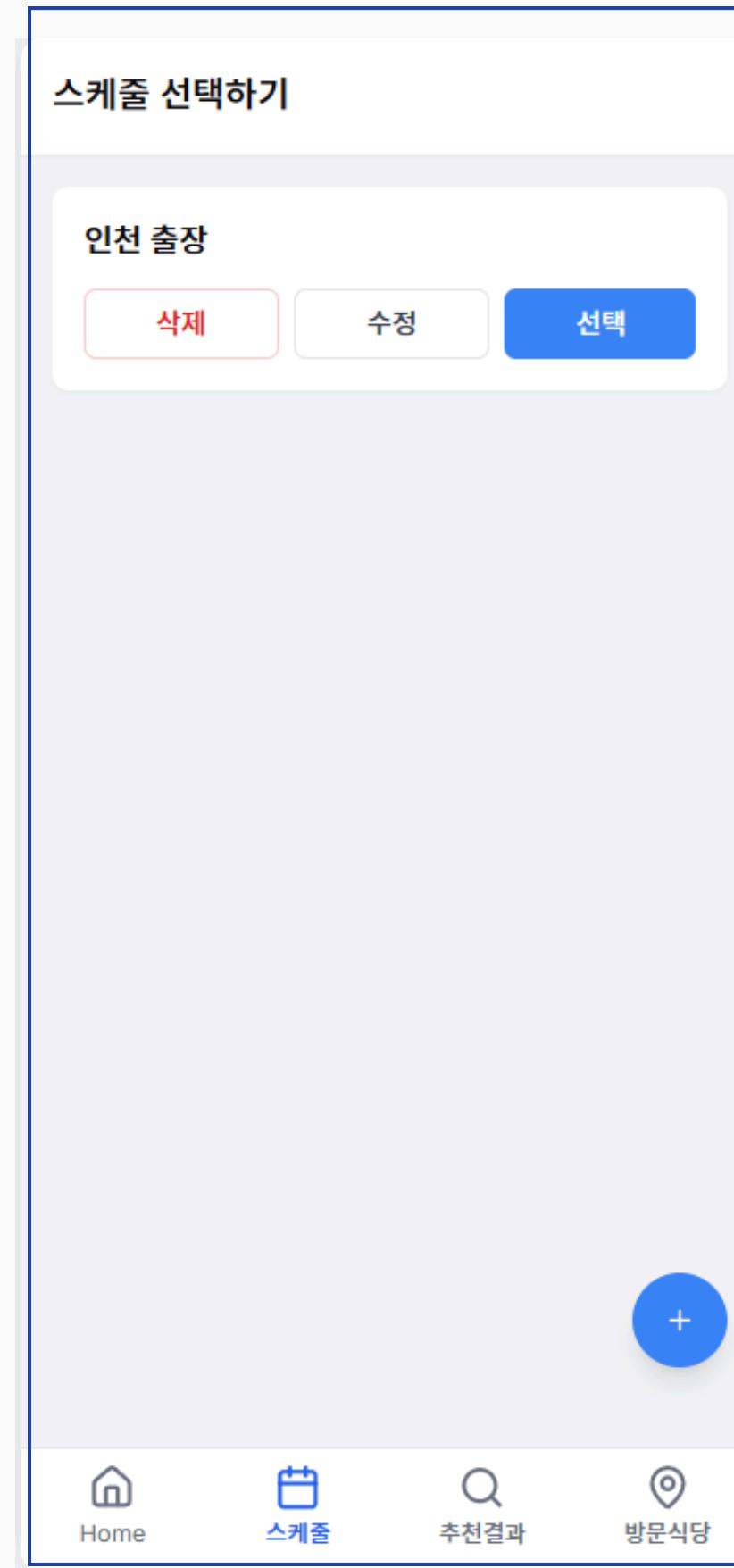
로그인 후 accessToken, refreshToken을 쿠키로 발급

accessToken 만료 시 인터셉터에서 refresh 요청



스케줄을 선택하지 않으면 선택 안내 페이지 노출

스케줄을 선택했으면 추천 결과 페이지 경유 후에 스케줄 요약 페이지 노출



스케줄 생성, 선택, 삭제, 수정 가능

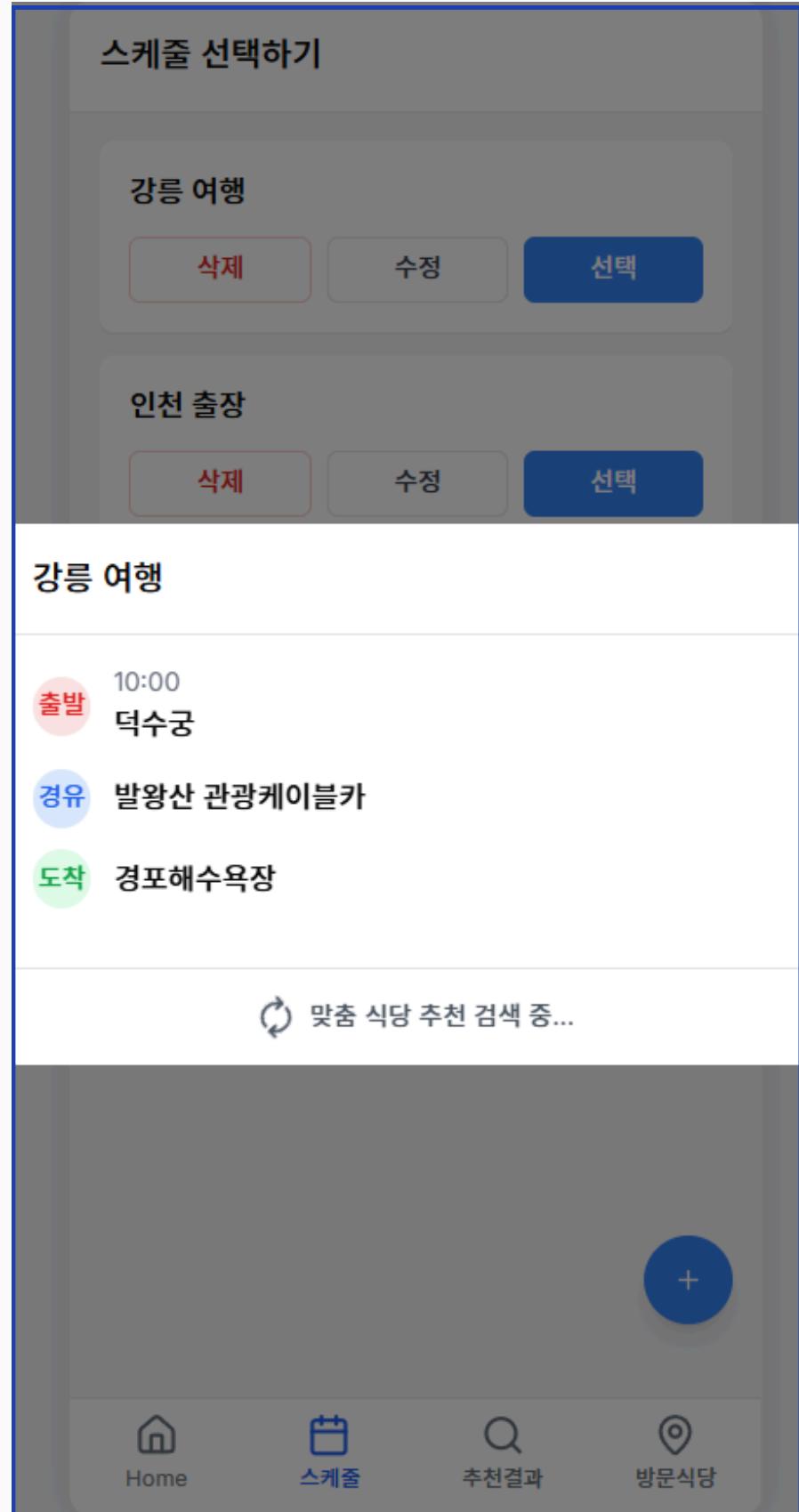
스케줄 선택은 하나만 가능하며 선택 시 추천결과 비동기 요청 시작

The figure illustrates a three-step process for generating a schedule:

- Step 1: 지역명 입력 (Location Input)**
A map of Incheon, South Korea, showing the starting point (경복궁) and destination (인천대공원). A blue dot marks the starting point and a red dot marks the destination. A blue arrow points from the bottom right of this screen to the next screen.
- Step 2: 필수 정보 입력 (Mandatory Information Input)**
This screen requires entering the schedule name (인천 출장), departure time (오후 12:00), and meal time (식사). It also includes meal radius options (5km, 10km, 20km) and a search bar for meal names. A blue arrow points from the bottom right of this screen to the next screen.
- Step 3: 선택 정보 입력 (Optional Information Input)**
This screen allows users to enter optional requirements (e.g., 원하는 맵지 않은 음식), specify the purpose of travel (출장), and choose passengers (혼자, 부모님, 연인, 친구, 자녀). A blue arrow points from the bottom right of this screen to the final step.

- 출발지, 도착지, 경유지 지정

- 필수 정보와 선택 정보 입력 후 생성 완료



추천 결과

사용자의 이동경로와 선호도에 따라 추천된 장소입니다

← 추천 결과 입력완료 (0/1)

식사1 (오후 1:05)

새로운 추천

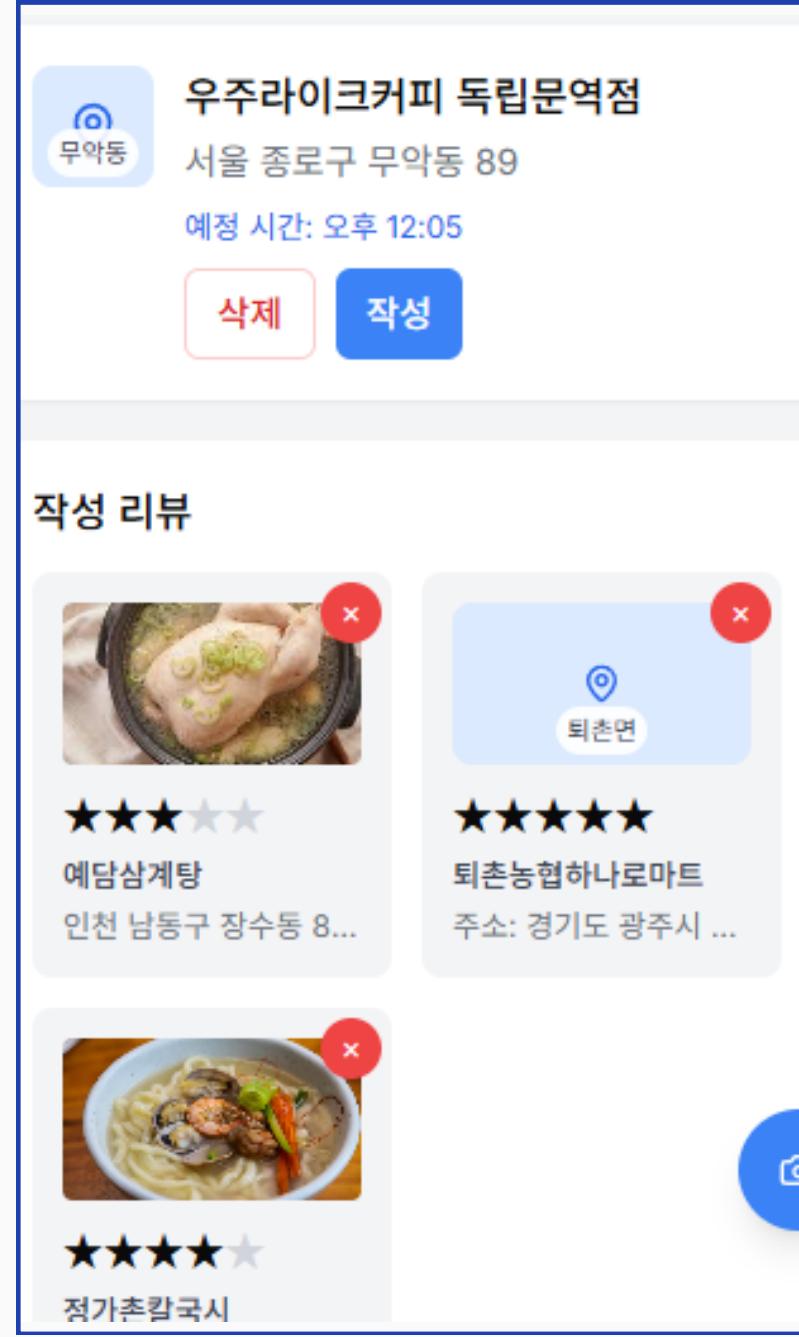
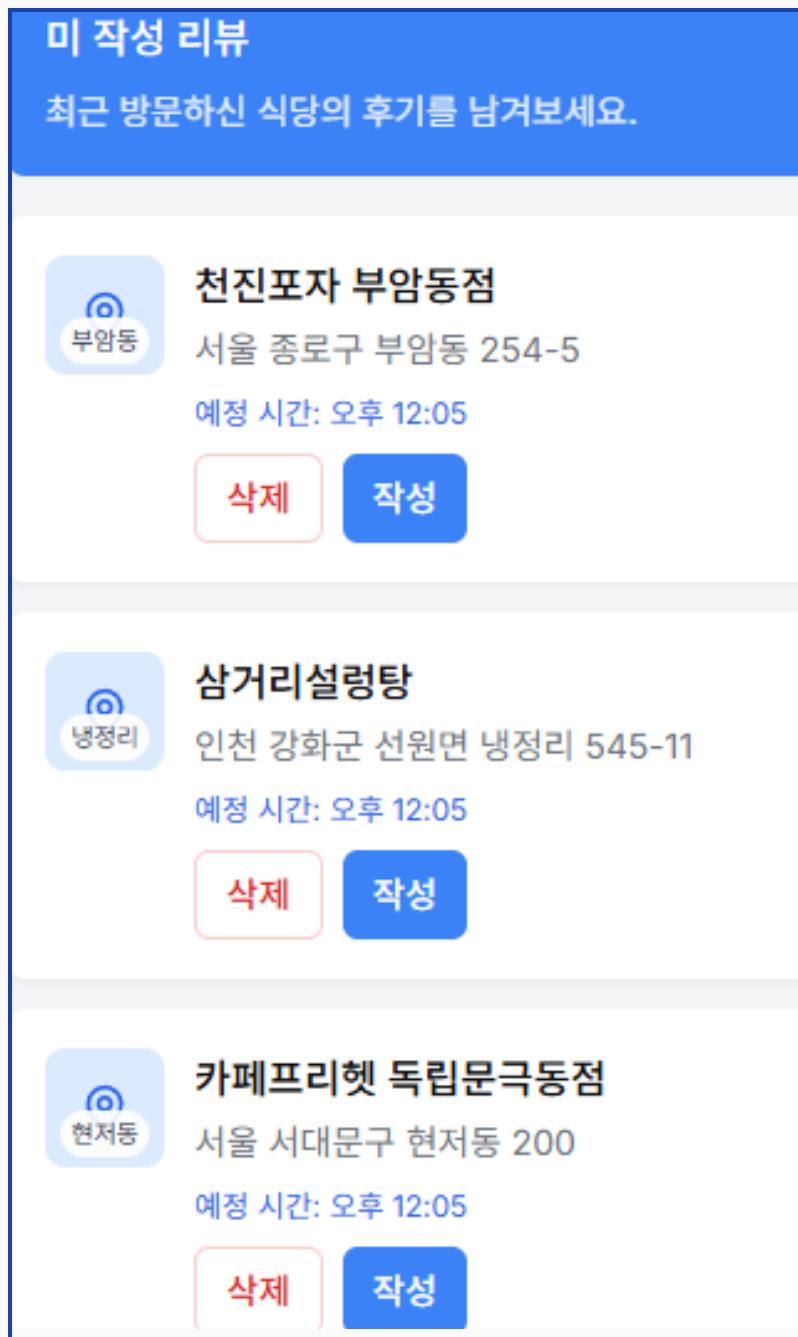
예담삼계탕
예담삼계탕은 시원하면서도 맵지 않은 삼계탕을 제공하여 사용자 요구에 부합함
[카카오 지도](#)

선택하기

녹차한돈갈비
녹차한돈갈비는 육류 전문점으로 출장 중 든든한 식사를 할 수 있음
[카카오 지도](#)

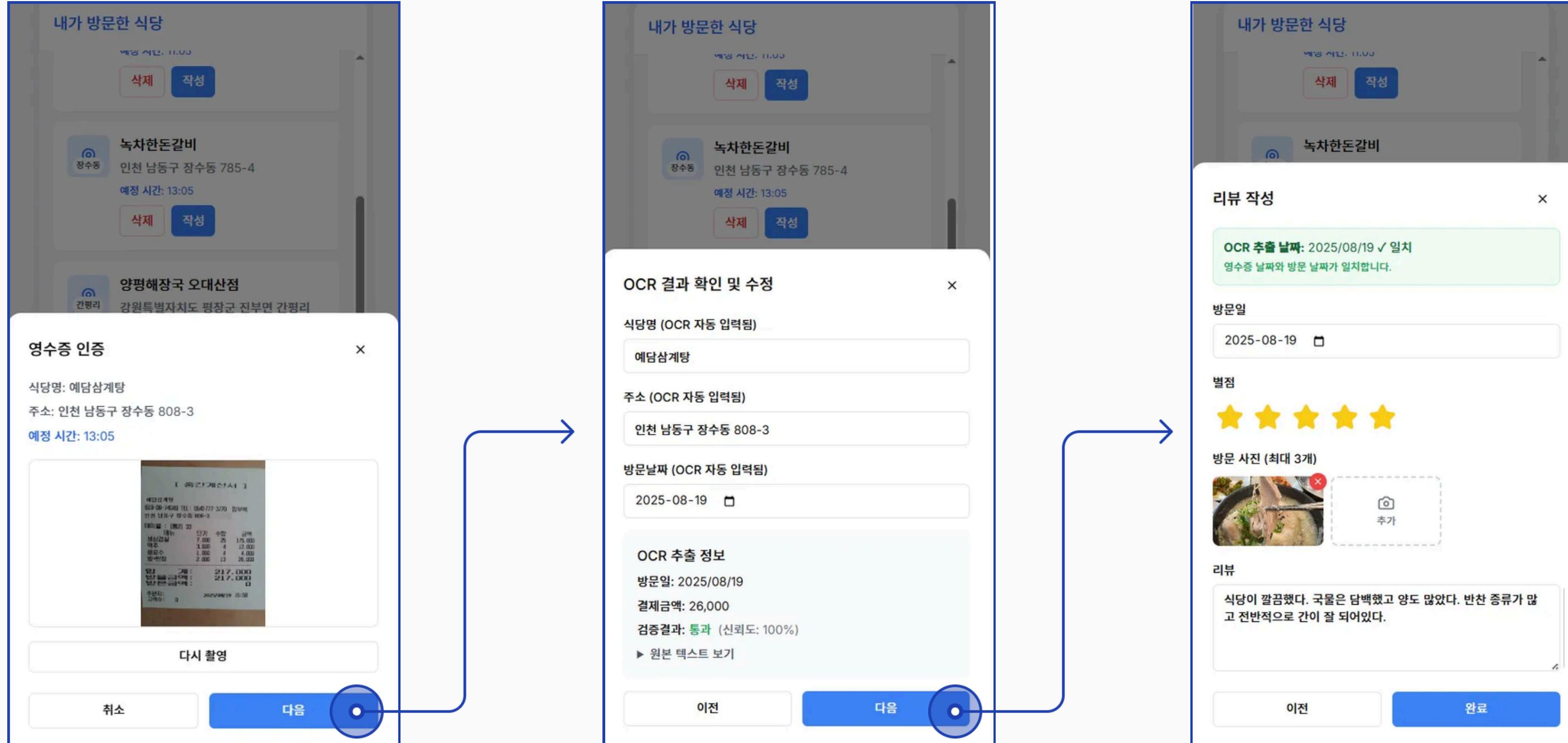
선택하기

- 스케줄 동선 요약 화면 노출 후 프론트엔드에서 주기적으로 pulling하여 비동기 처리
- 추천 결과 페이지에서 각 식사마다 3개의 식당을 추천
- 식당명, AI 추천 사유, 카카오 지도의 URI를 제공하여 보다 상세한 정보를 확인



- 추천결과에서 선택된 식당은 미작성 리뷰목록에 저장되며 미작성 리뷰를 작성하거나 사진모양 아이콘을 클릭하여 새 리뷰 작성 가능
- 미작성 리뷰 목록
 - 식당명과, 주소, 왼쪽에는 주소명 아이콘이 뜨고 삭제 작성이 가능함

- 작성 리뷰 목록
 - 리뷰 작성 시 첨부한 사진과 식당명, 별점 노출
 - 사진이 없을 시 미작성 리뷰와 동일하게 주소명 아이콘이 뜨고 클릭시 상세조회 가능
 - 상세조회창에서 리뷰 수정, 삭제 가능
- 리뷰 상세 보기
 - 작성 된 리뷰 클릭시 별점, 첨부한 사진, 리뷰 노출
 - 작성된 리뷰 수정, 삭제 가능함



- 카메라로 직접 사진을 촬영하거나 선택해 영수증 검증 진행 후에 리뷰 작성 가능
- 영수증 검증 항목: 식당명, 식당 주소,
- 방문 날짜가 식당이 선택된 날짜 이후 일주일 내

별점을 지정할 수 있으며 사진을 최대 3개까지 첨부하여 리뷰 작성 가능

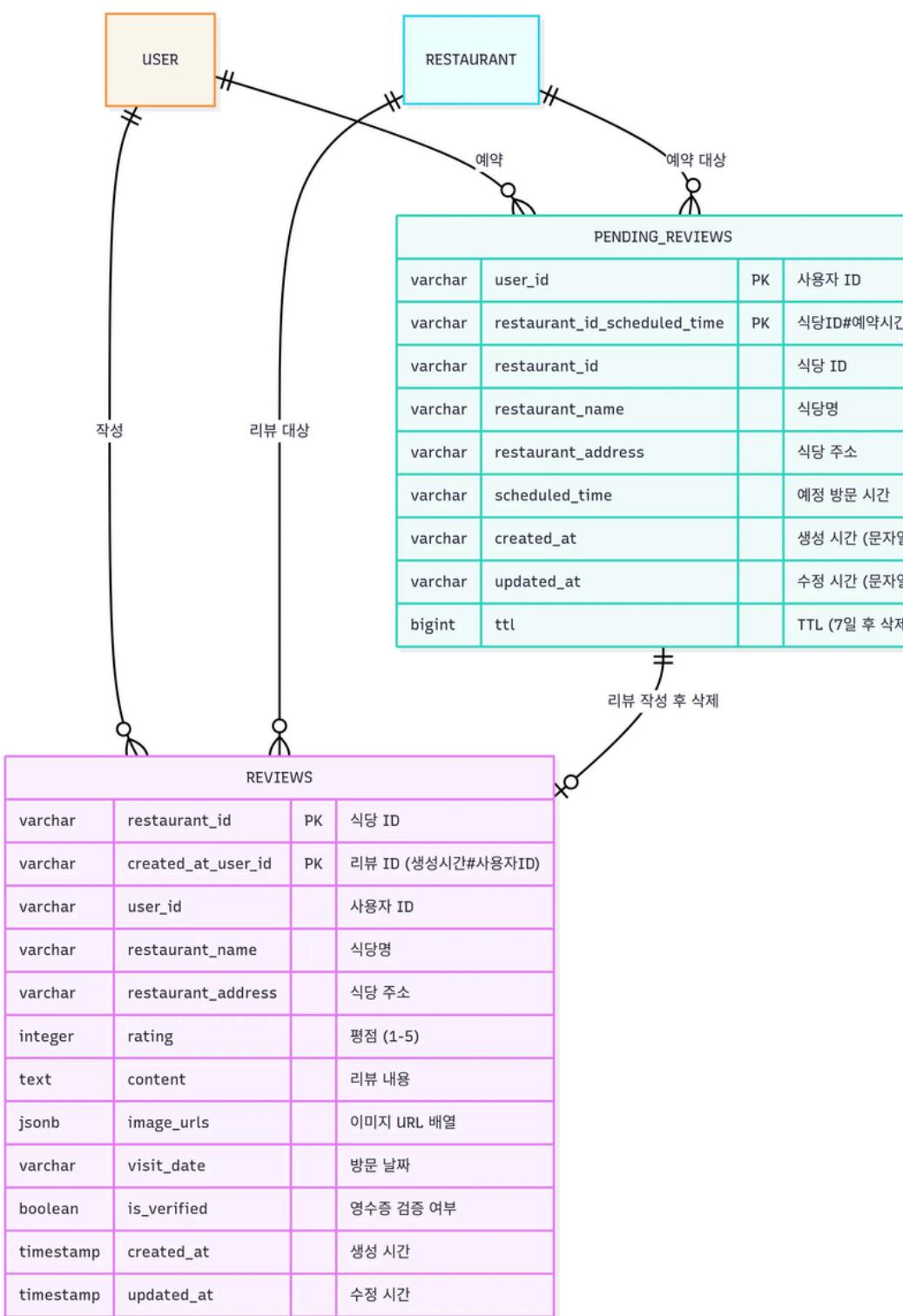
schedules

| schedules | | | |
|-----------|------------------------|----|--------------|
| | | PK | |
| string | id | PK | 스케줄 고유 ID |
| string | user_id | | 사용자 ID |
| string | title | | 스케줄 제목 |
| string | departure_time | | 출발 시간 |
| jsonb | departure_location | | 출발지 정보(JSON) |
| jsonb | destination_location | | 목적지 정보(JSON) |
| jsonb | waypoints | | 경유지 목록(JSON) |
| string | purpose | | 여행 목적 |
| jsonb | companions | | 동행자 목록(JSON) |
| integer | arrival_buffer_minutes | | 도착 여유 시간(분) |
| text | user_note | | 사용자 요구사항 |
| timestamp | created_at | | 생성 시간 |
| timestamp | updated_at | | 수정 시간 |

1:N

| meal_time_slots | | | |
|-----------------|----------------|----|----------------------|
| | | PK | |
| string | id | PK | 시간 슬롯 고유 ID |
| string | schedule_id | FK | 스케줄 ID |
| integer | meal_type | | 식사/간식 구분(0:식사, 1:간식) |
| string | scheduled_time | | 예정 시간 |
| integer | radius | | 검색 반경(미터) |
| timestamp | created_at | | 생성 시간 |

review



recommend

| RECOMMENDATION_SELECTION | | | |
|--------------------------|-----------------------|----|-----------------------------|
| | | PK | |
| bigint | id | PK | Primary Key, Auto Increment |
| double | average_rating | | 평균 평점 |
| integer | meal_type | | 식사 유형 (1:아침, 2:점심, 3:저녁) |
| date | selected_date | | 선택된 날짜 |
| varchar255 | address_name | | 주소명 |
| varchar255 | distance | | 거리 정보 |
| varchar255 | place_id | | 장소 ID (외부 API) |
| varchar255 | place_name | | 장소명 |
| varchar255 | place_url | | 장소 URL |
| varchar255 | reason | | 추천 이유 |
| varchar255 | representative_review | | 대표 리뷰 |
| varchar255 | schedule_id | | 스케줄 ID |
| varchar255 | scheduled_time | | 예약 시간 |
| varchar255 | slot_id | | 슬롯 ID |
| varchar255 | user_id | | 사용자 ID |

schedules

| 컬럼명 | 데이터타입 | NULL허용 | 기본값 | 설명 | 비고 |
|------------------------|--------------|--------|-------------------|-----------------------|-------------|
| id | VARCHAR(50) | N | - | 스케줄 고유 ID | PK |
| user_id | VARCHAR(50) | N | - | 스케줄을 생성한 사용자 ID | - |
| title | VARCHAR(200) | N | - | 스케줄 제목 | - |
| departure_time | VARCHAR(20) | N | - | 출발 시간 (예: "오전 09:30") | - |
| departure_location | JSONB | Y | - | 출발지 정보 (JSON 형태) | Location 객체 |
| destination_location | JSONB | Y | - | 목적지 정보 (JSON 형태) | Location 객체 |
| waypoints | JSONB | Y | - | 경유지 목록 (JSON 형태) | Waypoint 배열 |
| purpose | VARCHAR(200) | Y | - | 여행 목적 | - |
| companions | JSONB | Y | - | 동행자 목록 (JSON 형태) | String 배열 |
| arrival_buffer_minutes | INTEGER | Y | - | 도착 여유 시간 (분) | - |
| user_note | TEXT | Y | - | 사용자 요구사항 및 메모 | - |
| created_at | TIMESTAMP | N | CURRENT_TIMESTAMP | 레코드 생성 시간 | 자동 생성 |
| updated_at | TIMESTAMP | N | CURRENT_TIMESTAMP | 레코드 마지막 수정 시간 | 자동 업데이트 |

사용자의 여행 스케줄 정보를 저장하는 메인 테이블

meal_time_slots

| 컬럼명 | 데이터타입 | NULL허용 | 기본값 | 설명 | 비고 |
|----------------|-------------|--------|-------------------|------------------------------|-------------------|
| id | VARCHAR(50) | N | - | 시간 슬롯 고유 ID | PK |
| schedule_id | VARCHAR(50) | N | - | 연관된 스케줄 ID | FK → schedules.id |
| meal_type | INTEGER | N | - | 식사/간식 구분 (0:식사, 1:간식) | - |
| scheduled_time | VARCHAR(20) | N | - | 예정된 식사/간식 시간 (예: "오후 12:30") | - |
| radius | INTEGER | N | 1000 | 맛집 검색 반경 (미터 단위) | 기본값 1000m |
| created_at | TIMESTAMP | N | CURRENT_TIMESTAMP | 레코드 생성 시간 | 자동 생성 |

스케줄별 식사/간식 시간 슬롯 정보를 저장하는 테이블

review

| 컬럼명 | 데이터타입 | NULL허용 | 기본값 | 설명 | 비고 |
|--------------------|--------------|--------|-------------------|-------------------------|--------------|
| restaurant_id | VARCHAR(50) | N | - | 식당 고유 ID | PK |
| created_at_user_id | VARCHAR(100) | N | - | 리뷰 고유 ID (생성시간#사용자 ID) | PK, Sort Key |
| user_id | VARCHAR(50) | N | - | 리뷰 작성자 사용자 ID | - |
| restaurant_name | VARCHAR(200) | Y | - | 식당명 | - |
| restaurant_address | VARCHAR(500) | Y | - | 식당 주소 | - |
| rating | INTEGER | Y | - | 평점 (1-5점) | CHECK 제약조건 |
| content | TEXT | Y | - | 리뷰 내용 | - |
| image_urls | JSONB | Y | - | 리뷰 이미지 URL 목록 (JSON 형태) | String 배열 |
| visit_date | VARCHAR(20) | Y | - | 방문 날짜 (ISO 형식) | - |
| is_verified | BOOLEAN | Y | FALSE | 영수증 검증 여부 | OCR 검증 결과 |
| created_at | TIMESTAMP | N | CURRENT_TIMESTAMP | 레코드 생성 시간 | 자동 생성 |
| updated_at | TIMESTAMP | N | CURRENT_TIMESTAMP | 레코드 마지막 수정 시간 | 자동 업데이트 |

사용자가 작성한 리뷰 정보를 저장하는 메인 테이블

pending_review

| 컬럼명 | 데이터타입 | NULL허용 | 기본값 | 설명 | 비고 |
|-------------------------------|--------------|--------|-----|----------------------------|--------------|
| user_id | VARCHAR(50) | N | - | 사용자 ID | PK |
| restaurant_id_sc heduled_time | VARCHAR(150) | N | - | 복합키 (식당ID# 예약시간) | PK, Sort Key |
| restaurant_id | VARCHAR(50) | N | - | 식당 고유 ID | - |
| restaurant_name | VARCHAR(200) | Y | - | 식당명 | - |
| restaurant_address | VARCHAR(500) | Y | - | 식당 주소 | - |
| scheduled_time | VARCHAR(30) | N | - | 예정된 방문 시간 (ISO 형식) | - |
| created_at | VARCHAR(30) | N | - | 레코드 생성 시간 (ISO 형식) | 문자열로 저장 |
| updated_at | VARCHAR(30) | N | - | 레코드 마지막 수정 시간 (ISO 형식) | 문자열로 저장 |
| ttl | BIGINT | Y | - | TTL 만료 시간 (Unix timestamp) | 7일 후 자동 삭제 |

방문 예정이지만 아직 리뷰를 작성하지 않은 식당 정보를 저장하는 테이블

recommendation_selection

| 컬럼명 | 데이터타입 | NULL허용 | 기본값 | 설명 | 비고 |
|-----------------------|------------------|--------|-----|--------------------|--------------------------|
| id | BIGSERIAL | N | - | 고유 ID | PK (자동 증가) |
| user_id | VARCHAR(50) | N | - | 사용자 ID | - |
| schedule_id | VARCHAR(50) | N | - | 스케줄 ID | FK → schedules(id) |
| slot_id | VARCHAR(50) | N | - | 슬롯 ID | FK → meal_time_slots(id) |
| place_id | VARCHAR(100) | N | - | 장소 고유 ID | - |
| place_name | VARCHAR(200) | N | - | 장소 이름 | - |
| meal_type | INTEGER | Y | - | 식사 유형 (0=식사, 1=간식) | - |
| scheduled_time | VARCHAR(20) | Y | - | 예정 시간 (예: "12:30") | - |
| reason | TEXT | Y | - | 추천 이유 | - |
| distance | VARCHAR(50) | Y | - | 거리 정보 (예: "500m") | - |
| place_url | TEXT | Y | - | 장소 상세 URL | - |
| address_name | TEXT | Y | - | 장소 주소 | - |
| selected_date | DATE | Y | - | 사용자가 선택한 날짜 | - |
| average_rating | DOUBLE PRECISION | Y | - | 평균 평점 | - |
| representative_review | TEXT | Y | - | 대표 리뷰 | - |

선택한 식당 정보를 저장하는 테이블

추천 서버

Key-Value 전략

- 기본 패턴: recommend:{userId}:{scheduleId}:{slotId}:{placeN}
 - 슬롯 단위로 후보 최대 3개 저장 (place1, place2, place3)
 - 값은 PlaceInfo JSON (id, 이름, 거리, URL, 평점 등)
- 상태 관리 키
 - recommend:{userId}:{scheduleId}:runId → 요청 버전(UUID)
- 스케줄 핵심 정보 관리 키
 - scheduleDetail:{userId}:{scheduleId} → Hash 구조 (출발/도착/경유지/업데이트 이벤트)

TTL 정책

- 모든 키는 자정(23:59:59) 만료
- 하루 단위 스케줄/추천 데이터는 하루 안에서만 유효
- 부분 업데이트 시 해당 슬롯 키만 덮어쓰되 TTL은 동일하게 유지
- 한 사용자가 스케줄 선택시 다른 스케줄에 대한 정보들은 삭제

인증 서버

Key-Value 전략

- 기본 패턴: rt:token:{tokenHash} → kakaold (String)
 - 용도: 리프레시 토큰 → 사용자 식별 (멀티 세션 허용)

TTL 정책

- rt:token:* 전부 1일
- 재발급 시 이전 토큰 삭제, 새 토큰 설정 (+TTL)
- 단일 로그아웃: 해당 토큰 키만 삭제

스케줄 서버

Key-Value 전략

- 기본 패턴: user:{userId}:selectedSchedule
 - userId를 동적으로 포함하는 문자열 키 패턴을 사용합니다.

TTL 정책

- 사용자가 특정 스케줄을 선택하면, 해당 Key-Value 데이터는 24시간의 유효기간을 갖게 됩니다.

리뷰 서버

1. 캐시 (Cache)

Key-Value 전략

- 사용자별 리뷰 목록: userReviews::{userId}{page}{size}
- 값: List<ReviewEntity> (직렬화된 객체)
- 식당별 리뷰 목록: restaurantReviews::{restaurantId}{page}{size}
- 값: List<ReviewEntity> (직렬화된 객체)
- 리뷰 통계:
- reviewStats::count_{restaurantId} → 리뷰 개수 (Long)
- reviewStats::avg_rating_{restaurantId} → 평균 평점 (Double)
- 단일 리뷰:
- singleReview::{restaurantId}_{reviewId} → ReviewEntity (직렬화된 객체)
- singleReviewById::{reviewId} → ReviewEntity (직렬화된 객체)

TTL 정책

- 모든 캐시 키는 기본 10분으로 설정
- 리뷰가 생성, 수정, 삭제될 때마다, @CacheEvict 어노테이션을 통해 관련된 모든 캐시(userReviews, restaurantReviews, reviewStats 등) 즉시 삭제 → 최신 상태 유지

2. 미작성 리뷰 후보 관리

Key-Value 전략

- Key: places_for_review:{userId}
- Value: PlaceData 정보가 담긴 JSON 문자열 목록
 - 예: "[{'id':..., 'placeName':..., 'addressName':..., ...}]"

TTL 정책

- 7일로 설정
- 사용자가 새로운 스케줄을 선택하고 추천을 받으면, 기존 키(places_for_review:{userId})는 삭제되고 새로운 식당 목록으로 덮어쓰여짐

| RPC Method | Service | 설명 | Request Type | Response Type | 비고 |
|-------------------|-----------------|----------------------|--------------------------|---------------------------|--------|
| 스케줄 | | | | | |
| GetScheduleDetail | ScheduleService | 특정 스케줄의 상세 정보를 조회합니다 | GetScheduleDetailRequest | GetScheduleDetailResponse | 순수 조회용 |

| RPC Method | Service | 설명 | Request Type | Response Type | 비고 |
|-----------------------------------|---------------|-----------------------|------------------------------------|-------------------------------------|-------------------------|
| 리뷰 | | | | | |
| GetReviewsForRecommendation | ReviewService | 추천 시스템용 리뷰 데이터를 조회합니다 | GetReviewsForRecommendationRequest | GetReviewsForRecommendationResponse | 추천 서버 전용, 지역/카테고리별 리뷰 |
| StorePlacesForReview | ReviewService | 선택된 식당을 미작성 리뷰로 저장합니다 | StorePlacesForReviewRequest | StorePlacesForReviewResponse | 추천 서버에서 호출, 사용자 선택 완료 시 |
| GetReviewSummaryForRecommendation | ReviewService | 특정 식당들의 리뷰 요약을 제공합니다 | GetReviewSummaryRequest | GetReviewSummaryResponse | 추천 서버 전용, 평점/대표리뷰 제공 |

| RPC Method | Service | 설명 | Request Type | Response Type | 비고 |
|-------------------|------------------|------------------------|----------------------------------|-----------------------------------|----------------------|
| 추천 | | | | | |
| GetScheduleDetail | RecommendService | 특정 스케줄에 대한 내용 조회 | GetSelectedScheduleDetailRequest | GetSelectedScheduleDetailResponse | 추천 요청시 스케줄 서버에 정보 받기 |
| PlacesForReview | RecommendService | 리뷰 서버용: 리뷰 대상 식당 목록 제공 | PlacesForReviewRequest | PlacesForReviewResponse | 추천 서버 → 리뷰 서버 호출 |

| API | Method | Endpoint | 설명 | Request | Response | Status Code |
|------------|--------|----------------------|---|-------------------|------------------------------|----------------------------|
| 인증 | | | | | | |
| 토ken 생성 | POST | /auth/kakao/callback | access, refresh 토큰 생성, 쿠키로 응답합니다 | AuthorizationCode | access, refresh token cookie | 200: 생성 성공 400: 생성 실패 |
| 카카오 아이디 조회 | GET | /auth/me/kakao/id | 고유 kakao id를 조회합니다 | - | Kakaoid | 200: 조회 성공 400: 조회 실패 |
| 토큰 재발급 | POST | /auth/token/refresh | refresh token을 통해 access token을 재발급 합니다 | RefreshToken | access token cookie | 200: 재발급 성공 400: 재발급 실패 |
| 로그아웃 | POST | /auth/token/logout | access, refresh 토큰 삭제, 쿠키 만료 시킵니다 | RefreshToken | expired cookie | 200: 삭제 성공 400: 삭제 실패 |

| API | Method | Endpoint | 설명 | Request | Response | Status Code |
|--------------|--------|-------------------------------|------------------------|------------------------|----------------------------|--|
| 스케줄 | | | | | | |
| 스케줄 생성 | POST | /schedule | 새로운 스케줄을 생성합니다 | CreateSchedule Request | CreateSchedule Response | 201: 생성 성공 400: 잘못된 요청 401: 인증 실패 |
| 스케줄 목록 조회 | GET | /schedule | 사용자의 스케줄 목록을 조회합니다 | - | GetScheduleList Response | 200: 조회 성공 401: 인증 실패 |
| 스케줄 상세 조회 | GET | /schedule/{scheduleId} | 특정 스케줄의 상세 정보를 조회합니다 | - | GetScheduleDetailResponse | 200: 조회 성공 404: 스케줄 없음 401: 인증 실패 |
| 스케줄 선택 | GET | /schedule/{scheduleId}:select | 스케줄을 선택하고 상세 정보를 조회합니다 | - | GetScheduleDetailResponse | 200: 선택 성공 404: 스케줄 없음 401: 인증 실패 |
| 스케줄 선택 상태 조회 | GET | /schedule/selectedStatus | 사용자의 스케줄 선택 상태를 확인합니다 | - | IsScheduleSelectedResponse | 200: 조회 성공 401: 인증 실패 |
| 스케줄 수정 | PUT | /schedule/{scheduleId} | 기존 스케줄을 수정합니다 | UpdateScheduleRequest | GetScheduleDetailResponse | 200: 수정 성공 404: 스케줄 없음 400: 잘못된 요청 401: 인증 실패 |
| 스케줄 삭제 | DELETE | /schedule/{scheduleId} | 스케줄을 삭제합니다 | - | DeleteSchedule Response | 200: 삭제 성공 404: 스케줄 없음 401: 인증 실패 |
| 스케줄 선택 해제 | DELETE | /schedule/selection | 선택된 스케줄을 해제합니다 | - | DeselectScheduleResponse | 200: 해제 성공 401: 인증 실패 |

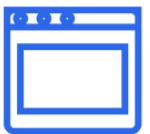
| API | Method | Endpoint | 설명 | Request | Response | Status Code |
|---------------------|--------|--|-------------------------------|-----------------------------|--------------------------------|--|
| 리뷰 | | | | | | |
| 리뷰 작성 | POST | /review | 새로운 리뷰를 작성합니다 (영수증 OCR 검증 포함) | CreateReviewRequest | CreateReviewResponse | 201: 작성 성공 400: 잘못된 요청 401: 인증 실패 422: OCR 검증 실패 |
| 사용자 리뷰 목록 조회 | GET | /review/user | 사용자가 작성한 리뷰 목록을 조회합니다 | page, size (query params) | GetUserReviews Response | 200: 조회 성공 401: 인증 실패 |
| 리뷰 상세 조회 | GET | /review/{restaurant_id}/{review_id} | 특정 리뷰의 상세 정보를 조회합니다 | - | GetReviewResponse | 200: 조회 성공 404: 리뷰 없음 401: 인증 실패 |
| 리뷰 수정 | PUT | /review/{restaurant_id}/{review_id} | 기존 리뷰를 수정합니다 | UpdateReviewRequest | UpdateReviewResponse | 200: 수정 성공 404: 리뷰 없음 400: 잘못된 요청 401: 인증 실패 |
| 리뷰 삭제 | DELETE | /review/{restaurant_id}/{review_id} | 리뷰를 삭제합니다 | - | DeleteReviewResponse | 200: 수정 성공 404: 리뷰 없음 400: 잘못된 요청 401: 인증 실패 |
| 리뷰 삭제 | PUT | /review/{restaurant_id}/{review_id} | 기존 스케줄을 수정합니다 | UpdateScheduleRequest | GetScheduleDetailResponse | 200: 수정 성공 404: 스케줄 없음 400: 잘못된 요청 401: 인증 실패 |
| 영수증 검증 | POST | /review/verify-receipt | 영수증 이미지를 OCR로 검증합니다 | VerifyReceiptRequest | VerifyReceiptResponse | 200: 검증 완료 400: 잘못된 요청 401: 인증 실패 422: 검증 실패 |
| 미작성 리뷰 목록 조회 | GET | /review/pending | 사용자의 미작성 리뷰 목록을 조회합니다 | - | GetPlacesForReviewResponse | 200: 조회 성공 401: 인증 실패 |
| 미작성 리뷰 상세 조회 | GET | /review/pending/{restaurant_id}/detail | 특정 미작성 리뷰의 상세 정보를 조회합니다 | scheduledTime (query param) | GetPendingReviewDetailResponse | 200: 조회 성공 404: 미작성 리뷰 없음 401: 인증 실패 |
| 미작성 리뷰 삭제 | DELETE | /review/pending/{restaurant_id} | 미작성 리뷰를 삭제합니다 (안간 경우) | scheduledTime (query param) | DeletePendingReviewResponse | 200: 삭제 성공 404: 미작성 리뷰 없음 401: 인증 실패 |

| API | Method | Endpoint | 설명 | Request | Response | Status Code |
|---------------------|--------|----------------------------------|---|----------------------------------|-----------------------------------|---|
| 추천 | | | | | | |
| 추천 요청/업데이트 트리거 | POST | /recommend/request | 스케줄 기반 추천 실행 또는 현재 위치/시간 반영 업데이트 | RecommendRequest | RecommendResponse | 200: 요청 성공 400: 잘못된 요청 401: 인증 실패 500: 서버 오류 |
| 추천 결과 조회 | GET | /recommend/result | 슬롯별 추천 결과 조회 (PENDING/OK 포함, 풀링용) (추천한 식당 선택화면에 필요한 정보) | GetRecommendationResultsRequest | GetRecommendationResultsResponse | 200: 조회 성공 202: 처리 중 (PENDING) 400: 파라미터 오류 401: 인증 실패 404: 결과 없음 500: 서버 오류 |
| 선택 결과 제출 | POST | /recommend/submit | 사용자가 슬롯별 선택 식당을 확정 제출 | SelectedPlaceRequest | SubmitResponse | 200: 저장 성공 400: 유효성 실패 401: 인증 실패 409: 중복 제출 500: 서버 오류 |
| 스케줄 상세 조회 | GET | /recommend/schedule/{scheduleId} | 스케줄 요약 화면에 필요한 스케줄 정보 조회 | GetSelectedScheduleDetailRequest | GetSelectedScheduleDetailResponse | 200: 조회 성공 400: 파라미터 오류 401: 인증 실패 404: 스케줄 없음 500: 서버 오류 |



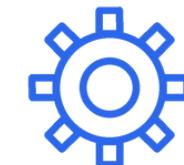
Backend

- **변경 감지:** Git diff로 변경된 마이크로서비스만 식별
- **병렬 빌드:** Matrix 전략으로 여러 서비스 동시 빌드
- **컨테이너화:** Maven 빌드 → Docker 이미지 생성 → ECR 푸시
- **GitOps 연동:** 성공한 서비스만 Infra 레포의 ArgoCD YAML 업데이트



Frontend

- **정적 사이트 생성:** Next.js 빌드로 정적 파일 생성
- **AWS 배포:** S3 업로드 + CloudFront 캐시 무효화
- **환경 변수 주입:** 빌드 시 API URL, 카카오맵 키 등 설정



Config

- **설정 변경 감지:** 개별 서비스 설정 vs 공통 설정 구분
- **중복 배포 방지:** Backend 변경과 겹치는 서비스 제외
- **무중단 재시작:** kubectl로 Kubernetes 배포 롤링 재시작



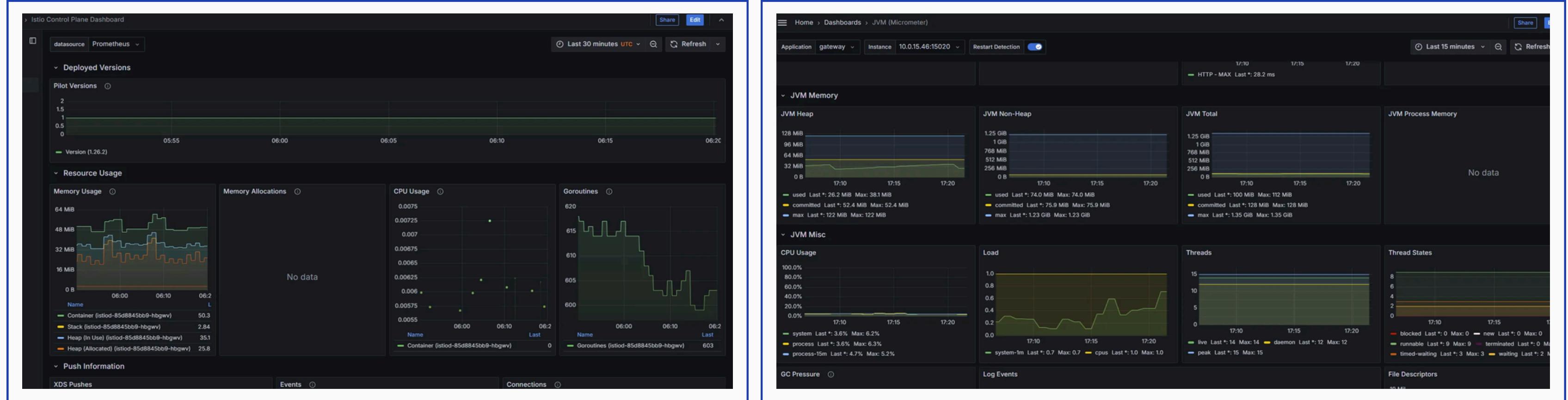
Protocol Buffer

- **Proto 파일 수집:** 모든 .pb 파일을 하나의 이미지로 패키징
- **InitContainer 업데이트:** 서비스 시작 전 최신 Proto 파일 복사
- **자동 배포:** ArgoCD YAML의 InitContainer 이미지 태그 업데이트



Jira

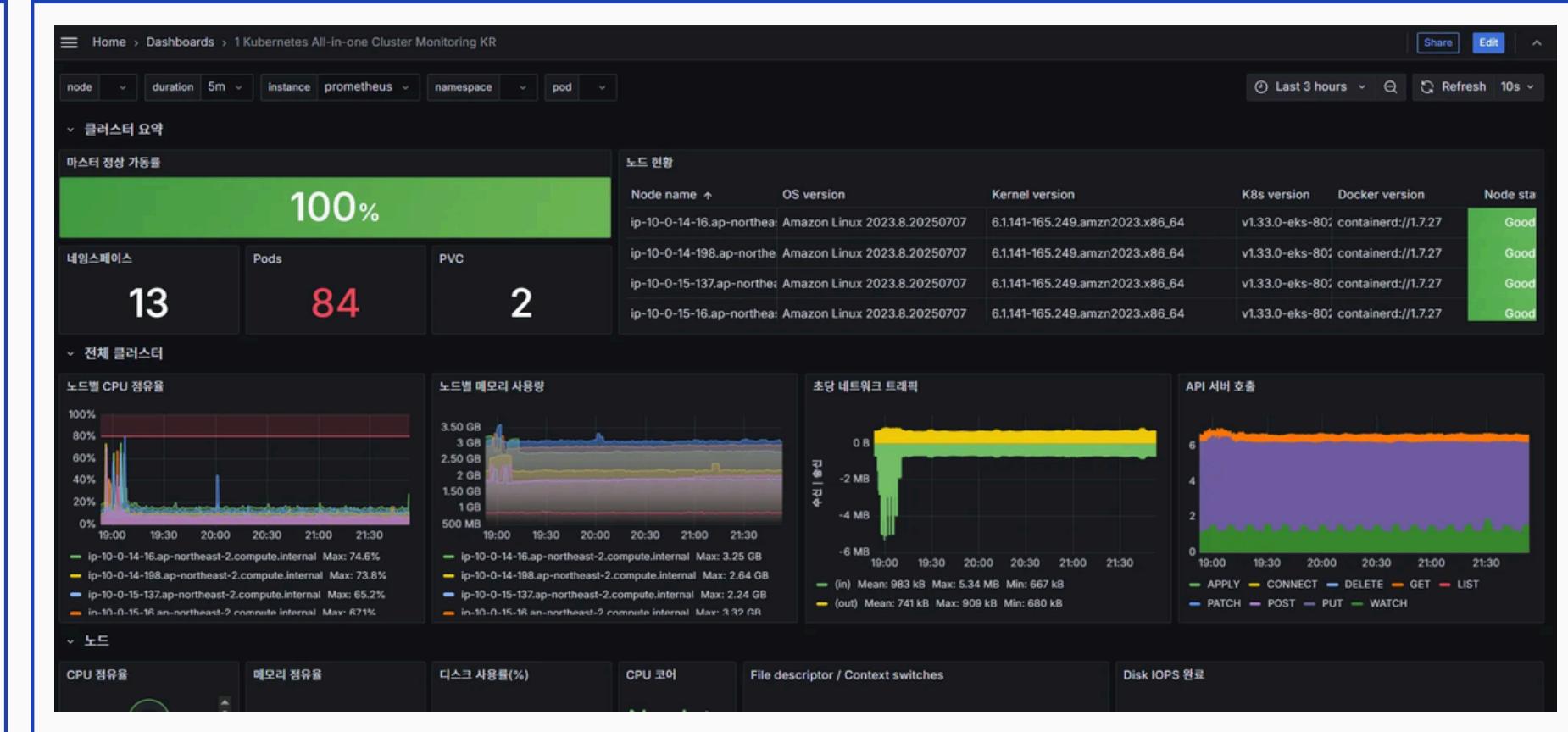
- **이슈 동기화:** GitHub 이슈 → Jira 티켓 자동 생성
- **브랜치 생성:** Jira 티켓 번호로 개발 브랜치 자동 생성
- **상태 동기화:** GitHub 이슈 종료 시 Jira 티켓도 완료 처리



ISTIO CONTROL PLANE

- Prometheus를 통해 Istio, Java Springboot 등 각 메트릭을 수집하고 CloudWatch 메트릭을 Grafana와 연동해 대시보드 구성

JVM



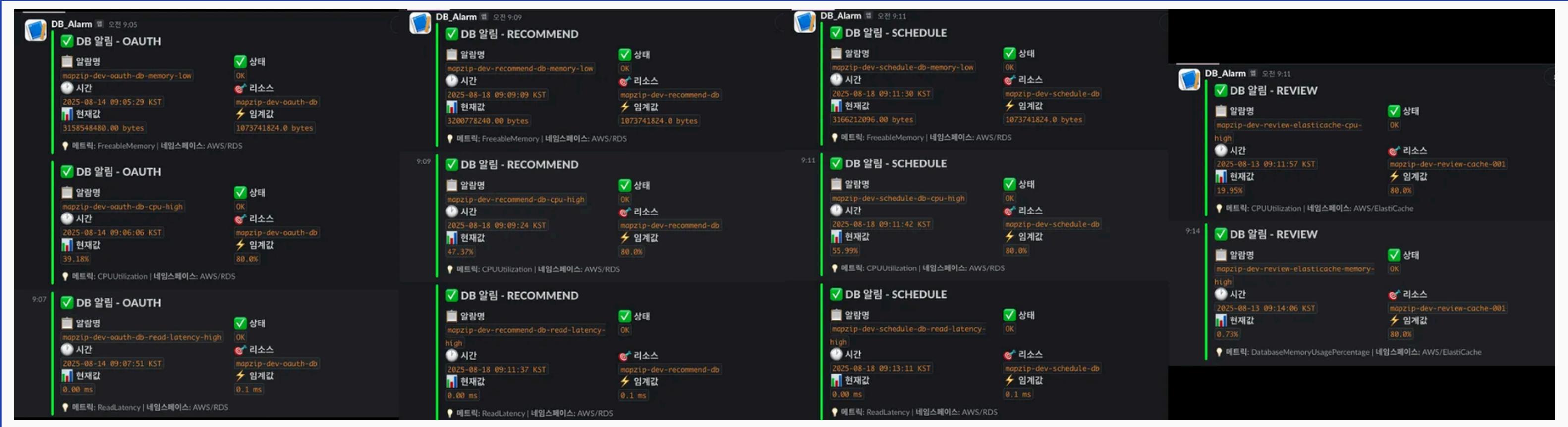
ELASTICACHE

KUBERNETES



RDS

각 서비스 알람



알람 메트릭

데이터베이스 슬랙 채널별 알람

- 장애 전 징후를 탐지하고 실시간으로 대응하도록 서비스별 Slack 채널에 Lambda 기반 실시간 알람 시스템 구축

| 서비스 | 메트릭 | 임계값 |
|---------------------|---------------|--------|
| Aurora | CPU Util | >80% |
| | Memory | <1GB |
| | Read Latency | >100ms |
| | Write Latency | >100ms |
| ElastiCache(Valkey) | CPU Util | >80% |
| | DB Usage | >80% |
| DynamoDB | Read Request | >5 |
| | Write Request | >5 |

 argocd-notifications-bot 웹 오후 8:06
✓ 애플리케이션 mapzip-dev-service-recommend 이 성공적으로 배포되었습니다.

mapzip-dev-service-recommend - 정상 상태
동기화 상태 Synced
Repository  https://github.com/CLD3rd-Team4/Infra

리비전
c0724f105e901fc7049c9d693f8192af38a
0243e

 argocd-notifications-bot 웹 오후 11:35
✓ 애플리케이션 mapzip-dev-service-platform 이 성공적으로 배포되었습니다.

mapzip-dev-service-platform - 정상 상태
동기화 상태 Synced
Repository  https://github.com/CLD3rd-Team4/Infra

리비전
a6d6335eb22dcee4f107112256aef4c841
24ef33

 argocd-notifications-bot 웹 오전 12:38
✓ 애플리케이션 mapzip-dev-service-review 이 성공적으로 배포되었습니다.

mapzip-dev-service-review - 정상 상태
동기화 상태 Synced
Repository  https://github.com/CLD3rd-Team4/Infra

리비전
f1ef3f94098d80c03d495667f0eb2de2e13
79fc7

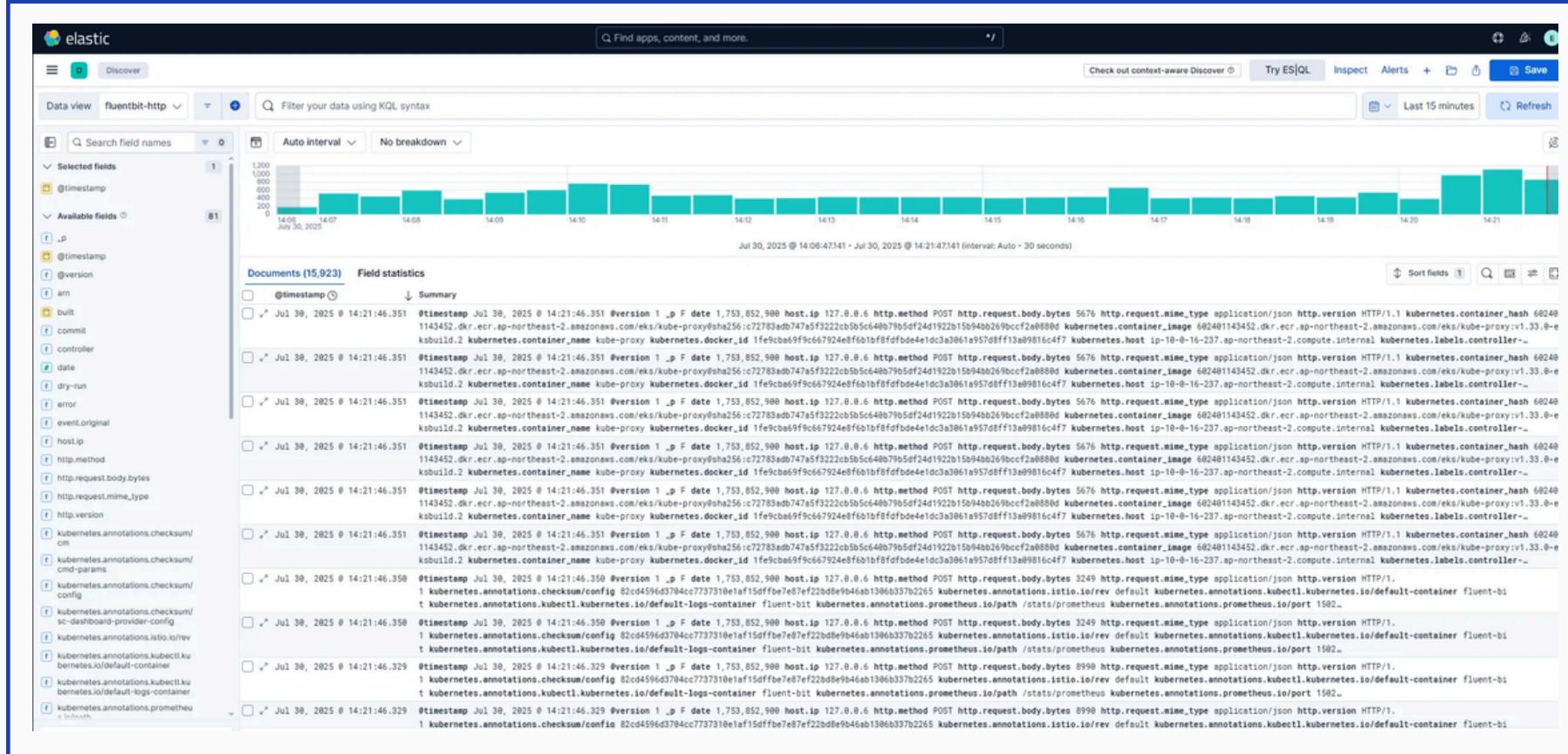
 argocd-notifications-bot 웹 오전 1:32
✓ 애플리케이션 mapzip-dev-service-schedule 이 성공적으로 배포되었습니다.

mapzip-dev-service-schedule - 정상 상태
동기화 상태 Synced
Repository  https://github.com/CLD3rd-Team4/Infra

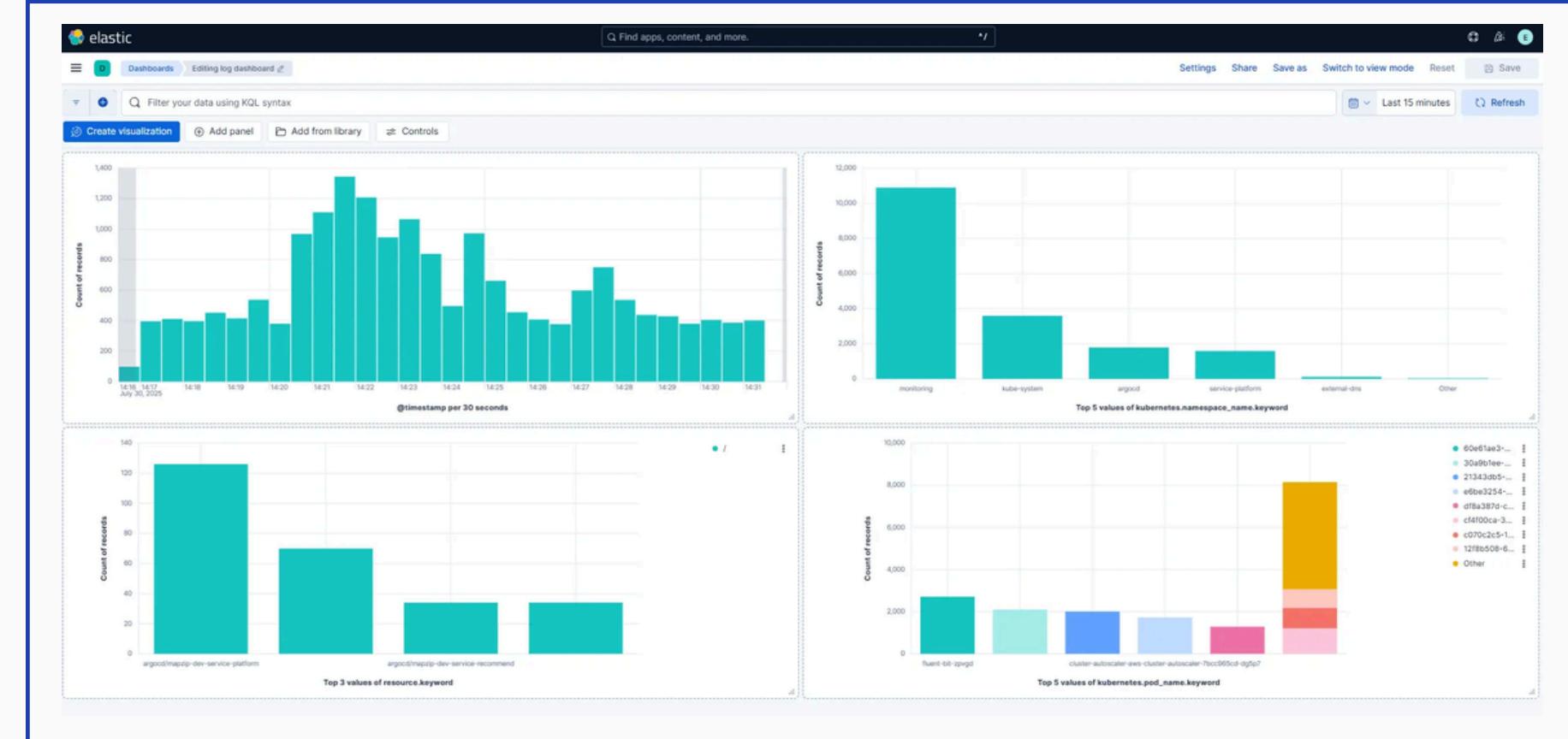
리비전
6823b1f509a49d6569c1a667c0f705ba55
68db92

ArgoCD 슬랙 채널별 알람

- 백엔드 서버별 application의 트리거 구독 설정을 각 서버별 슬랙 채널로 설정
- 동기화 성공 후 배포가 완료되어 application내의 전체 리소스 상태가 Healthy일 때 알람
- 동기화에 실패했을 때 알람



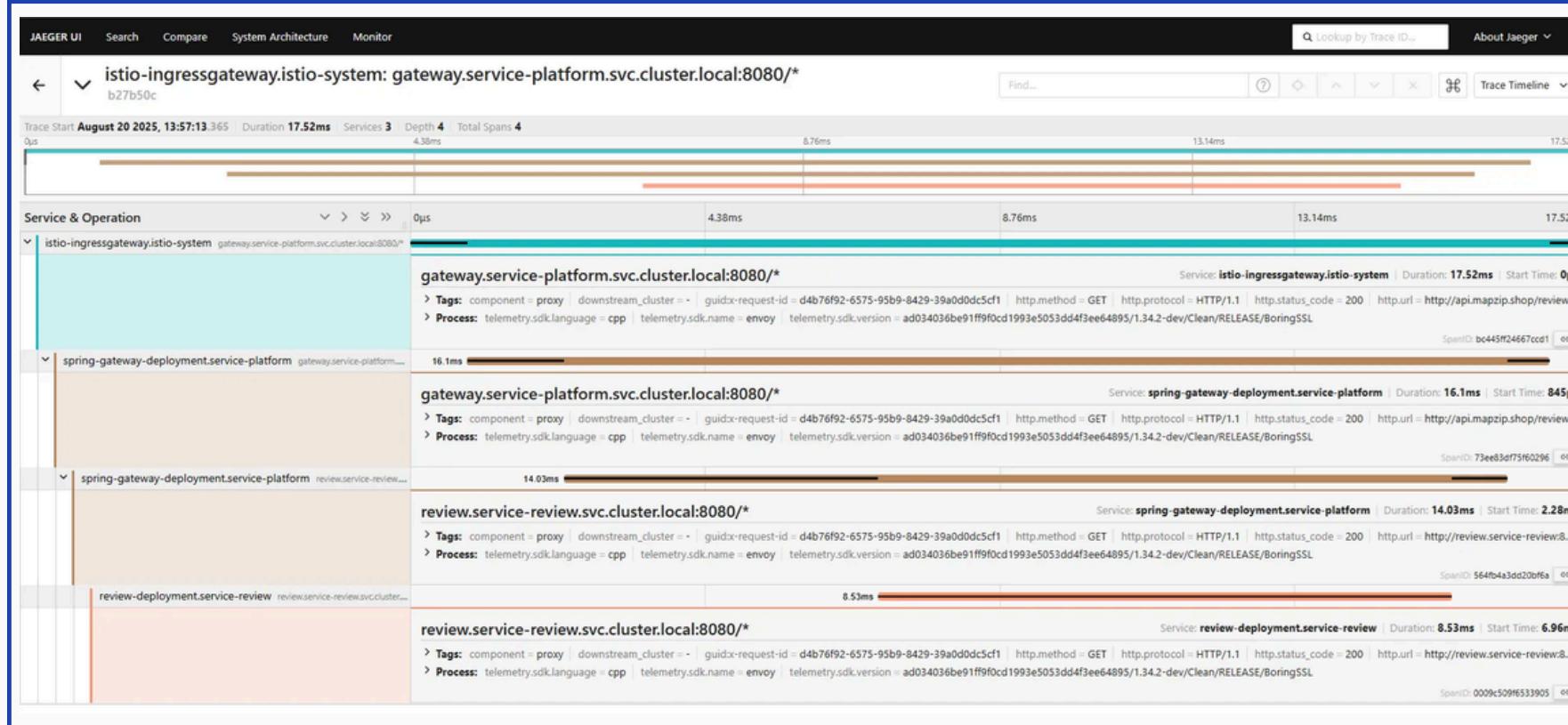
KIBANA LOG DISCOVER



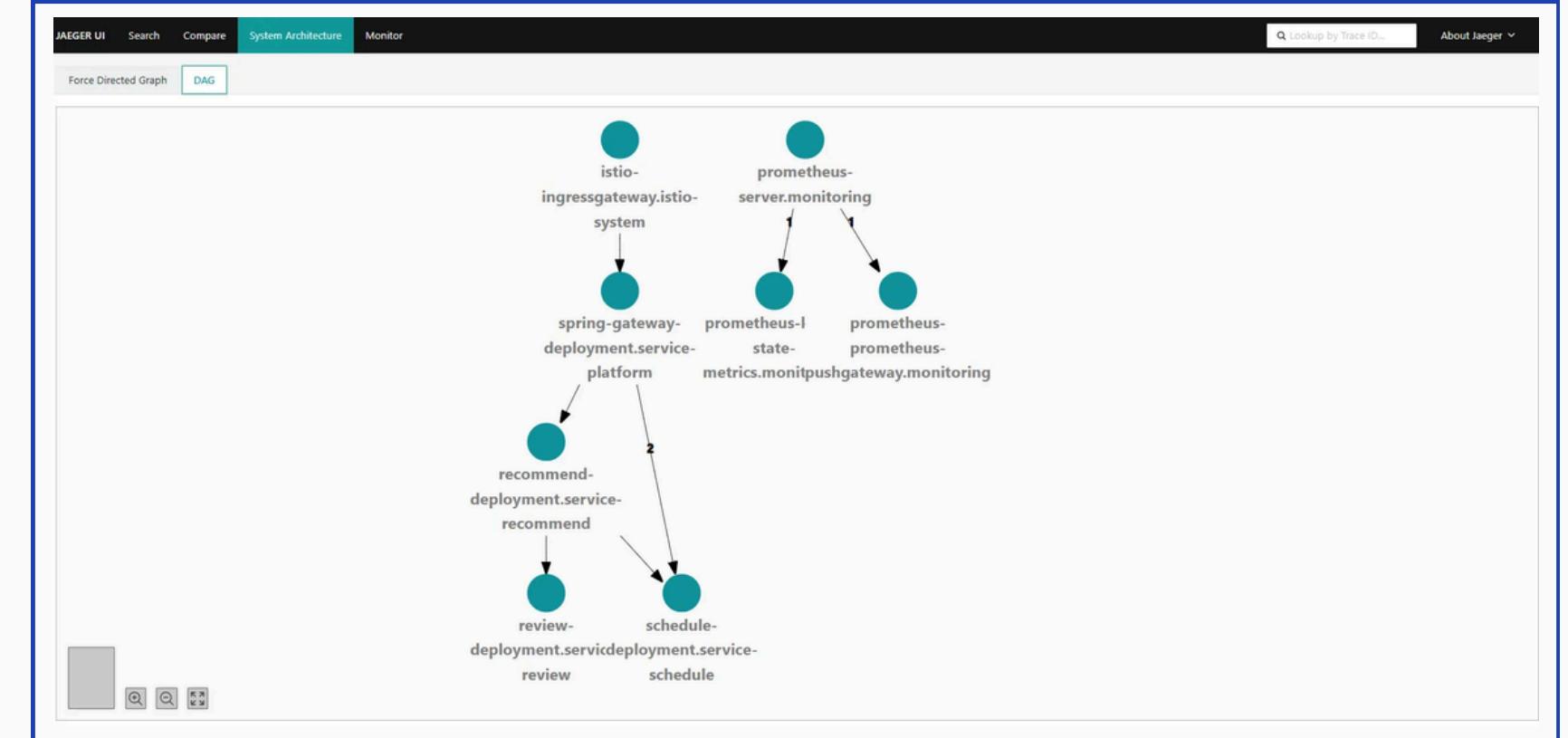
KIBANA LOG DASHBOARD

- EKS의 Fluentbit에서 로그를 수집하여 온프레미스 K8S의 Logstash로 전송 후 ElasticSearch를 거쳐 Kibana로 로그 인덱스 조회 및 시각화

<https://api.mapzip.shop/review/pending> 요청 트래픽



추천 결과 요청 – 분산 경로 다이어그램



분산 추적

- 애플리케이션 Pod에 붙은 Envoy 사이드카 프록시가 모든 인/아웃바운드 트래픽을 가로채고, 각 요청마다 트레이싱 정보를 담아 전달
- 이때 지정된 샘플링 비율에 따라 요청의 추적 데이터(Span, Trace ID 등)를 수집
- 수집된 트레이스는 Istio → Jaeger로 전송되어, Jaeger UI에서 분산된 요청 흐름을 모니터링
- 지연과 에러가 발생하는 지점을 파악하는데 용이

스케줄 서버

스케줄 목록 조회 (DB 조회)

```
■ TOTAL RESULTS

HTTP
http_req_duration.....: avg=34.4ms min=25.73ms med=33.06ms max=60.23ms p(90)=42.82ms p(95)=48.55ms
{ expected_response:true }....: avg=34.4ms min=25.73ms med=33.06ms max=60.23ms p(90)=42.82ms p(95)=48.55ms
http_req_failed.....: 0.00% 0 out of 100
http_reqs.....: 100 9.552818/s

EXECUTION
iteration_duration.....: avg=1.04s min=1.02s med=1.03s max=1.11s p(90)=1.06s p(95)=1.09s
iterations.....: 100 9.552818/s
vus.....: 10 min=10 max=10
vus_max.....: 10 min=10 max=10

NETWORK
data_received.....: 97 kB 9.2 kB/s
data_sent.....: 11 kB 1.1 kB/s
```

스케줄 수정 (DB 데이터 변경)

```
■ TOTAL RESULTS

HTTP
http_req_duration.....: avg=128.8ms min=37.62ms med=73.29ms max=501.69ms p(90)=294.32ms p(95)=404.28ms
{ expected_response:true }....: avg=128.8ms min=37.62ms med=73.29ms max=501.69ms p(90)=294.32ms p(95)=404.28ms
http_req_failed.....: 0.00% 0 out of 92
http_reqs.....: 92 8.31335/s

EXECUTION
iteration_duration.....: avg=1.13s min=1.03s med=1.07s max=1.56s p(90)=1.34s p(95)=1.46s
iterations.....: 92 8.31335/s
vus.....: 2 min=2 max=10
vus_max.....: 10 min=10 max=10

NETWORK
data_received.....: 180 kB 16 kB/s
data_sent.....: 84 kB 7.6 kB/s
```

리뷰 서버

리뷰 상세 조회 (캐시 조회)

```
■ TOTAL RESULTS

checks_total.....: 80      7.28648/s
checks_succeeded.: 100.00% 80 out of 80
checks_failed....: 0.00%   0 out of 80

✓ request is successful

HTTP
http_req_duration.....: avg=344.65ms min=27.98ms med=39.35ms max=2.52s p(90)=2.4s p(95)=2.49s
{ expected_response:true }....: avg=344.65ms min=27.98ms med=39.35ms max=2.52s p(90)=2.4s p(95)=2.49s
http_req_failed.....: 0.00% 0 out of 80
http_reqs.....: 80      7.28648/s

EXECUTION
iteration_duration.....: avg=1.35s  min=1.02s  med=1.04s  max=3.58s p(90)=3.47s p(95)=3.56s
iterations.....: 80      7.28648/s
vus.....: 4      min=4      max=10
vus_max.....: 10     min=10     max=10

NETWORK
data_received.....: 72 kB 6.5 kB/s
data_sent.....: 11 kB 984 B/s
```

처음 조회 시 캐시 미스로 DB에서 불러오는데 시간이 소요됨

```
■ TOTAL RESULTS

checks_total.....: 100     9.509784/s
checks_succeeded.: 100.00% 100 out of 100
checks_failed....: 0.00%   0 out of 100

✓ request is successful

HTTP
http_req_duration.....: avg=39.42ms min=27.79ms med=36.59ms max=80.64ms p(90)=52.17ms p(95)=56.76ms
{ expected_response:true }....: avg=39.42ms min=27.79ms med=36.59ms max=80.64ms p(90)=52.17ms p(95)=56.76ms
http_req_failed.....: 0.00% 0 out of 100
http_reqs.....: 100     9.509784/s

EXECUTION
iteration_duration.....: avg=1.04s  min=1.02s  med=1.04s  max=1.1s  p(90)=1.08s  p(95)=1.09s
iterations.....: 100     9.509784/s
vus.....: 10     min=10     max=10
vus_max.....: 10     min=10     max=10

NETWORK
data_received.....: 79 kB 7.5 kB/s
data_sent.....: 12 kB 1.1 kB/s
```

동일 조건에서 한번 더 요청 시 확연히 감소

리뷰 서버

리뷰 작성 (DynamoDB 쓰기, S3 업로드)

```

TOTAL RESULTS
checks_total.....: 400      33.598138/s
checks_succeeded.: 100.00% 400 out of 400
checks_failed....: 0.00%   0 out of 400

✓ 리뷰 생성 성공
✓ 리뷰 생성 응답시간
✓ 리뷰 ID 반환됨
✓ 응답 JSON 유효

HTTP
http_req_duration.....: avg=163.19ms min=56.02ms med=77.06ms max=1.06s p(90)=175.45ms p(95)=1.05s
  { expected_response:true }....: avg=163.19ms min=56.02ms med=77.06ms max=1.06s p(90)=175.45ms p(95)=1.05s
http_req_failed.....: 0.00% 0 out of 100
http_reqs.....: 100     8.399534/s

EXECUTION
iteration_duration.....: avg=1.17s  min=1.05s  med=1.08s  max=2.13s p(90)=1.18s  p(95)=2.12s
iterations.....: 100     8.399534/s
vus.....: 10      min=10      max=10
vus_max.....: 10      min=10      max=10

NETWORK
data_received.....: 130 kB 11 kB/s
data_sent.....: 138 kB 12 kB/s

```

처음 요청 시 콜드 스타트 지연으로 느린 응답

```

TOTAL RESULTS
checks_total.....: 400      35.19955/s
checks_succeeded.: 100.00% 400 out of 400
checks_failed....: 0.00%   0 out of 400

✓ 리뷰 생성 성공
✓ 리뷰 생성 응답시간
✓ 리뷰 ID 반환됨
✓ 응답 JSON 유효

HTTP
http_req_duration.....: avg=102.15ms min=55.93ms med=70.05ms max=333.68ms p(90)=237.78ms p(95)=254.12ms
  { expected_response:true }....: avg=102.15ms min=55.93ms med=70.05ms max=333.68ms p(90)=237.78ms p(95)=254.12ms
http_req_failed.....: 0.00% 0 out of 100
http_reqs.....: 100     8.799888/s

EXECUTION
iteration_duration.....: avg=1.11s  min=1.05s  med=1.07s  max=1.39s  p(90)=1.29s  p(95)=1.3s
iterations.....: 100     8.799888/s
vus.....: 9       min=9       max=10
vus_max.....: 10      min=10      max=10

NETWORK
data_received.....: 130 kB 11 kB/s
data_sent.....: 138 kB 12 kB/s

```

같은 조건으로 다시 테스트하면 응답이 빨라짐

리뷰 서버

영수증 OCR 검증

```
| TOTAL RESULTS
checks_total.....: 20      10.097847/s
checks_succeeded.: 100.00% 20 out of 20
checks_failed....: 0.00%   0 out of 20

✓ request is successful
✓ OCR 검증 성공

HTTP
http_req_duration.....: avg=1.02s min=115.68ms med=1.02s max=1.92s p(90)=1.91s p(95)=1.92s
  { expected_response:true }...: avg=1.02s min=115.68ms med=1.02s max=1.92s p(90)=1.91s p(95)=1.92s
http_req_failed.....: 0.00% 0 out of 10
http_reqs.....: 10      5.048924/s

EXECUTION
iteration_duration....: avg=1.08s min=175.06ms med=1.08s max=1.97s p(90)=1.97s p(95)=1.97s
iterations.....: 10      5.048924/s
vus.....: 5      min=5      max=5
vus_max.....: 10      min=10     max=10

NETWORK
data_received.....: 64 kB 33 kB/s
data_sent.....: 1.3 MB 674 kB/s
```

평균적으로 1초 소요

추천 서버

추천 결과 호출 (지도 API, AI API 사용)

```

TOTAL RESULTS

checks_total.....: 10      0.107158/s
checks_succeeded.: 100.00% 10 out of 10
checks_failed....: 0.00%   0 out of 10

✓ initial request is successful
✓ status is OK

HTTP
http_req_duration.....: avg=45.64ms min=21.65ms med=26.96ms max=777.81ms p(90)=38.98ms p(95)=65.06ms
  { expected_response:true }....: avg=45.64ms min=21.65ms med=26.96ms max=777.81ms p(90)=38.98ms p(95)=65.06ms
http_req_failed.....: 0.00%  0 out of 264
http_reqs.....: 264     2.828966/s

EXECUTION
iteration_duration....: avg=53.49s  min=26.05s  med=50.6s   max=1m33s    p(90)=1m24s   p(95)=1m28s
iterations.....: 5       0.053579/s
vus.....: 1       min=1       max=5
vus_max.....: 5       min=5       max=5

NETWORK
data_received.....: 158 kB 1.7 kB/s
data_sent.....: 16 kB   175 B/s

```

서울 출발 – 대전 경유 – 부산 도착, 식사 개수 3개인 스케줄은 평균 50초 소요

```

TOTAL RESULTS

checks_total.....: 10      0.255325/s
checks_succeeded.: 100.00% 10 out of 10
checks_failed....: 0.00%   0 out of 10

✓ initial request is successful
✓ status is OK

HTTP
http_req_duration.....: avg=28.39ms min=20.47ms med=25.56ms max=98.27ms p(90)=39.08ms p(95)=46.49ms
  { expected_response:true }....: avg=28.39ms min=20.47ms med=25.56ms max=98.27ms p(90)=39.08ms p(95)=46.49ms
http_req_failed.....: 0.00%  0 out of 107
http_reqs.....: 107     2.731978/s

EXECUTION
iteration_duration....: avg=20.11s  min=8.41s   med=16.6s   max=39.16s  p(90)=34.29s  p(95)=36.73s
iterations.....: 5       0.127663/s
vus.....: 1       min=1       max=5
vus_max.....: 5       min=5       max=5

NETWORK
data_received.....: 76 kB  1.9 kB/s
data_sent.....: 9.1 kB  231 B/s

```

고양시 대화역 출발 – 에버랜드 도착, 식사 개수 1개인 스케줄은 평균 20초 소요

스케줄 서버

스케줄 목록 조회 (DB 조회)

```
| TOTAL RESULTS

checks_total.....: 8637    106.634463/s
checks_succeeded.: 100.00% 8637 out of 8637
checks_failed....: 0.00%   0 out of 8637

✓ status is 200

HTTP
http_req_duration.....: avg=1.76s min=22.17ms med=874.76ms max=36.01s p(90)=4.74s p(95)=5.94s
{ expected_response:true }...: avg=1.76s min=22.17ms med=874.76ms max=36.01s p(90)=4.74s p(95)=5.94s
http_req_failed.....: 0.00% 0 out of 8637
http_reqs.....: 8637    106.634463/s

EXECUTION
iteration_duration.....: avg=5.74s min=1.02s med=2.68s max=46.5s p(90)=14.93s p(95)=22.55s
iterations.....: 8637    106.634463/s
vus.....: 1      min=0      max=967
vus_max.....: 1000   min=109   max=1000

NETWORK
data_received.....: 9.3 MB 115 kB/s
data_sent.....: 1.1 MB 13 kB/s
```

최대 VU 1000명까지 에러없이 조회 확인했으며 그 이상은 로컬 k6 환경에서 테스트 불가

스케줄 수정 (DB 데이터 변경)

```
| TOTAL RESULTS

HTTP
http_req_duration.....: avg=3.3s min=24.86ms med=984ms max=46.97s p(90)=8.54s p(95)=10.9s
{ expected_response:true }...: avg=3.3s min=24.86ms med=984ms max=46.97s p(90)=8.54s p(95)=10.9s
http_req_failed.....: 0.00% 0 out of 4932
http_reqs.....: 4932    71.606996/s

EXECUTION
iteration_duration.....: avg=6.52s min=1.02s med=2.95s max=54.42s p(90)=16.57s p(95)=19.15s
iterations.....: 4783    69.443686/s
vus.....: 337    min=0      max=728
vus_max.....: 800    min=89     max=800

NETWORK
data_received.....: 11 MB 155 kB/s
data_sent.....: 4.8 MB 69 kB/s
```

비관적 락을 걸어 최대 VU 800명까지 에러없이 수정함을 확인했으나 응답 속도는 VU가 늘수록 증가

인증 서버

CORS 자격증명 설정 이슈

- **문제상황:**

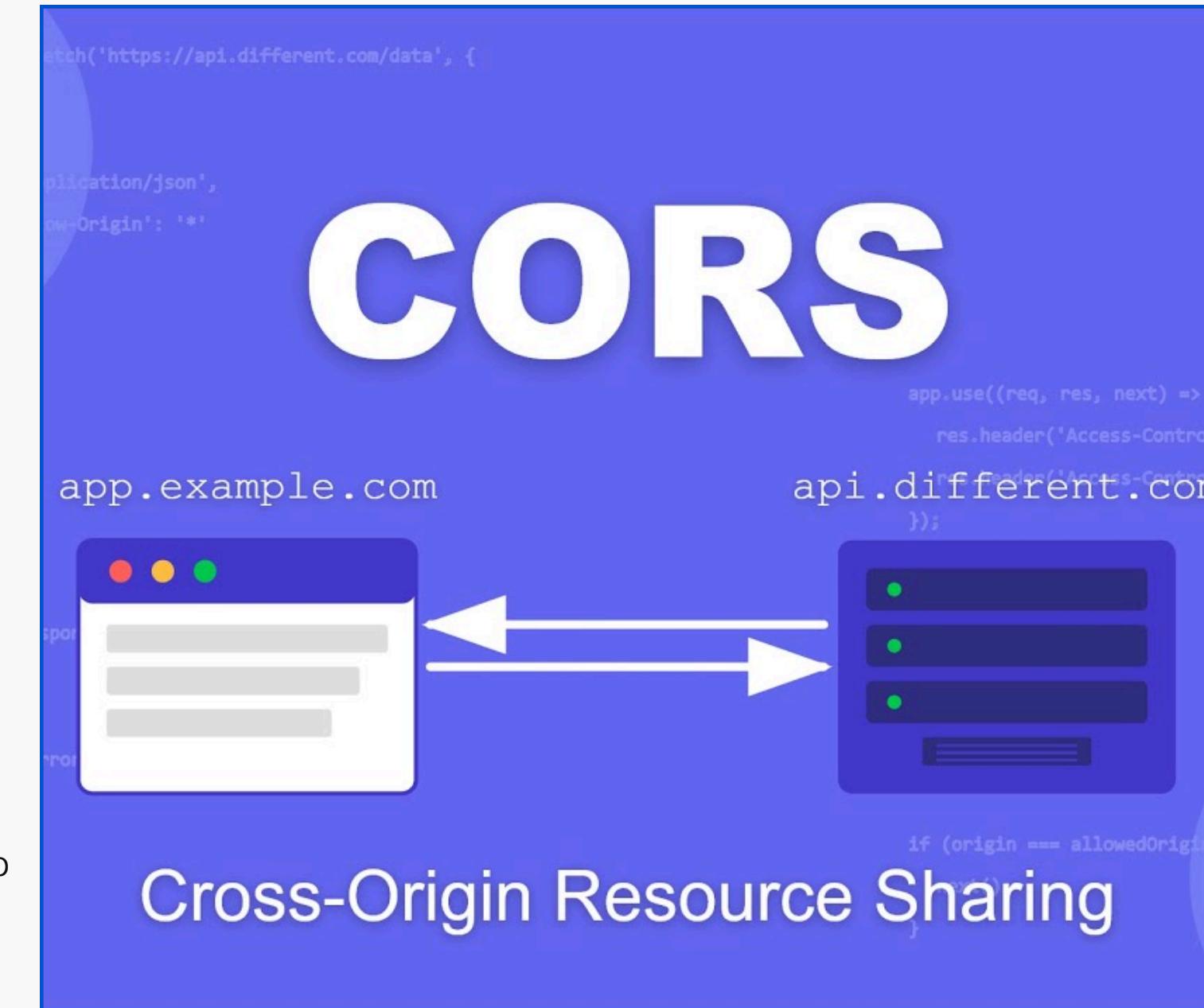
- 서브도메인 간 요청에서 인증 쿠키가 전송/저장되지 않고, 프리플라이트가 막혀 로그인 후 API 호출이 401로 이어짐

- **검토사항:**

- Origin 화이트리스트 부정확 (와일드카드와 credentials 충돌 가능)
- 허용 메서드/헤더 미지정, OPTIONS 미허용으로 프리플라이트 실패

- **해결방안:**

- 정확한 Origin을 명시하고 credentials 허용, 허용 메서드/헤더 지정, OPTIONS 전역 허용
- 운영 쿠키는 SameSite=None, Secure(활성), Domain=.mapzip.shop 으로 통일



추천 서버

대용량 경로 데이터 처리 이슈

- **문제상황:**

- TMAP API에서 받은 경로 데이터가 예상보다 큰 용량 (300KB)
- 서울-부산 등 장거리 경로에서만 발생

- **검토사항:**

- 카프카 메시지 크기 제한 확인
- 웹클라이언트 버퍼 용량 점검
- 단거리 vs 장거리 응답 데이터 크기 비교 분석

- **해결방안:**

- Kafka message.max.bytes 설정 증대
- 웹클라이언트 버퍼 용량 확장



| 추천 서버 | 지역 | 호출 방식 | 결과 |
|---------------|---|--|--|
| bedrock 호출 이슈 |  로컬 (us-east-1) | Model ID 호출 (anthropic.claude-3-sonnet-20240229-v1:0) |  정상 동작 |
| |  서버 (ap-northeast-2) | Inference Profile ID 호출 필요 (Model ID만 사용 시 403) |  실패 (Model ID)  정상 (Profile ID) |

스케줄 서버

상태 관리 및 캐싱 구조 설계

- **문제상황:**

- 홈화면의 분기를 위해 선택 상태 정보의 저장이 필요
- 프론트엔드와 백엔드 간 상태 동기화 복잡성

- **검토사항:**

- Valkey TTL 기반 임시 데이터 저장 vs 영구 저장소 선택
- 로컬스토리지와 서버 간 상태 동기화 방식
- 데이터 일관성 보장을 위한 캐싱 전략

- **해결방안:**

- 팀 내 논의를 통해 serverless DB 솔루션 적용으로 결정
- Valkey를 1차 캐시로 활용하여 TTL 기반 상태 관리
- 로컬스토리지 생성시간 검증을 통한 클라이언트 측 캐시 무효화

| 구분 | ElastiCache Valkey (Node-based) | ElastiCache Valkey Serverless |
|------------|---------------------------------|-------------------------------|
| 최소 시작 비용 | 인스턴스 크기에 따라 ~\$30-50/월 | \$6/월 |
| 과금 방식 | 시간당 인스턴스 요금 (24/7) | 데이터 저장량 (GB-시간) + ECPU 사용량 |
| 최소 데이터 저장량 | 인스턴스 메모리 전체 과금 | 100MB (Redis 대비 90% 절약) |
| 확장성 | 수동 스케일링 | 자동 스케일링 |
| 유류 시간 비용 | 계속 과금 | 사용량만큼만 과금 |
| 예측 가능성 | 고정 비용 | 사용량에 따라 변동 |
| 운영 복잡성 | 인스턴스 관리 필요 | 관리형 서비스 |
| 성능 | 일관된 성능 | 사용량에 따라 변동 |

Config 서버

암호화 키 관리 이슈

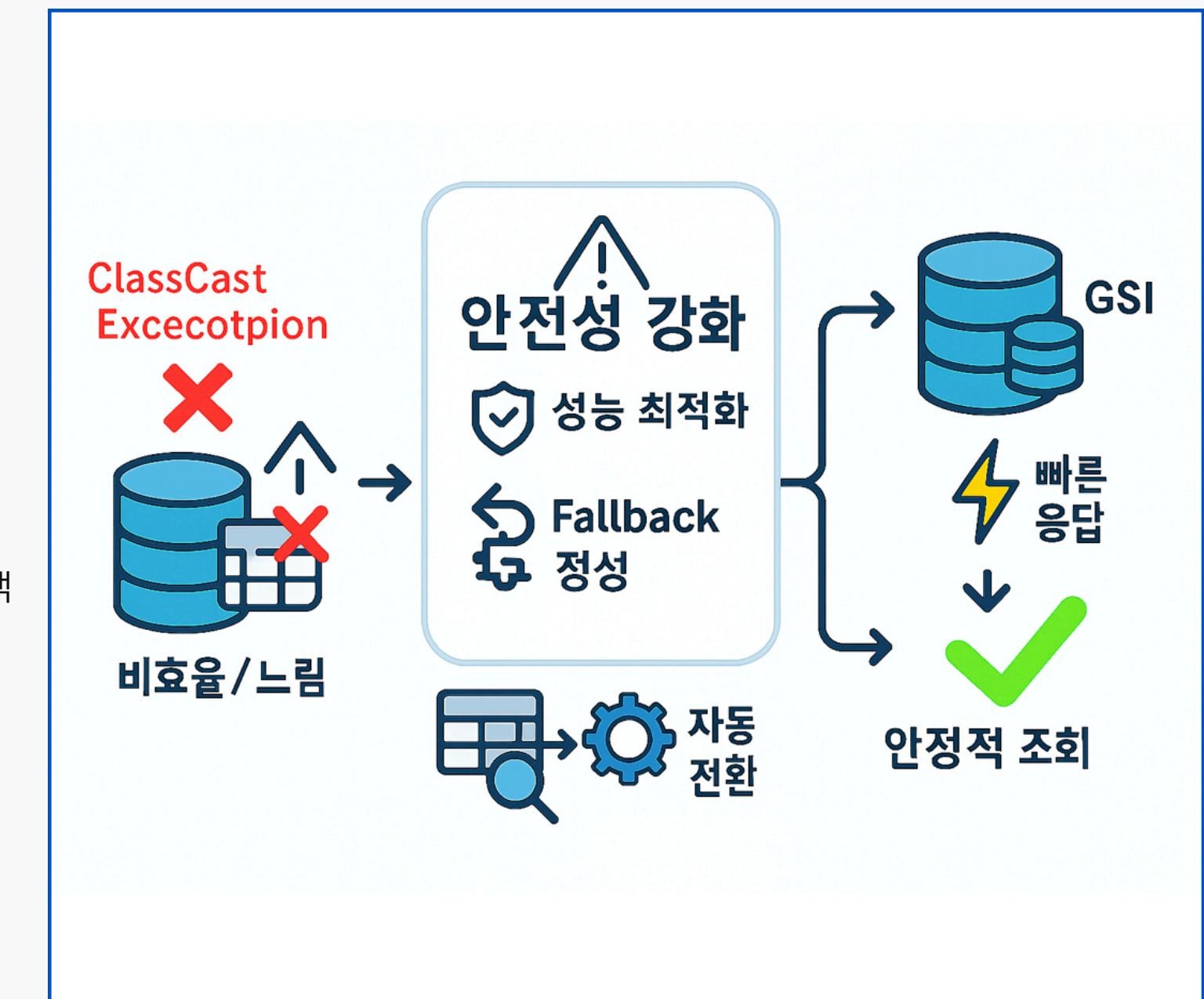
- **문제상황:** MSA 환경에서 Config Server 암호화 키 불일치로 인해 서비스 장애 발생
- **검토사항:**
 - 매일 인프라를 재생성하는 환경에 적합한 Secret 관리 솔루션을 모색
 - AWS Secrets Manager, Git Repository 기반 암호화, AWS SSM Parameter Store를 중심으로 비교·검토
- **해결방안:** AWS SSM Parameter Store 중앙화를 통해 키를 일관되게 관리



리뷰 서버

리뷰 목록 조회 기능 장애 및 최적화

- 문제상황:** DynamoDB 조회 결과의 타입 불일치(LinkedHashMap vs ReviewEntity)로 인해 ClassCastException이 발생, 리뷰 목록 조회 기능 장애
- 검토사항:**
 - ClassCastException을 해결하기 위한 안정적인 탑재 처리 방안
 - 비효율적인 전체 테이블 스캔(Table Scan)을 대체할 성능 최적화 전략
 - 쿼리 최적화 전략 실패 시에도 데이터를 안정적으로 조회할 수 있는 폴백(Fallback) 구조
- 해결방안:** GSI(Global Secondary Index) 우선 조회 및 테이블 스캔 폴백(Fallback) 구조 도입
 - 성능 최적화: userId를 파티션 키로 사용하는 GSI를 우선적으로 쿼리하여 응답 속도 향상.
 - 안정성 강화: GSI 쿼리 실패 또는 결과가 없을 경우, 기존의 테이블 스캔 방식으로 자동 전환하여 데이터 조회 누락 방지.
 - 버그 수정: 스캔 결과의 타입 불일치에 대비하여 instanceof 타입 체크를 추가, ClassCastException 차단.







박시윤

새로운 시도를 다양하게 해보았다. 초기에 본격적으로 개발에 들어가기 전에 기획한 아키텍처가 제대로 동작하는지 확인이 필요했는데 로컬에서 구축한 K8S를 활용해 테스트를 진행하였고 네트워크 설정을 재구성하면서 많은 것을 학습했다. 개발이 진행되면서 아키텍처에 따른 다양한 문제들이 발생했고 이를 해결하기 위해 고민하고 디버깅하는 과정을 거치며 아키텍처 구성에 대한 이해가 높아졌다.



양정모

비전공자로서 인프라와 백엔드, 프론트엔드를 동시에 경험하며 많은 것을 배웠다. 특히 Tmap API 연동 과정에서 당초 복잡한 보간법 로직을 계획했지만, API 응답 구조를 자세히 파악한 후 훨씬 간단하게 해결할 수 있었던 경험이 매우 인상적이었다. 이를 통해 개발 전 충분한 사전 조사와 API 문서 숙지의 중요성을 깨달았고, 예상보다 효율적인 해결책이 있을 수 있다는 점을 배웠다.



서예은

이번 프로젝트는 설계부터 인프라까지 직접 참여한 첫 경험으로, 단순 실습과 달리 다양한 요소를 함께 고려해야 함을 배웠다. 이를 통해 인프라 전반에 대한 이해도가 크게 향상되었다. 또한 MSK, Bedrock 같은 AWS 리소스를 사용하며 로컬에서는 문제없던 부분이 배포 환경에서는 오류로 드러날 수 있다는 점을 경험했고, 배포 후 연동 과정의 중요성을 깊이 깨달았다. 기능을 우선 구현하고 합치는 방식으로 진행하다 보니 설계 부족으로 수정과 확장이 비효율적인 부분도 있었지만, 이를 통해 더 나은 설계와 효율적인 협업 방식의 필요성을 배울 수 있었다.



한동연

MSA 구조 위에 인증서버와 EKS 인프라를 다루며 실제 서비스의 흐름을 처음부터 끝까지 경험했습니다. 쿠키·CORS, 프라이빗 RDS·Valkey 접근 같은 이슈를 추적해 원인을 바로잡고 서비스를 안정화하는 과정에서, 개발·테스트·배포 환경을 최대한 동일하게 맞추고 서비스 상태를 즉시 확인할 수 있도록 준비하는 일이 얼마나 중요한지 깨달았습니다. 낯설었지만 끝까지 해결하며 많이 성장했습니다.



조성민

리뷰 서비스 풀스택 개발 부터 인프라 구축까지 전체 참여를 했습니다. 마이크로서비스 아키텍처에서 DynamoDB GSI 설계, ElastiCache (Valkey) 구성, MSK 메시지 크기 최적화 등을 통해 데이터 처리 시스템을 구축해봤습니다. gRPC+REST 하이브리드 통신, OCR서비스, Next.js 탑재 안전 프론트엔드 구현으로 풀스택 개발 역량을 키웠고, Terraform IaC로 인프라를 코드화하여 관리했습니다. 디버깅과 문제 해결 과정에서 시스템적 사고력이 향상되었으나, 초기 설계 부족으로 인한 비효율성을 경험하며 사전 설계의 중요성을 깨달았습니다.



조성욱

문제 하나를 해결하기 위해 거쳐야 하는 단계들이 점점 체계화되었다. 먼저 kubectl get pods -n 네임스페이스로 상태 확인, kubectl describe pod로 이벤트 로그 체크, kubectl logs로 애플리케이션 로그 분석, 그래도 안 되면 kubectl top nodes로 리소스 상황 파악하는 식으로. 이런 디버깅 플로우가 몸에 배면서, 각 서비스가 가진 부족한 부분들을 체계적으로 분석할 수 있게 되었다.

THANK YOU