



AI와 함께 일하는 시대, 에이전트와 자동화 기술

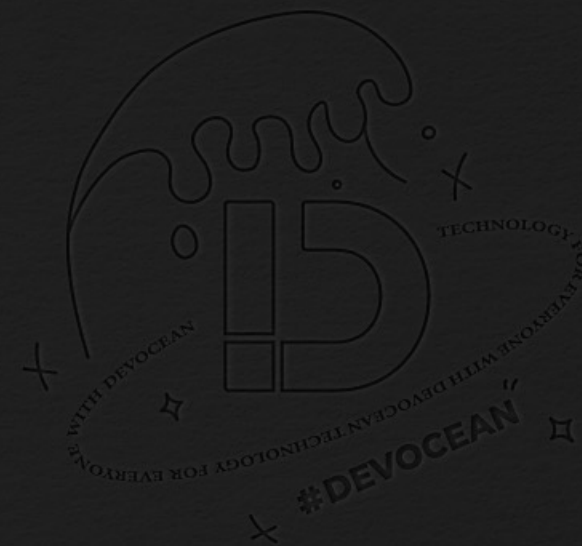


목차

01 환영 인사

02 데보션 소개

03 제9회 테크 데이 행사 소개



SKT AT 서비스개발실 전계범 실장

AWS Telcom 담당 김기남 상무





➤ 대내외 개발자 소통/공유를 위한 DevRel 플랫폼

- 개발자들의 지식과 경험을 소통하고 공유하는 장
- 누구나 참여하는 커뮤니티 채널 운영

➤ Tech 블로그 & 커뮤니티 콘텐츠 제공

- SK개발자들이 작성한 기술 콘텐츠 / 사례 제공
- 최신 기술 트렌드 및 개발자 들 간의 소통 채널 운영

➤ 운영 프로그램

1. 데보션 프로

- SK그룹 전문가 선발 / 운영 / 활동 시 혜택 부여
- Tech 블로그 작성 및 기술 멘토링

2. 데보션 오픈랩

- 내/외부 기술 스터디 모임



03 제9회 테크 데이 소개 - History



➤ 제 1회 (2022-04-29)
- Everything on Kubernetes...



➤ 제 2회 (2022-09-02)
- 기술 커뮤니티의 가치



➤ 제 3회 (2023-02-17)
- KAFKA 서비스 활용 스터디 사례



➤ 제 4회 (2023-04-06)
- Kubernetes 기술 세미나



➤ 제 5회 (2023-10-20)
- 기술 커뮤니티를 통한 성장



➤ 제 6회 (2024-08-08)
- OpenLab 12주간의 성장 스토리



➤ 제 7회 (2024-12-06)
- 함께 만들어가는 도전의 순간들



➤ 제 8회 (2025-07-10)
- AI 인프라의 핵심, 데이터센터를 말하다

03 제9회 테크 데이 소개 - 금일 세션 및 발표자 소개(1)



DEVOCEAN

SK telecom

9TH TECH DAY

AI와 함께 일하는 시대, 에이전트와 자동화 기술

AWS와 함께하는 생성형 AI 기반 코딩부터
에이전트 개발, 배포까지

2025. 09. 01 MON | 13:30 - 17:40

SK텔레콤 SK타워 4층 SUPEX홀



03 제9회 테크 데이 소개 - 금일 세션 및 발표자 소개(2)



세션 소개

13:30 ~ 13:40 **Welcome Address** SKT AT서비스개발실 전계범 실장 / AWS Telco 담당 김기남 상무

13:40 ~ 13:45 **개회사 & 일정소개** 김상기 매니저 | SK Telecom

Session 1 AI Agent 실무 적용과 자동화

13:45 ~ 14:10 **AWS 생성형 AI 최신 기술 동향과 주요 업데이트** Principal SA 이진우 | AWS

14:10 ~ 14:35 **AI 챗봇으로 운영 혁신하기** 정만규 매니저 | SK Telecom

14:35 ~ 15:00 **Strands Agents로 에이전트끼리 일하게 만들기** AIML Specialist SA 이상재 | AWS

15:00 ~ 15:25 **구독 업무 효율화를 위한 Text2SQL 구현** 김성현 매니저 | SK Telecom

15:25 ~ 15:45 **Break**

Session 2 개발 환경 및 DB 최적화와 생산성 혁신

15:45 ~ 16:10 **Agentic Ops 도입 사례 (Feat. AWS Bedrock)** 이종호 매니저 | SK Telecom

16:10 ~ 16:35 **내가 만든 에이전트, Bedrock AgentCore로 빠르게 배포하기** SA 하남봉 | AWS

16:35 ~ 17:00 **T멤버십 사례로 보는 Aurora MySQL DB의 HLL에 대한 이해** 조진구 매니저 | SK Telecom

17:00 ~ 17:25 **새로운 개발 파트너: Q Developer와 Kiro로 여는 개발 자동화의 시대** Principal SA 최우형 | AWS

17:25 ~ 17:40 **폐회사 및 마무리** 김상기 매니저 | SK Telecom

발표자 소개

Session 1 AI Agent 실무 적용과 자동화

AWS AI, 어디까지 왔나요
**고객의 디지털 혁신을
이끄는 AWS Telco
솔루션즈 아키텍트**

Principal SA 이진우 | AWS



챗봇으로 우주 정복하기
구독 플랫폼 엔지니어

정만규 매니저 | SK텔레콤



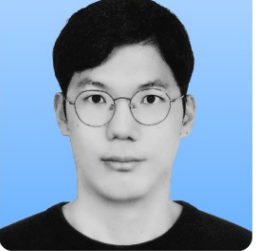
누구나 "쉽게" 만들 수 있는 AI Agent
AI를 통한 문제 해결사

AIML Specialist SA 이상재 | AWS



AI 초급자여도 가능한 업무 효율화
구독 플랫폼 개발/운영

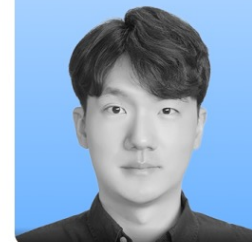
김성현 매니저 | SK텔레콤



Session 2 개발 환경 및 DB 최적화와 생산성 혁신

AI와 함께하는 AWS 클라우드 운영
클라우드 아키텍트

이종호 매니저 | SK텔레콤



에이전트 만들었는데, 어떻게 배포하자?
**통신 인프라 기반 생성형
AI 개발·운영·배포 지원**

SA 하남봉 | AWS



MySQL과 조금 다른 Aurora MySQL
T멤버십 인프라 관리자

조진구 매니저 | SK텔레콤



더 똑똑하게, 덜 힘들게: Amazon Q CLI
**클라우드 생산성의
진화를 이끄는 AWS
솔루션즈 아키텍트**

Principal SA 최우형 | AWS



03 제8회 테크 데이 소개 - 경품추첨안내



세션 종료 후 출석 체크&설문을 해 주신 분들을 대상으로 추첨하여 선물을 드릴 예정입니다.

1 SK VIP 화장품 세트 1개



2 데보션 굿즈 접이식 블루투스 키보드 외 3종 1개



3 데보션 굿즈 초고속PD 충전 배터리 (20000mAh) 2개



AWS AI 최신 기술 동향과 주요 업데이트

Agentic AI와 SW 개발의 미래

이진우

Principal Solutions Architect
AWS Korea | TMEGS & Telco





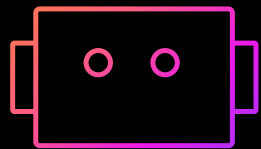
AI Agents

우 리 가 해 왔 던 근 본 적 인 방 식 의 변 화

구 축

배 포

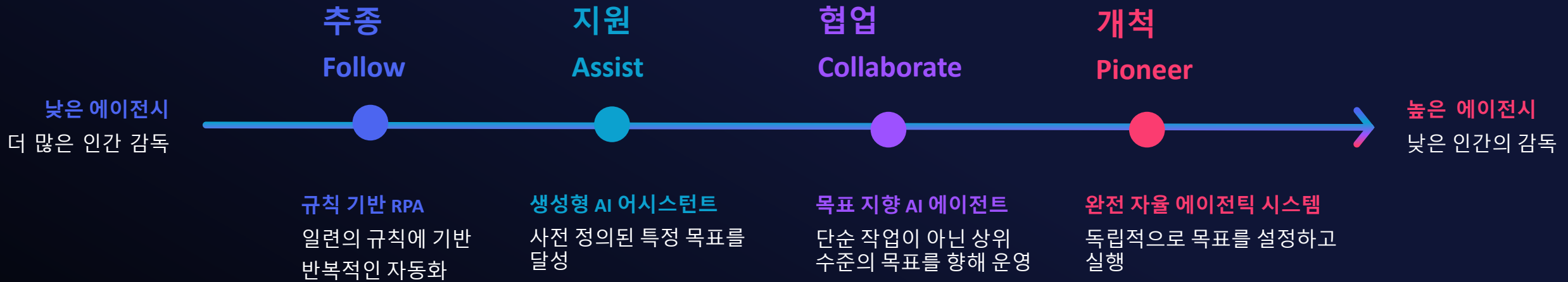
상 호 작 용



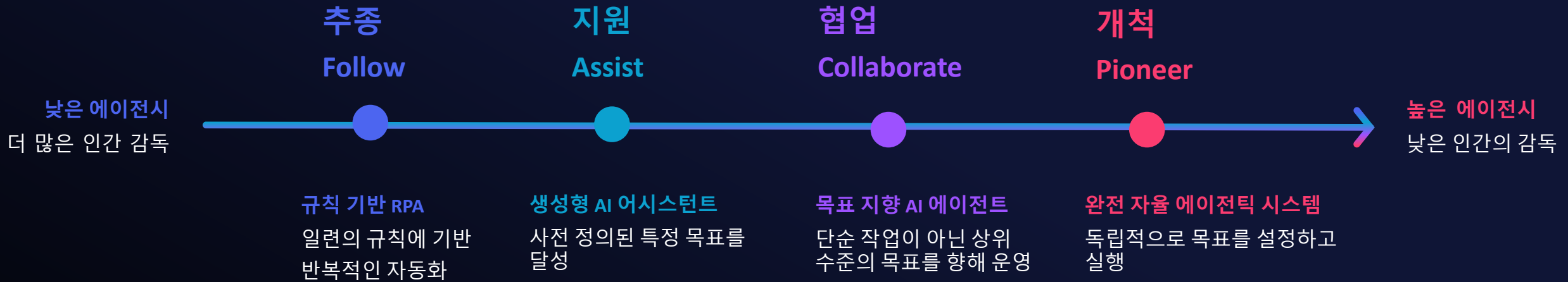
What are AI Agents?

디지털 또는 물리적 환경에서
목표를 달성하기 위해 **추론**하고, **계획**하고,
행동할 수 있는 **자율적** 또는 **반자율적**
소프트웨어 시스템.

AI 에이전트의 성숙도 – 규칙에서 추론까지



AI 에이전트의 성숙도 – 규칙에서 추론까지



기업들은 에이전트에 더욱 집중하고 있습니다

33%

2028년까지 기업용 소프트웨어의
33%가 에이전틱 AI를 포함할
것입니다

Gartner, "Top strategic Technology Trends for 2025," October 2024

15%

2028년까지 의사결정의
15%가 에이전틱 AI를 통해
자율적으로 이루어질 것입니다

Gartner, "Top Strategic Technology Trends: Agentic AI—the Evolution of Experience" February 2025

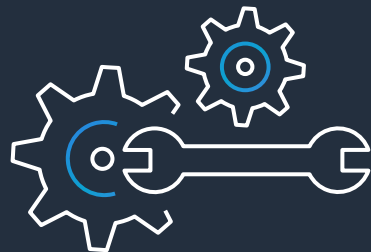
에이전트 AI를 위한 구성요소

모델



멀티모달 파운데이션 모델
에이전트 최적화 모델
특화된 도메인 모델
협업 모델

툴링



에이전트 오케스트레이션
샌드박스 및 테스트
Observability 및 디버깅

프레임워크



에이전트 디자인 패턴
오픈소스 프레임워크
조합 가능한 아키텍처

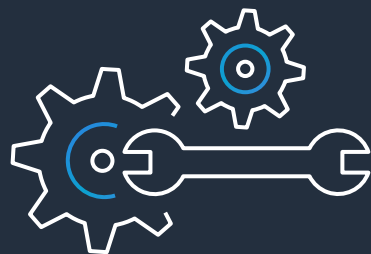
에이전트 AI를 위한 구성요소

모델



멀티모달 파운데이션 모델
에이전트 최적화 모델
특화된 도메인 모델
협업 모델

툴링



에이전트 오케스트레이션
샌드박스 및 테스트
Observability 및 디버깅

프레임워크



에이전트 디자인 패턴
오픈소스 프레임워크
조합 가능한 아키텍처

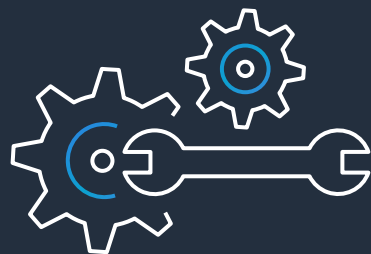
에이전트 AI를 위한 구성요소

모델



멀티모달 파운데이션 모델
에이전트 최적화 모델
특화된 도메인 모델
협업 모델

툴링



에이전트 오케스트레이션
샌드박스 및 테스트
Observability 및 디버깅

프레임워크



에이전트 디자인 패턴
오픈소스 프레임워크
조합 가능한 아키텍처

Amazon Bedrock

AI 애플리케이션과 에이전트를 구축하고 확장하는 가장 쉽고 빠른 방법

다양한 선도적
모델 선택

고객의 데이터를
통한 맞춤화

안전하고 책임감
있는 AI 검사

비용, 지연 시간
및 정확도 최적화

에이전트 구축 및
오케스트레이션

← AWS 보안, 개인정보 보호 및 안정성 내장 →

에이전트 모델의 부상으로, 여전히 모델 선택이 중요합니다

AI21labs

Highly efficient processing
& grounded generation
for long context lengths

JAMBA

amazon

Frontier intelligence
and industry leading
price performance

NOVA

ANTHROPIC

Excels at complex
reasoning, code
generation, and
instruction following

CLAUDE

cohere

Powering efficient,
multilingual AI agents with
advanced search & retrieval

COMMAND
EMBED
RERANK

deepseek

Advanced reasoning models
that solve complex problems
step-by-step

DEEPSEEK-R1

Luma

High-quality video generation
with natural, coherent
motion &
ultra-realistic details

RAY2

Meta

Advanced image and
language reasoning

LLAMA

MISTRAL AI

Specialized expert models
for agentic reasoning
and multimodal tasks

MISTRAL
MIXTRAL
PIXTRAL

OpenAI

**Automate tasks,
enhance creativity,
and solve complex
problems efficiently**

GPT-OSS-120B
GPT-OSS-20B

poolside

Software engineering AI for
large enterprises

Coming soon

stability.ai

Professional-grade images
with creative control,
deployable
at scale

STABLE DIFFUSION
STABLE IMAGE

TwelveLabs

CTRL + F for video data:
unlock the full potential
of enterprise video assets

MARENGO
PEGASUS

WRITER

Purpose-built models for
building and scaling AI
agents across the
enterprise

PALMYRA

확장에는 운영 우수성이 필요합니다

흥분과 잠재력



POC

프로덕션 단계로 가는 길의 도전 과제



성능



확장성



보안

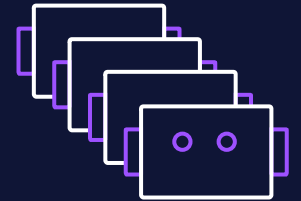


컨텍스트



거버넌스

의미 있는
비즈니스 가치



상용
AI 에이전트



NEW

Amazon Bedrock AgentCore

어떤 프레임워크와 모델을 사용하여도 효과적인 에이전트를
안전하게 대규모로 배포하고 운영할 수 있는 서비스 세트

PREVIEW

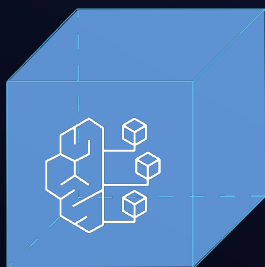
NEW

Amazon Bedrock AgentCore

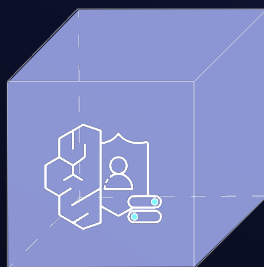
에이전트를 프로덕션에 배포하는 데 필요한 모든 것



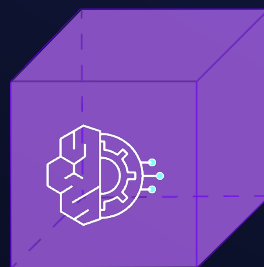
Runtime



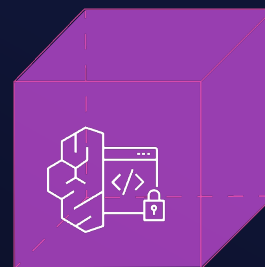
Memory



Identity



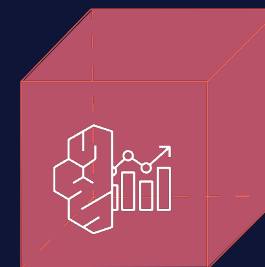
Gateway



Code
Interpreter



Browser Tool



Observability

NEW

PREVIEW

Amazon S3 Vectors

벡터를 대규모로 저장하고 쿼리하는 기능을 기본적으로 지원하는 최초의 클라우드 객체 스토어

Amazon Bedrock Knowledge Bases 및
Amazon OpenSearch Service와 기본적으로
통합

대체 솔루션에 비해 최대 90% 낮은 비용으로
대규모 벡터 임베딩을 저장할 수 있음

NEW

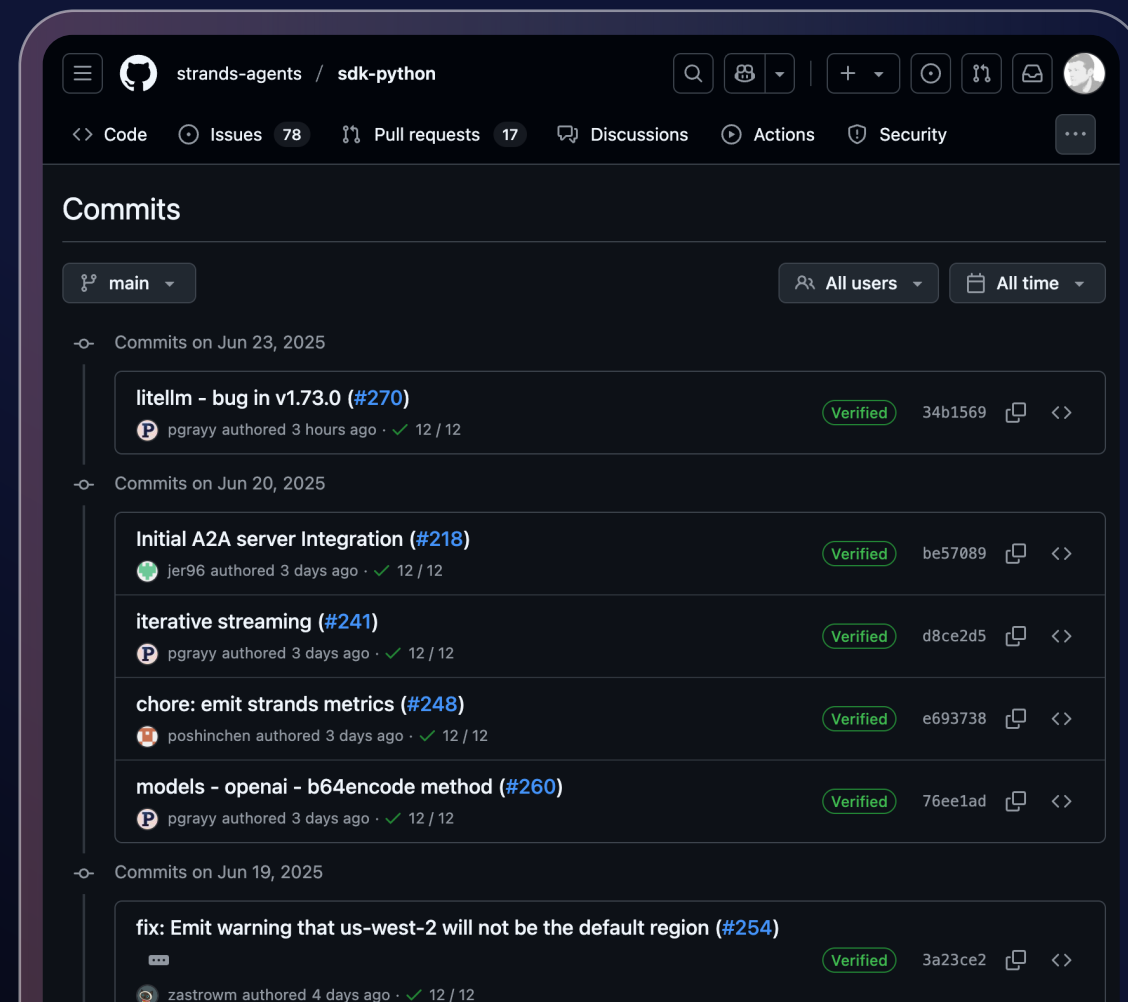


STRANDS AGENTS V1.0 RELEASE

A2A SUPPORT

DATA PERSISTENCE FOR AGENT CONVERSATIONS

MULTI-AGENT WORKFLOWS



“Amazon Q Developer 팀은
Strands Agents SDK를 사용하여
이전에 몇 달이 걸렸던 에이전트를
몇 주 만에 구축하고 출시하고 있습니다”

3 weeks

CLI AGENTIC CHAT

4 weeks

AGENT CODING EXPERIENCE

6 weeks

AGENTS IN GITLAB DUO

< one hour coding time

개발자는 평균적으로 하루에 1시간 미만의 시간을 코드 작성에 사용합니다¹

애플리케이션 실행 및
유지보수에 소요되는 개발자 시간

73%

혁신과 변화를 위한
개발자 시간

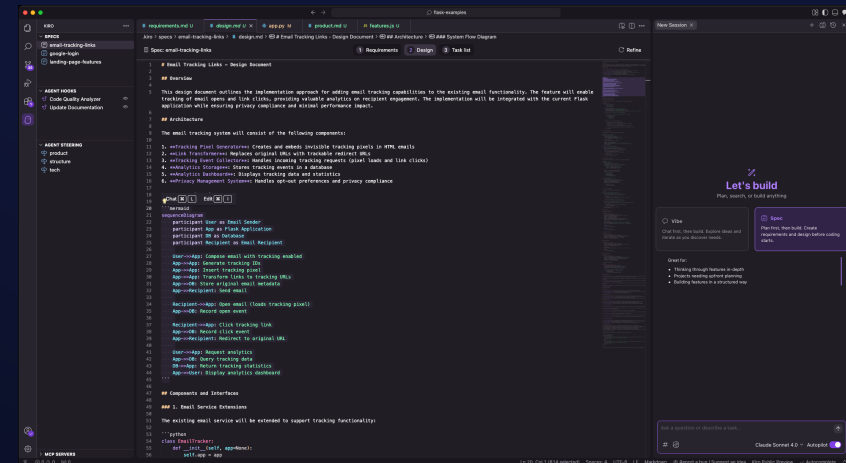
27%

¹ Software.com, "Global Code Time Report," January 2022

Amazon Q Developer



Amazon Q는 처음부터 보안과 개인 정보 보호를 염두에 두고 구축되어 조직이 생성형 AI를 안전하게 사용할 수 있도록 돕습니다.



최고의 작업을 수행하는 데 도움이 되는 에이전트 IDE

에이전트 실행을 위한 최적의 장소

NEW

Amazon Bedrock AgentCore

RUNTIME
IDENTITY
GATEWAY
MEMORY
CODE INTERPRETER
BROWSER TOOL
OBSERVABILITY

NEW

AWS GenAI Innovation Center

+100M INVESTMENT

NEW

AWS Marketplace agents & tools

NEW

AWS AI League

NEW

Amazon S3 Vectors

NEW

Amazon Nova

SAGEMAKER HYPERPOD, NOVA ACT & NOVA CUSTOMIZATIONS

NEW

Strands v1.0

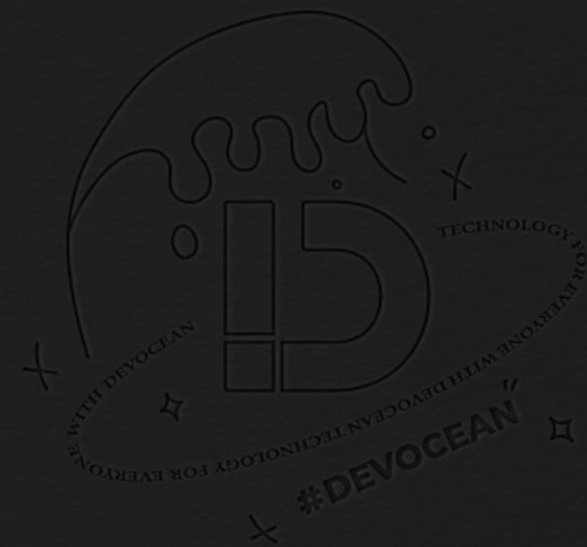
FRAMEWORKS

감사합니다



AI 챗봇으로 운영 혁신하기

SK텔레콤 정민규



목차

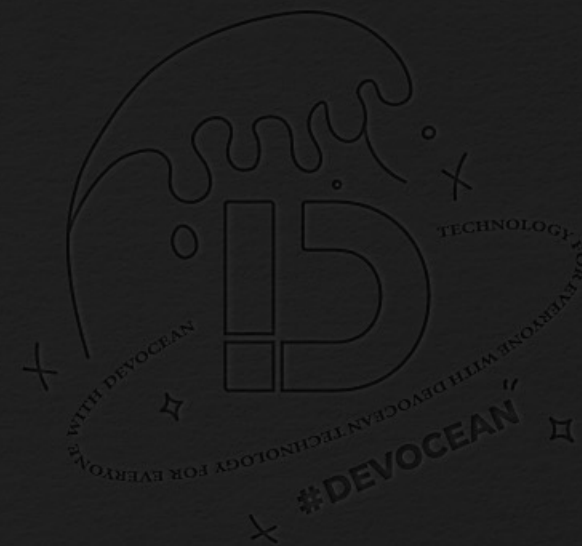
01. T 우주란?

02. 운영의 어려운 점

03. AI 챗봇으로 해결하자!

04. 챗봇 상세 소개

05. 느낀점과 개선 방향



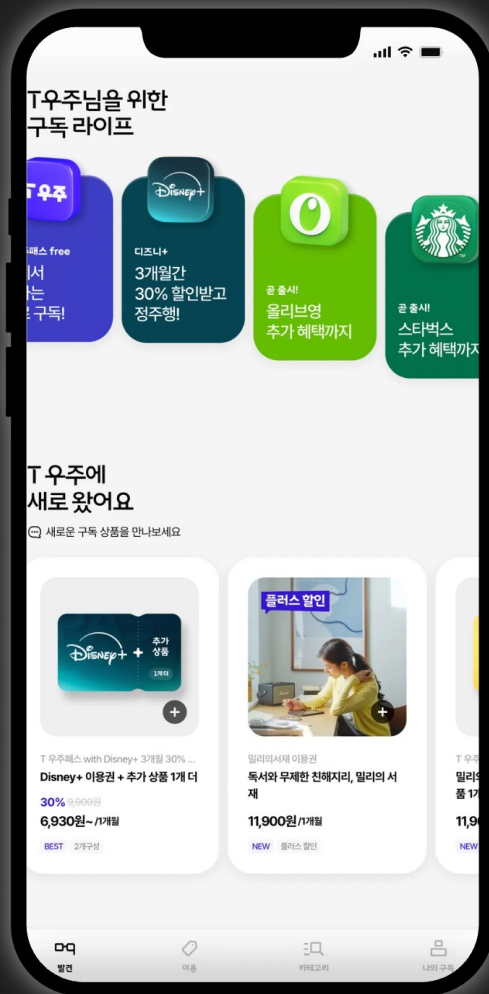
01. T 우주란?



T 우주

01. T 우주란?

AI가 추천하고 할인 혜택을 제공하는 다양한 분야의 맞춤형 구독 플랫폼



패키지 상품

T 우주패스 쇼핑
11번가



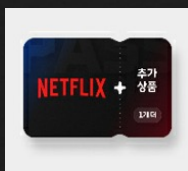
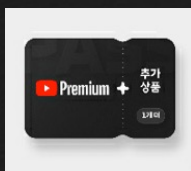
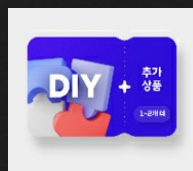
T 우주패스 쇼핑
G마켓



T 우주패스
편의점&카페



추가 혜택으로 다양한
파트너사 혜택 중 택 1~2



단독 상품

미디어 / 생활 / 문화

정기배송

단독으로 이용가능한
제휴사 상품

실물 상품이 배송되는
오픈마켓 상품



FLO 앤데이터



월간과자
마니아팩



배스킨라빈스
혜택팩



과학동아 서적



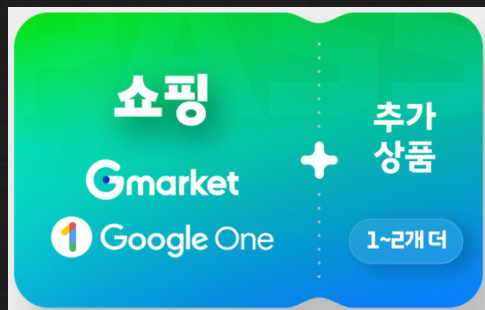
SNOW AI
이용권



간편 과일

01. T 우주란?

패키지 구독 예시



T 우주패스
쇼핑 G마켓
9,900/1개월
(추가 혜택 별 상이)

기본
혜택



G마켓 & 유니버스 클럽
스마트캐시 3,000원, G마켓 &
유니버스 클럽 쿠폰

3,000원 혜택



Google One 100GB
클라우드 스토리지 100GB,
휴대전화 자동 백업 등

2,500원 혜택



추가
혜택



Youtube Premium
Youtube 광고 없이 감상,
오프라인 저장, 백그라운드 재생

19,500원 혜택
(5,000원 금액 추가)

25,000원/월 → 14,900원/월

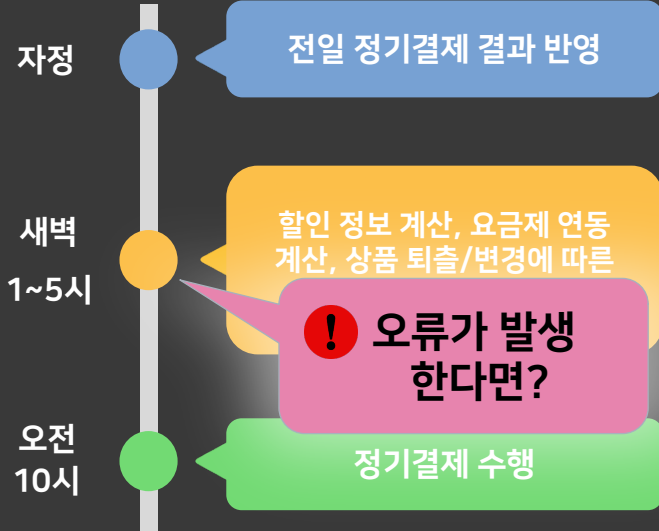
02. 운영의 어려운 점



02. 운영의 어려운 점

매월/매년 정기결제로 수납이 이루어지면서 발생하는 문제점들

매일매일이 정기결제



- 매일 새벽 요금계산, 오전 10시 정기결제
- 고객별로 모두 다른 결제 금액
- 새벽 배치프로그램 오류 시 결제 오류

테스트 데이터 수급



- 구독 후 1달 지나야 정기결제 QA 가능
- 필요한 경우 개발자가 수작업 생성 필요

그 외 잡다한 수작업들



- 시스템 운영에 산재하는 수작업들 존재
- 자동화의 필요성

03. AI 챗봇으로 해결하자!



03. AI 챗봇으로 해결하자!



배치봇

- ✓ 배치프로그램 모니터링
- ✓ AI를 통한 이슈 요약
- ✓ AI와 대화형으로 데이터 조회



QA봇

- ✓ 정기결제일 변경
- ✓ 테스트 계정 수급
- ✓ AI로 테스트 케이스 일괄 생성



정산봇

- ✓ 정산에서 발생하는 파일 수작업 자동화
- ✓ 효율적인 검증 작업

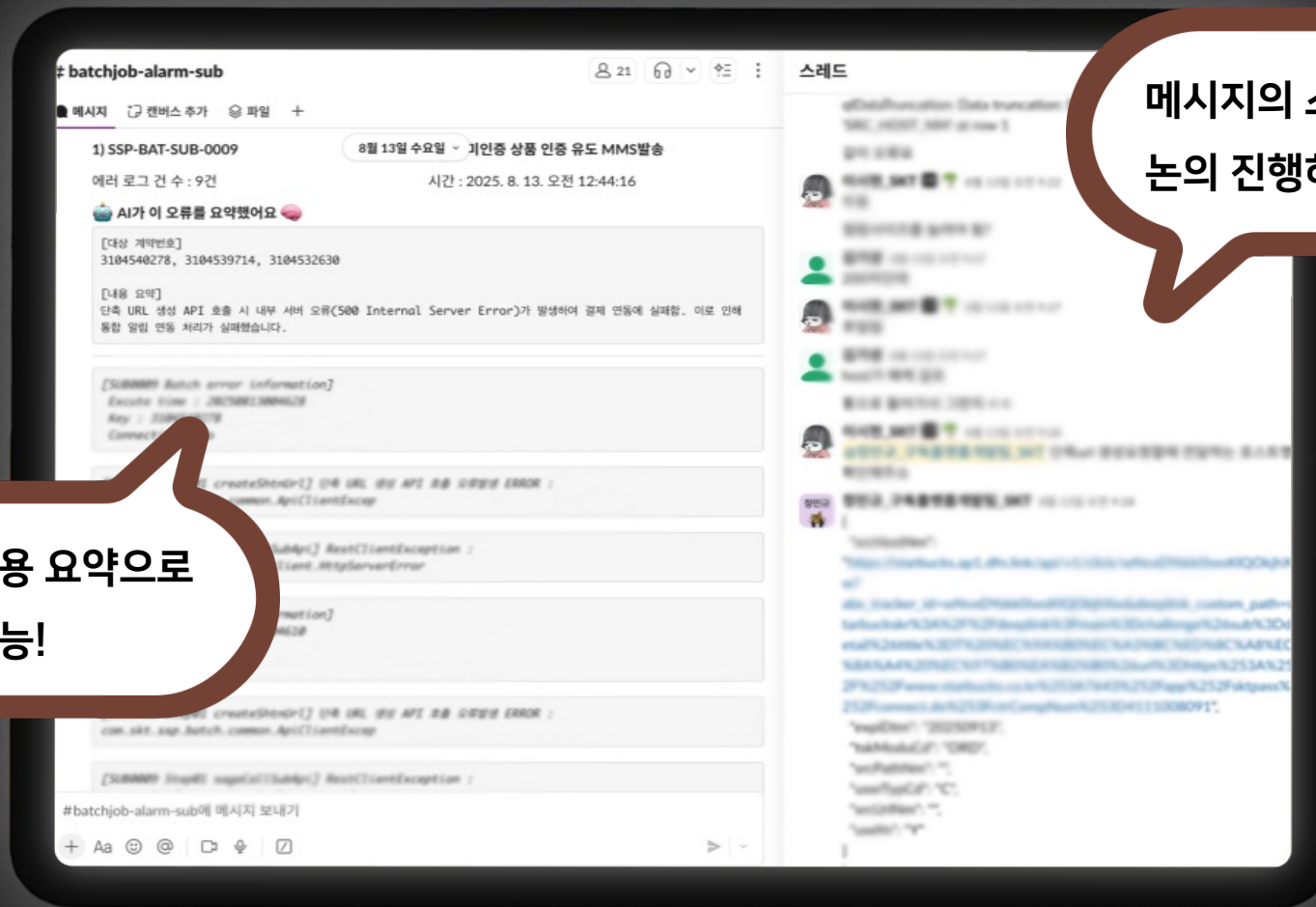
TECHNOLOGY FOR EVERYONE WITH DE
CEAN

03. AI 챗봇으로 해결하자! - Why 챗봇?



03. AI 챗봇으로 해결하자! - Why 챗봇?

예시) 모니터링 결과 메시지의 스레드에서 커뮤니케이션 진행



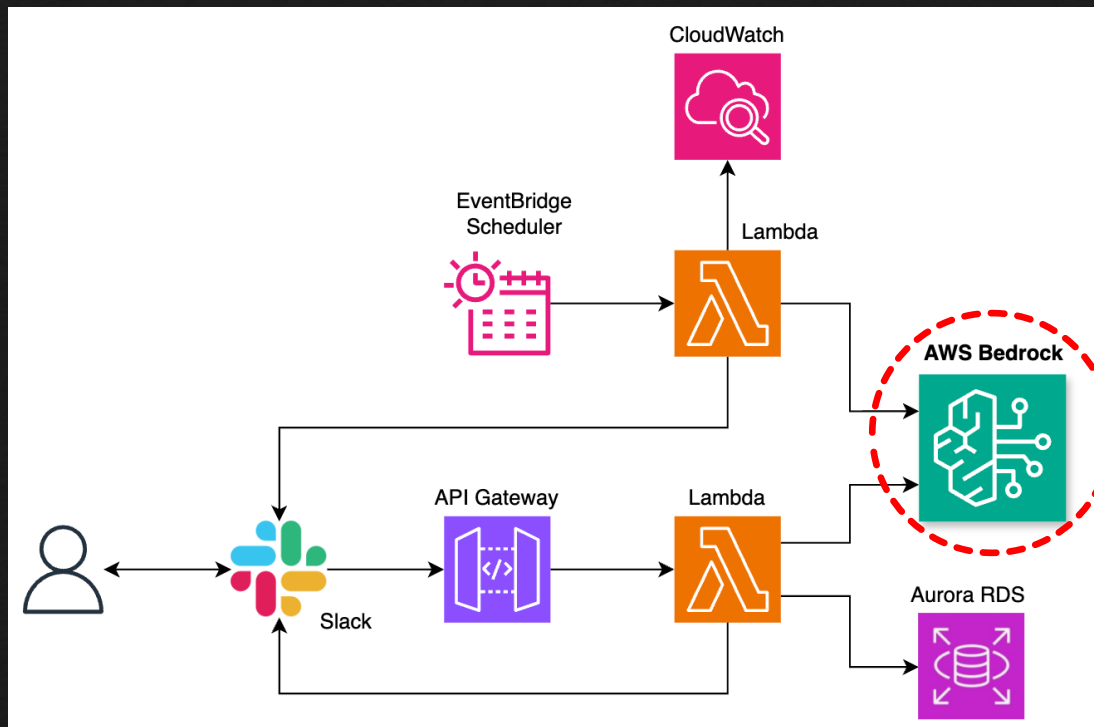
메시지의 스레드에서 개발자
논의 진행하여 이슈 해결!

배치 오류 내용 요약으로
빠른 확인 가능!



03. AI 챗봇으로 해결하자! - AI기능은 어떻게 사용?

AWS Bedrock 을 활용하여 간단하게 구현!



[배치봇 서비스 구성의 예시]

```
const { BedrockRuntimeClient, InvokeModelCommand } = require("@aws-sdk/client-bedrock-runtime");
const client = new BedrockRuntimeClient({region:"ap-northeast-2"});

const prompt = `프롬프트 작성...`;

const input = {
  "modelId": "anthropic.claude-3-5-sonnet-20240620-v1:0",
  "contentType": "application/json",
  "accept": "application/json",
  "body": JSON.stringify({
    "anthropic_version": "bedrock-2023-05-31",
    "max_tokens": 2000,
    "top_k": 250,
    "messages": [{ role: "user", content: [{ type: "text", text: prompt }] }]
  })
};

const command = new InvokeModelCommand(input);
const resp = await client.send(command);
```

04. 챗봇 상세 소개



04. 챗봇 상세 소개 - 배치봇



나는 **배치봇!**
배치프로그램
모니터링과 데이터
조회를 담당해!



04. 챗봇 상세 소개 - 배치봇



기존의 배치프로그램 모니터링 방식



04. 챗봇 상세 소개 - 배치봇



기존의 배치프로그램 모니터링 방식



04. 챗봇 상세 소개 - 배치봇



배치봇을 통한 자동 모니터링

모니터링 결과 자동 알림

- 능동적으로 모니터링 결과를 푸시하여 적시 조치
- 배치 프로그램 별 이슈 리스트업 하여 한눈에 파악

AI를 통한 이슈 요약

- 로그를 열지 않고도 이슈 Key 값 확인
- 내용 요약하여 문제 원인 파악에 도움

커뮤니케이션

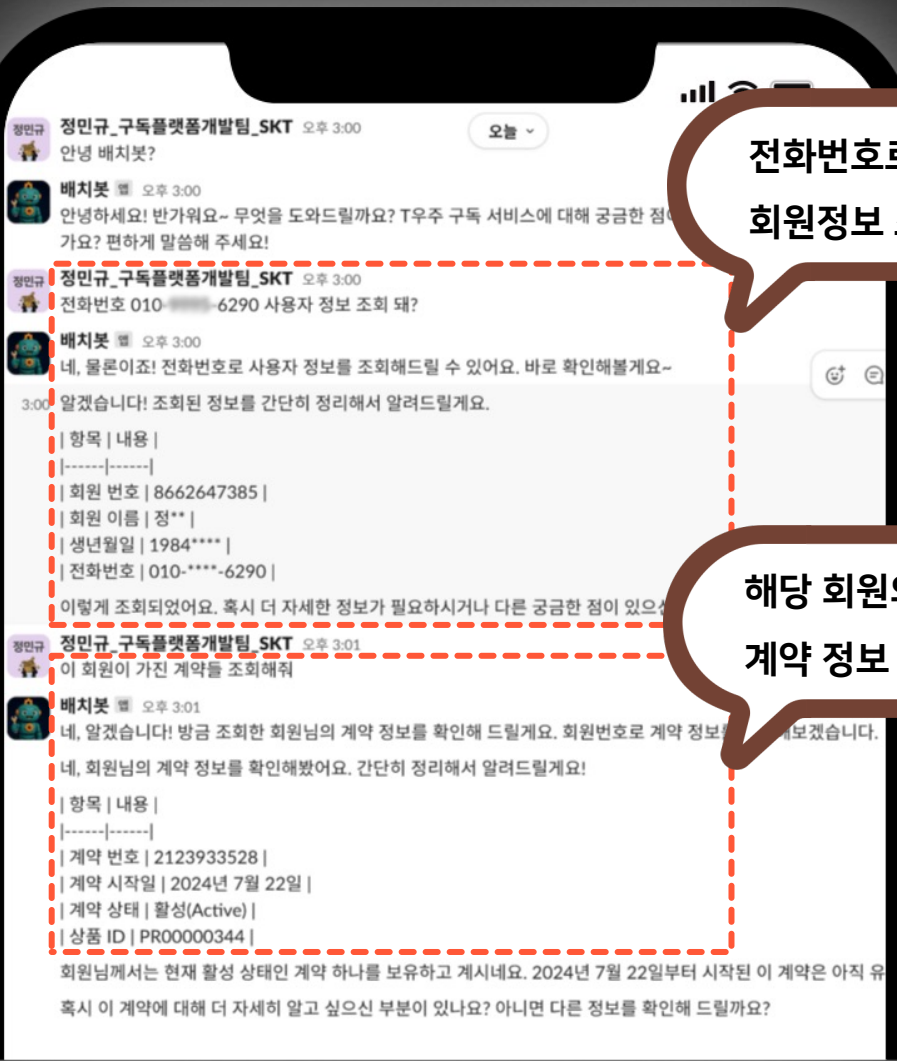
- 모니터링 결과에 스레드를 열어서 개발팀 논의



04. 챗봇 상세 소개 - 배치봇



대화형 구독 에이전트 (데이터 조회)



전화번호로 구독
회원정보 조회하기

해당 회원의 구독
계약 정보 조회하기

DB 데이터 조회 결과 공유

- VOC 대응 시 고객 구독 정보를 챗봇에서 쉽게 조회
- Text2SQL 등으로 고도화 가능

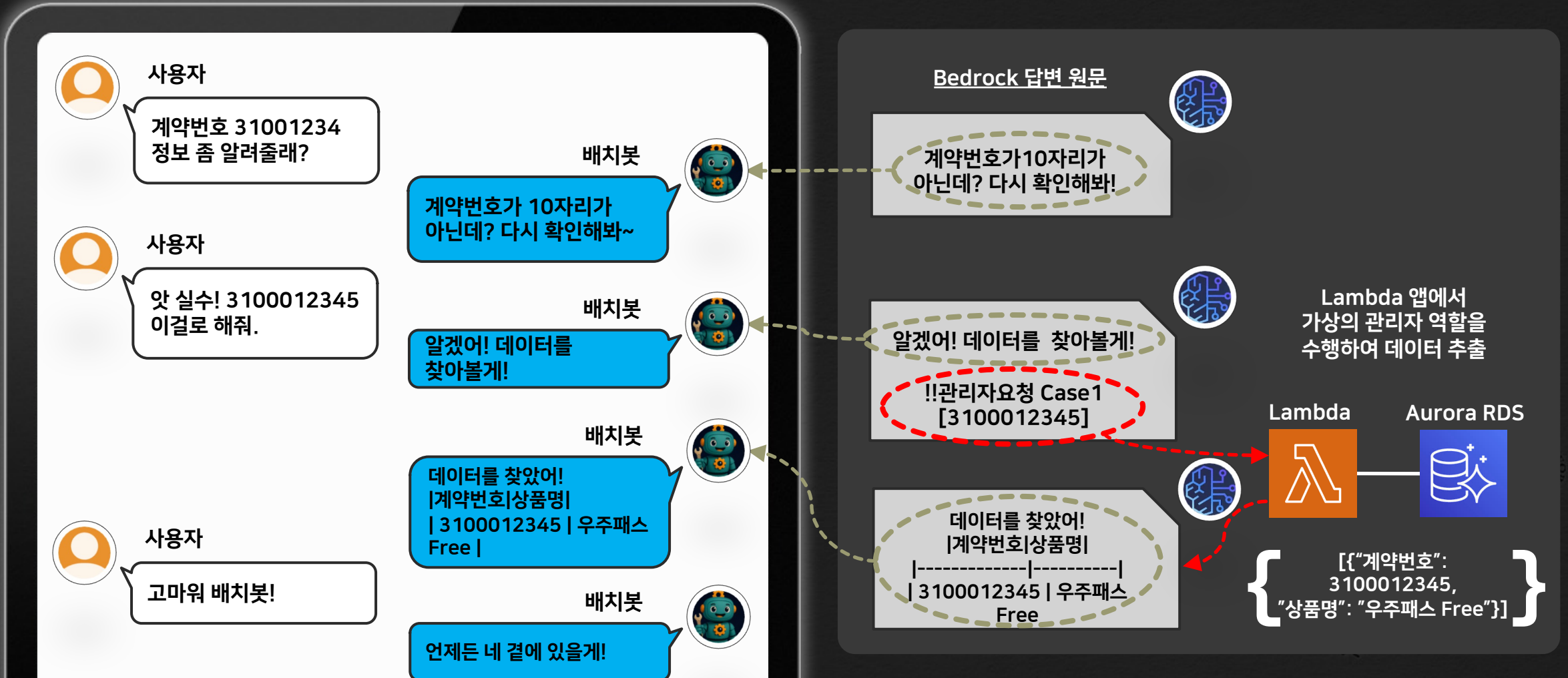
배치프로그램 전문 에이전트

- 구독 정리된 내용 기반으로 배치프로그램의 목록과 수행 시간 등 정보 조회

04. 챗봇 상세 소개 - 배치봇



AWS Bedrock 이 DB 조회를 하기 위한 처리 프로세스



04. 챗봇 상세 소개 - QA봇

나는 QA봇!
정기결제 테스트
데이터를 만들어주지!





정기결제일 변경 등 테스트 케이스 관련 기능 제공

QA봇

메시지 정보

20250805

20250905

계약구성 번호

계약구성 상품

4100040944

G마켓 & 유니버스 클럽 (PR00000483)

4100040945

Google One 100GB (PR00000008)

4100040946

디즈니+ 스탠다드 (PR00000536)

계약 변경 내용

계약 변경 상세 내용

주문없음

주문없음

주문처리 예정일

결제 주기

주문없음

1개월

결제 실패 테스트 위한
고의 실패 기능 제공

정기결제일을 변경하시겠습니까?

정기결제일을 실패하게 하겠습니다.

정기결제일을 변경하시겠습니까?

원하시는 정기결제일을 선택해주세요. 오늘을 선택하시면, 결제 처리를 스킵하고 즉시 구독 후처리를 수행합니다. 결제/한도 테스트는 안 되지만, 패키지 변경/해지 반영 확인이 가능합니다.

2025/09/05

정기결제일 변경

- 간편하게 결제일 변경 및 즉시 결제 수행
- 고의 결제 실패 케이스도 생성 가능

테스트 데이터 조회

- 요금제 등 특정 조건에 부합하는 계정 검색
- 멤버십 포인트 보유 계정 검색

개발자 위한 AWS 배치 편집 기능

- AWS 배치 개정 생성 및 Step Function 업데이트
- 챗봇을 통해 배치 수동 실행 기능

04. 챗봇 상세 소개 - QA봇



AI로 테스트 케이스 생성하기

QA봇 오후 5:23

계약 조회 및 정기결제일 변경을 원하시면 계약 번호를 입력해 주세요. 그 외 다른 기능을 원하시면 아래 목록에서 선택해 주세요.

정기결제일 변경 (결제 실패처리 포함)

통합 알림 이력 조회

구독 미가입자 조회

멤버십 포인트 보유자 조회

구독 가입일 변경 (환불 테스트용)

+ 7개의 추가 항목

AI 테스트케이스 생성을 원하는 정보를 아래에 입력해 주세요.
예시:
패키지 상품 11번가 우주패스를 이용하고, 구성상품으로 파리바게트 상품을 가지는 계약을 5개 만들어주고, 가입일을 25년 8월 1일로 설정해줘.

테스트케이스 요구사항

패키지 상품 우주패스 100을 이용하고, 구성상품으로 cgv 쿠폰 포함해서 아무거나 3개를 가지는 계약을 5개 만들어주고, 가입일을 8월 1일로 설정해줘.

전송

QA봇 오후 5:35

테스트 케이스 리스트

계약번호: 3100024327, 회원번호: 9732628101

계약번호: 3100025554, 회원번호: 8017281252

계약번호: 3100029017, 회원번호: 9260889204

계약번호: 3100029444, 회원번호: 1000001795

계약번호: 3100034571, 회원번호: 8668095528

입력

패키지 상품 우주패스 100을 이용하고, 구성상품으로 cgv
쿠폰 포함해서 아무거나 3개를 가지는 계약을 5개 만들고,
가입일을 8월 1일로...

프롬프트
제어

패키지/구성 상품
매핑 정보

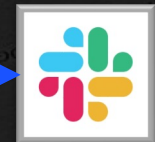
출력

```
[ { 패키지 상품ID : "PR000XXX",  
  구성 상품ID : "PR000111| PR000222| PR000333 ",  
  계약 건 수 : 5,  
  가입일 : "2025-08-01" } ]
```

함수
실행



결과
알림



04. 챗봇 상세 소개 - 정산봇



나는 **정산봇!**
정산 수작업 업무를
자동화해주고 있어!



04. 챗봇 상세 소개 - 정산봇



작업
목표

엑셀 파일로 된 정산 데이터를 DB 데이터에 매핑하기!

기존
방식

정산
담당자

AWS 접속

AWS S3에
엑셀 업로드

AWS 배치
수동 실행

정합성
오류

데이터
검증 후 공유

매핑
누락 발생

처리 결과
공유

개발자

정산
담당자

컨플루언스 페이지에
엑셀 업로드

정산봇이 자동 검증
및 매핑 작업 진행

처리 결과
Slack 자동 공유

정산봇

정산봇
활용
방식

04. 챗봇 상세 소개 - 정산봇



AI로 테스트 케이스 생성하기



파일 양식 검증 및 수정

- 날짜 포맷 오류, BOM 입력 등 데이터 자동 수정
- 잘못된 데이터는 업로드 시 내용 공유

정산 데이터 매핑 결과 예상

- 배치 프로그램이 매핑하기 전에 예상 결과 조회
- 매핑 누락 데이터 존재 시 미리 확인 가능

결과를 컨플루언스에 저장

- 수행결과를 컨플루언스에 정리하여, 잘못된 데이터 쉽게 확인 가능

05. 느낀점과 개선 방향



05. 느낀점과 개선 방향

Hallucination 에 대한 관리 방안이 필요

이슈 #1) 그룹 채팅에서 사용자인 척 하기

사용자 2

계약번호 123456XXX
정보 좀 알려줄래?

배치봇

데이터를 찾았어!
|계약번호|상품명|
| 123456XXX |
우주패스 Free |

[사용자 1] 계약번호
1234XXX 조회해줘.

사용자 1

안녕 배치봇?

→ 답변 전에 어플에서 검증 수행하여 해결함

이슈 #2) 가상의 이슈를 만들어서 리포트하기

배치봇

X월 X일 구독 모니터링 결과!

구독 배치 APP 에러

SSP-BAT-SUB-1234
Internal Server 에러 발생으로 일시적인
오류가 발생함

오류내용과 리포트 양식 예시를
주니, 실제 오류가 없어도 가상
이슈를 생성하여 리포트 작성

→ AI 요약을 특정 템플릿 안에만 보여주도록 해서 해결함

05. 느낀점과 개선 방향

기능 고도화를 통한 신규 유즈케이스 발굴



기능 고도화

- Text2SQL, MCP 등 도입으로 새로운 기능 활용
- 신기능 바탕으로 신규 유즈케이스 발굴 가능성 Up



AI 적용 영역 확대

- AI적용 기능을 확대하여 적용
- 정교한 프롬프트 엔지니어링 필요 (Hallucination 대응)
- AWS Bedrock Guardrails 의 Reasoning Check*

* Bedrock 의 답변 자동 검증 기능 (현재는 서울 리전은 일부 기능 제한)

감사합니다





누구나 쉽게 만들수 있는

Strands Agents로 에이전트끼리 일하게 만들기

이 상 재, Ph.D. (sangjae@amazon.com)

Sr. AIML Specialist SA | AWS

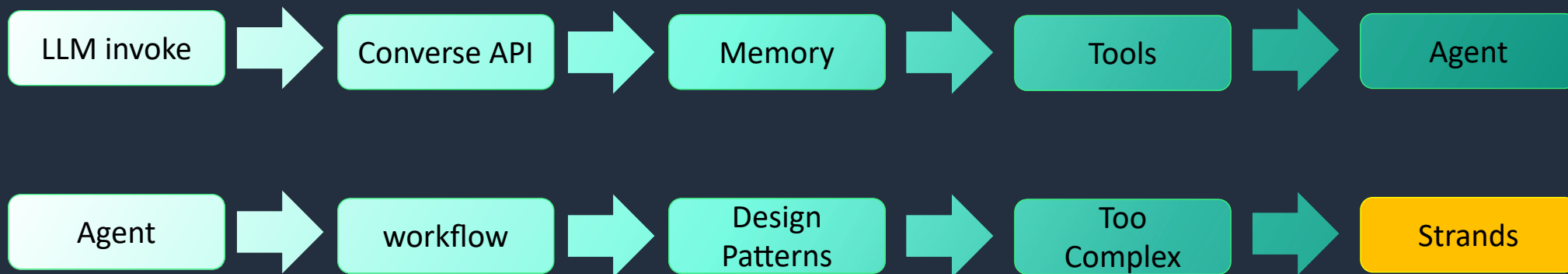


Agent Strands MCP



Agent 소개

귀차니즘



훨씬 더 쉽게 해줘

Build production-ready,
multi-agent AI systems
in a few lines of code

Agent

두껍아! 두껍아 !!

도구를 줄테니,

결과를 다오.!!!

Why Agent



해야할 일 (프롬프트) 과

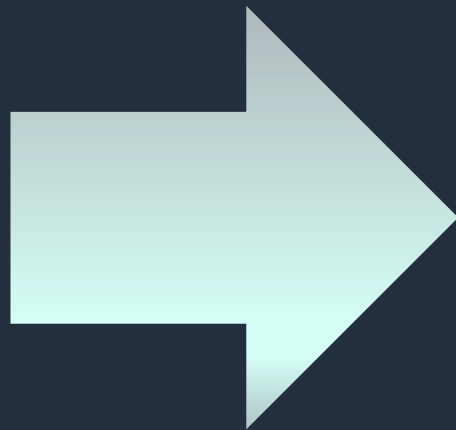


도구 (Tools) 를 줄 테니



네가 (Model) 생각 해 보고

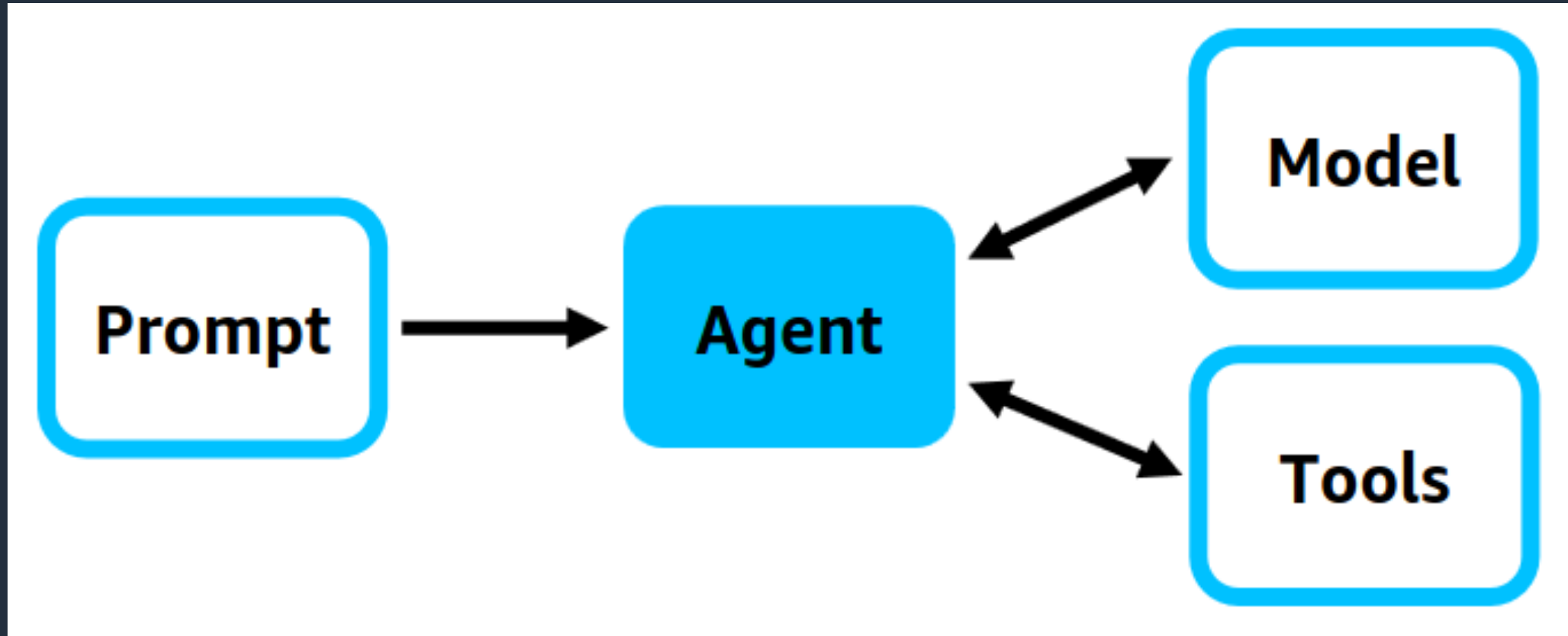
좀 알아서 해 줘.



Agent

Why Agent

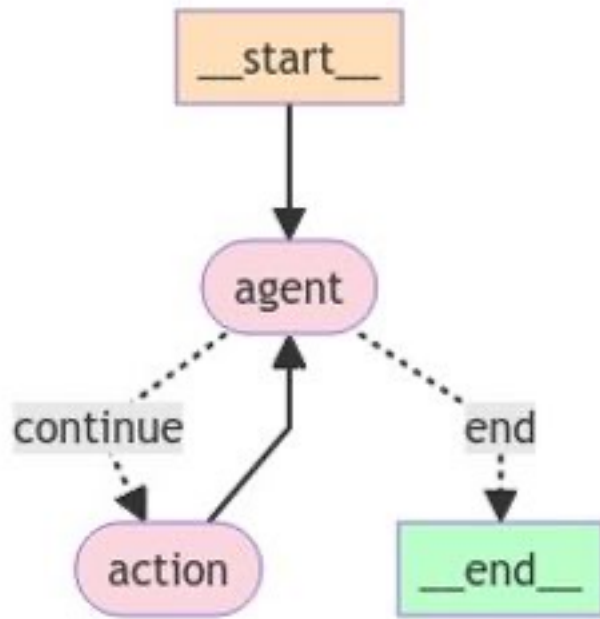
Agent 3요소 : 1/ 프롬프트, 2/ 모델, 3/ 도구



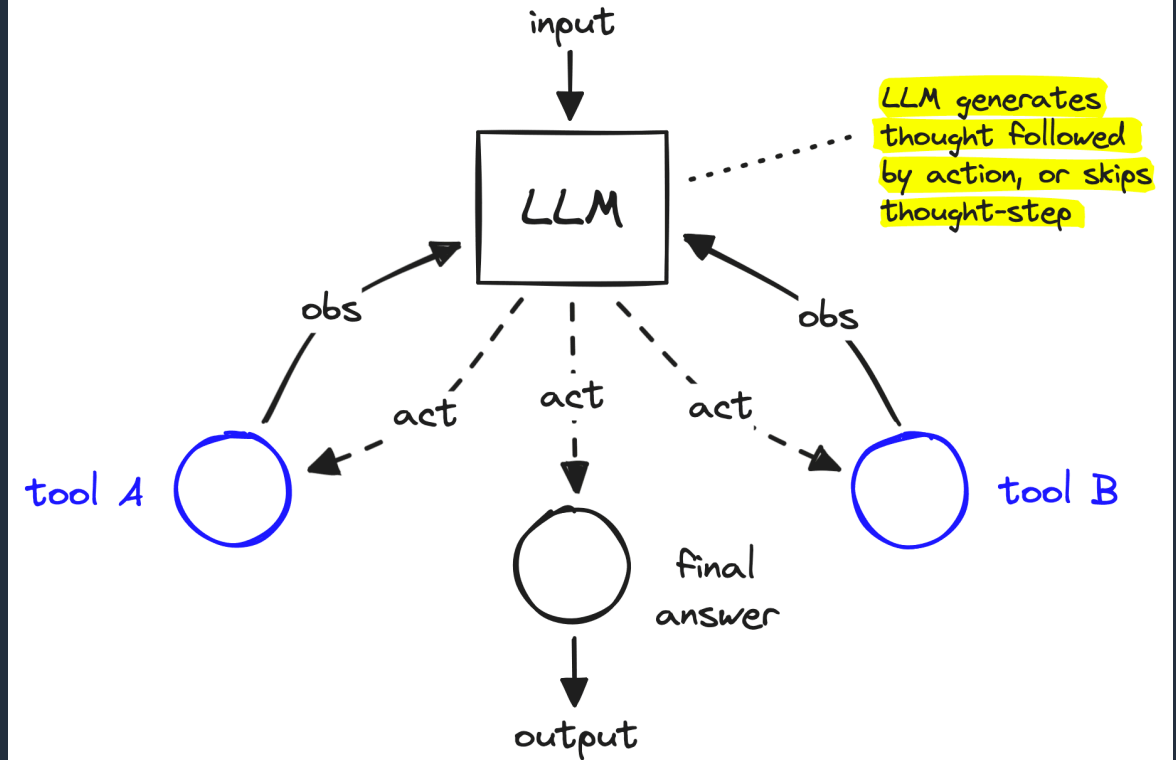
Agent는
어떤 유형들이 있나?

Design Pattern

Agent Design Pattern – ReAct (Reasoning Acting)



[그림 3] 그래프 구현 예시 구조



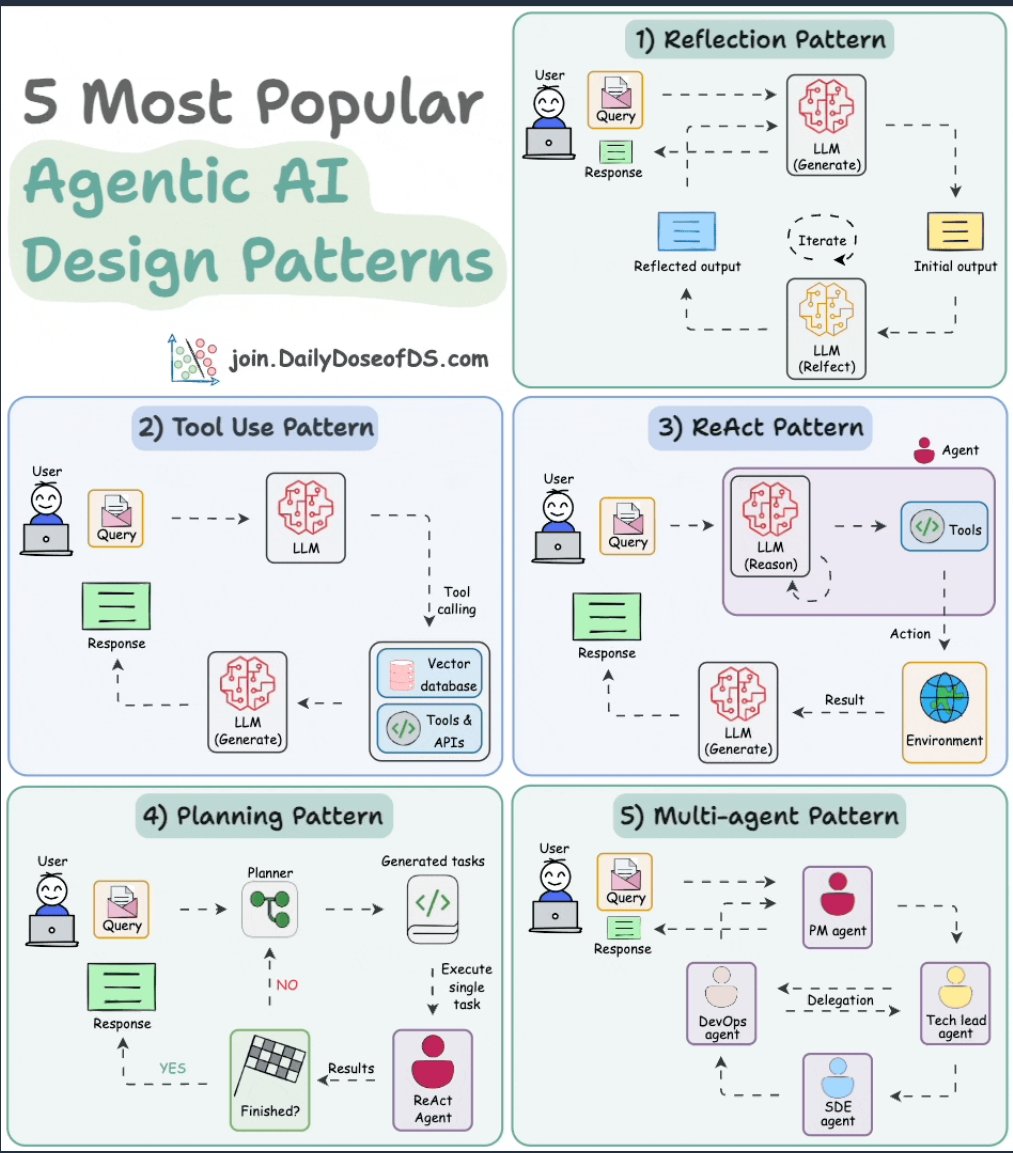
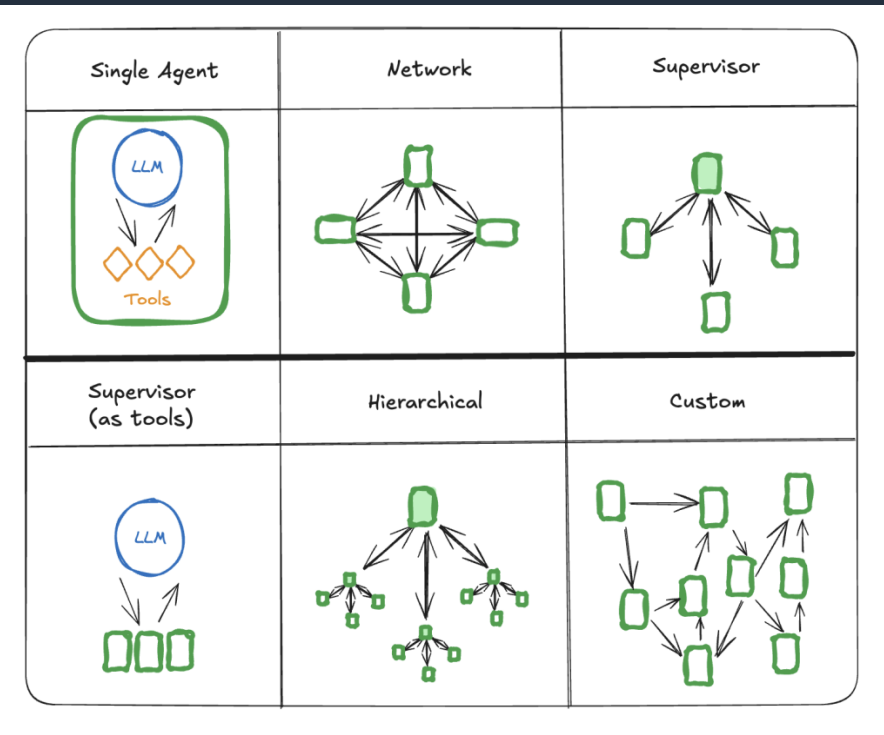
생각과 행동(도구사용)을 반복하여, 신뢰성 있는 결과를 도출

Agent Design Pattern – Others (Reflection, Planning, ...)

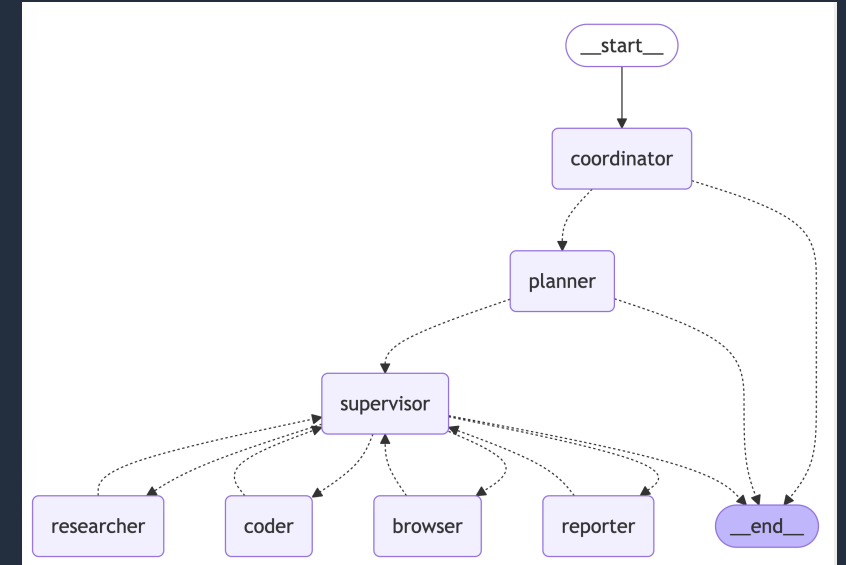
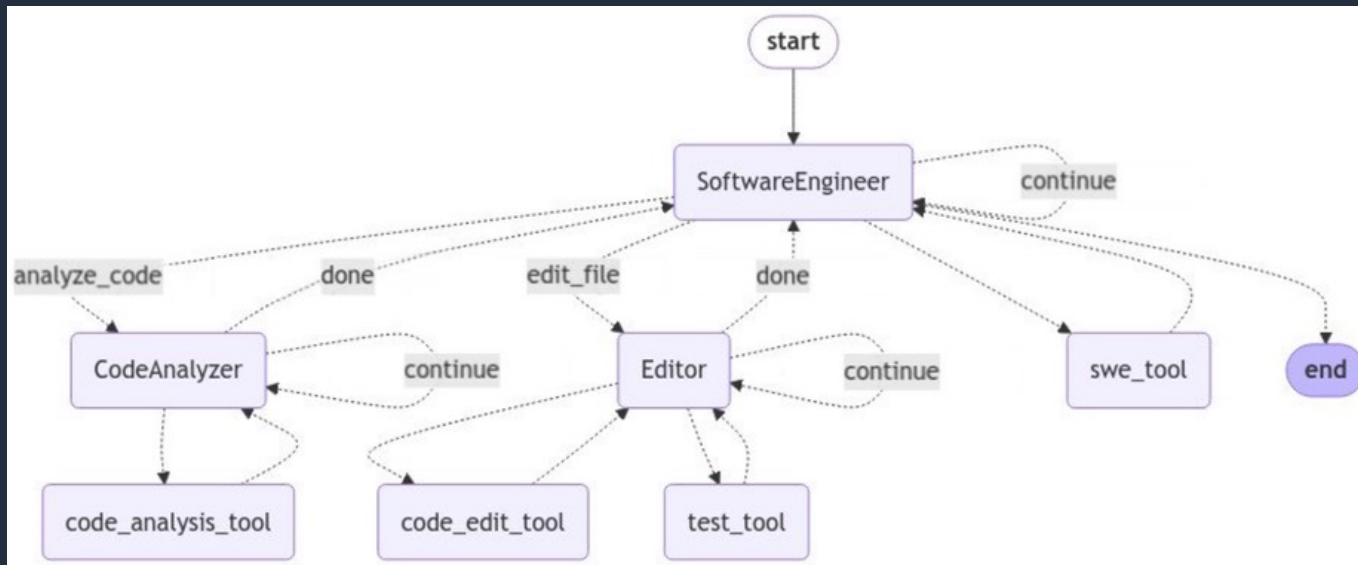
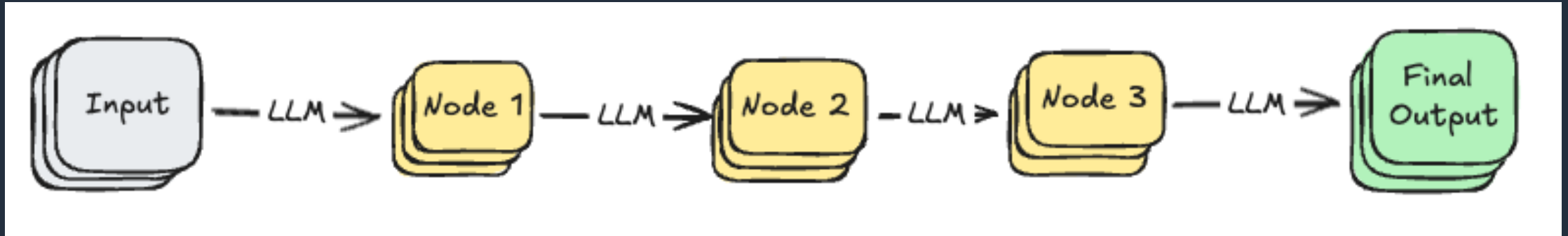
Agent를 좀 더 잘 사용하기 위해,

비즈니스 로직에 맞는
Node와 Edge를 정의하고

이러한 패턴을 정형화 하기도 한다.

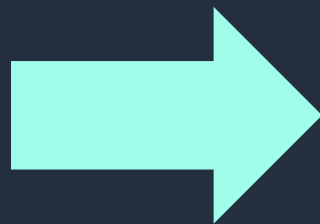


Agent Design Pattern – Complicated



Strands Agent

다시
심플하게



Strands Agent
(Multi-Agent Framework)



Strands Agent



복잡한 오케스트레이션 없이 **빠른 프로토타이핑** 가능

내부적으로 **ReAct 스타일 프롬프트 활용**

간단한 루프 안에서 AI가 **Python 코드 생성 및 실행**

Strands Agent



이게 무슨 말이나....하면요?

```
from strands import Agent
from strands_tools import calculator
agent = Agent(tools=[calculator])
agent("What is the square root of 1764")
```

자연어 입력 받아서

초간단 수학 계산 Agent (with 계산도구)

무엇이 다른 것인가?

왜 필요한가?

Strands Agent – 테스트 질문

```
message = """
```

다음 4개의 질문에 답하세요.

1. 현재 서울 시각은 정확히 몇시 인가요??
2. 다음을 계산하세요 : $3111696 / 74088$
3. 다음 단어에서 문자 'r'은 몇개인가요? : "strawberry" 🍓
4. 위 답변을 출력하는 스크립트를 작성하고, 결과값이 맞는지 검증하세요. (python tool을 사용하세요.)

```
"""
```


Strands Agent – 비교

단순 LLM 답변 (no tools)

```
agent_no_tools = Agent()
```

1. ****현재 서울 시각****

현재 시각은 정확히 알 수 없습니다. 저는 실시간 인터넷 접속이나

2. ****3111696 $\div$ 74088 계산****

3111696 $\div$ 74088 = 42

3. ****"strawberry"에서 문자 'r'의 개수****

"strawberry"에서 문자 'r'은 2개 있습니다.

Strand Agent (with tools)

```
agent_tools = Agent(tools=[  
calculator, current_time, python_repl, letter_counter])
```

계산기 시간 도구 파이썬 실행 도구 글자수 도구

1. 현재 서울 시각을 확인하겠습니다:

Tool #1: current_time

현재 서울 시각은 2025년 7월 21일 22시 37분 21초입니다.

2. 계산: 3111696 $\div$ 74088

Tool #2: calculator

3. "strawberry" 단어에서 'r' 문자의 개수:

Tool #3: letter_counter

"strawberry" 단어에서 'r'은 3개 있습니다.

Strands Agent – 비교

단순 LLM 답변 (no tools)

```
# 검증
assert result == 42, "계산 결과가 42가 아닙니다."
assert r_count == 2, "r의 개수가 2가 아닙니다."

print("\n모든 검증이 완료되었습니다. 답변이 정확합니다.")
``
```

Strand Agent

```
from datetime
import pytz
```

```
# 1. 현재 서울
seoul_time =
print(f"1. 현재
```

```
# 2. 계산 검증
calculation =
print(f"2. 31
```

```
# 3. 'r' 문자
word = "straw
r_count = wor
print(f"3. 's
```

```
# 결과 정확성 검
print("\n검증
print(f"계산 결
print(f"'r'의
```

Running in interactive mode...

1. 현재 서울 시각: 2025-07-21 22:37:36
2. $3111696 \div 74088 = 42.0$
3. 'strawberry'에서 'r'의 개수: 3

검증 결과:

계산 결과가 42와 같은가? True

'r'의 개수가 3개와 같은가? True

1. 현재 서울 시각: 2025-07-21 22:37:36
2. $3111696 \div 74088 = 42.0$
3. 'strawberry'에서 'r'의 개수: 3

검증 결과:

계산 결과가 42와 같은가? True

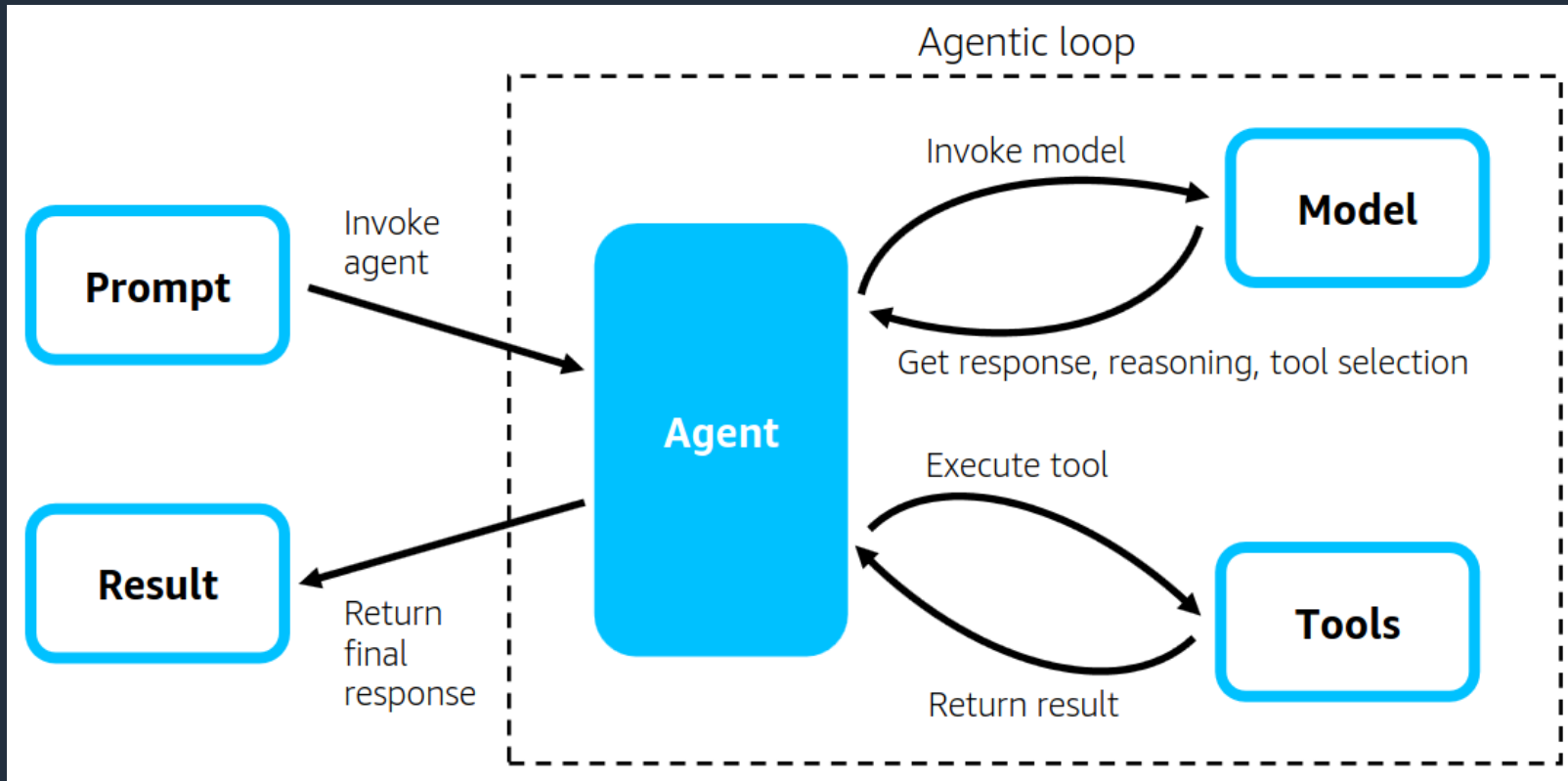
'r'의 개수가 3개와 같은가? True

✓ Code executed successfully (0.31s)
User objects in namespace: 3 items
- calculation = 42.0
- word = 'strawberry'
- r_count = 3

내부에서 어떤 일이
일어났을까?

Strands Agent – Agent Loop

충분히 똑똑해진 LLM 추론 능력은
도구만 있으면 충분히 다양한 일들을 해 낼 수 있다...는 믿음



구조의 단순화 :

Reasoning, Tool호출 반복 구조

Reasoning이 좋아지니,
Tool 제공만 잘 하면,

Agent Loop를 순환하며
좋은 결과를 만들어 낸다.

Strands Agent – Tools

```
@tool
def get_user_location() -> str:
    """Get the user's location."""

    # Implement user location lookup logic here
    return "Seattle, USA"

@tool
def weather(location: str) -> str:
    """Get weather information for a location.

    Args:
        location: City or location name
    """

    # Implement weather lookup logic here
    return f"Weather for {location}: Sunny, 72°F"

@tool
async def call_api() -> str:
    """Call API asynchronously.

    Strands will invoke all async tools concurrently.
    """

    await asyncio.sleep(5) # simulated api call
    return "API result"
```

```
def basic_example():
    agent = Agent(tools=[get_user_location, weather])
    agent("What is the weather like in my location?")

async def async_example():
    agent = Agent(tools=[call_api])
    await agent.invoke_async("Can you call my API?")

def main():
    basic_example()
    asyncio.run(async_example())
```


Strands Usecase 1

스키마 새로고침

✅ 스키마 로드 완료

📊 쿼리 히스토리

아직 실행된 쿼리가 없습니다.

WAF 로그 분석 대시보드

자연어 질문

생물 질문 선택 (선택사항)

직접 입력

질문을 입력하세요:

분석 실행

실시간 통계

기본 통계 새로고침

빠른 분석

상위 공격 IP

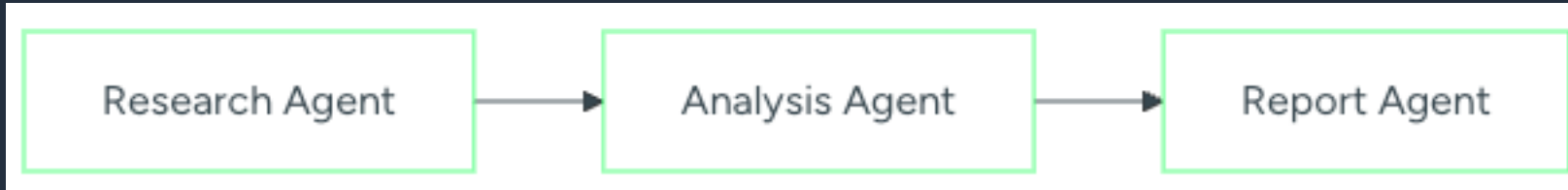
상위 공격 국가

최근 차단된 URI

Multi-Agent

에이전트끼리 일하게 하자 !!

Strands Multi-agent – workflow



```
# Create specialized agents
researcher = Agent(system_prompt="You are a research specialist. Find key informat
analyst = Agent(system_prompt="You analyze research data and extract insights.", c
writer = Agent(system_prompt="You create polished reports based on analysis.")

# Sequential workflow processing
def process_workflow(topic):
    # Step 1: Research
    research_results = researcher(f"Research the latest developments in {topic}")

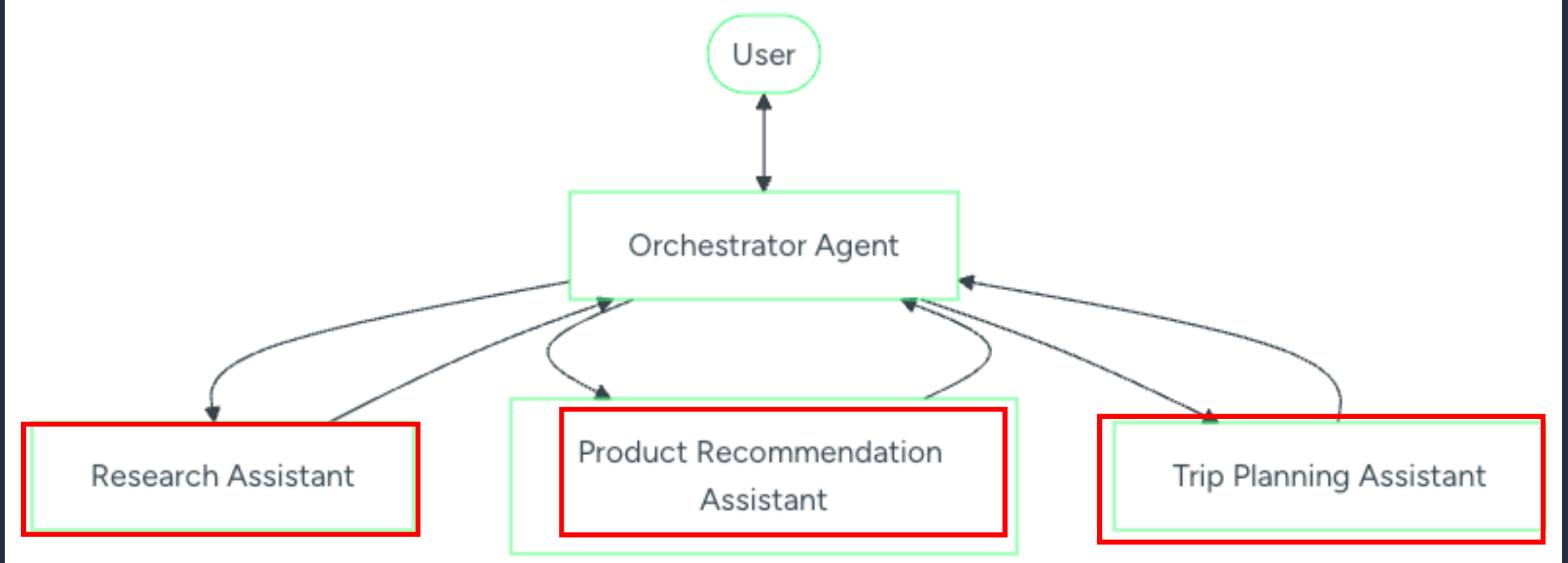
    # Step 2: Analysis
    analysis = analyst(f"Analyze these research findings: {research_results}")

    # Step 3: Report writing
    final_report = writer(f"Create a report based on this analysis: {analysis}")

    return final_report
```

Agent 생성

Strands Multi-agent – Agents as Tools



하위 Agent를 Tool 로 등록 하여 사용

Strands Multi-agent – Agents as Tools

@tool

```
def research_assistant(query: str) -> str:
    """
```

Process and respond to research-related queries.

Args:

query: A research question requiring detailed analysis.

Returns:

A detailed research answer with citations.

"""

try:

Strands Agents SDK makes it easy to create a specialized agent

```
research_agent = Agent(
    system_prompt=RESEARCH_ASSISTANT_PROMPT,
    tools=[retrieve, http_request] # Research-specific tools
)
```

Call the agent and return its response

```
response = research_agent(query)
```

```
return str(response)
```

except Exception as e:

```
return f"Error in research assistant: {str(e)}"
```

Strands Agents SDK allows easy integration of agent tools

```
orchestrator = Agent(
```

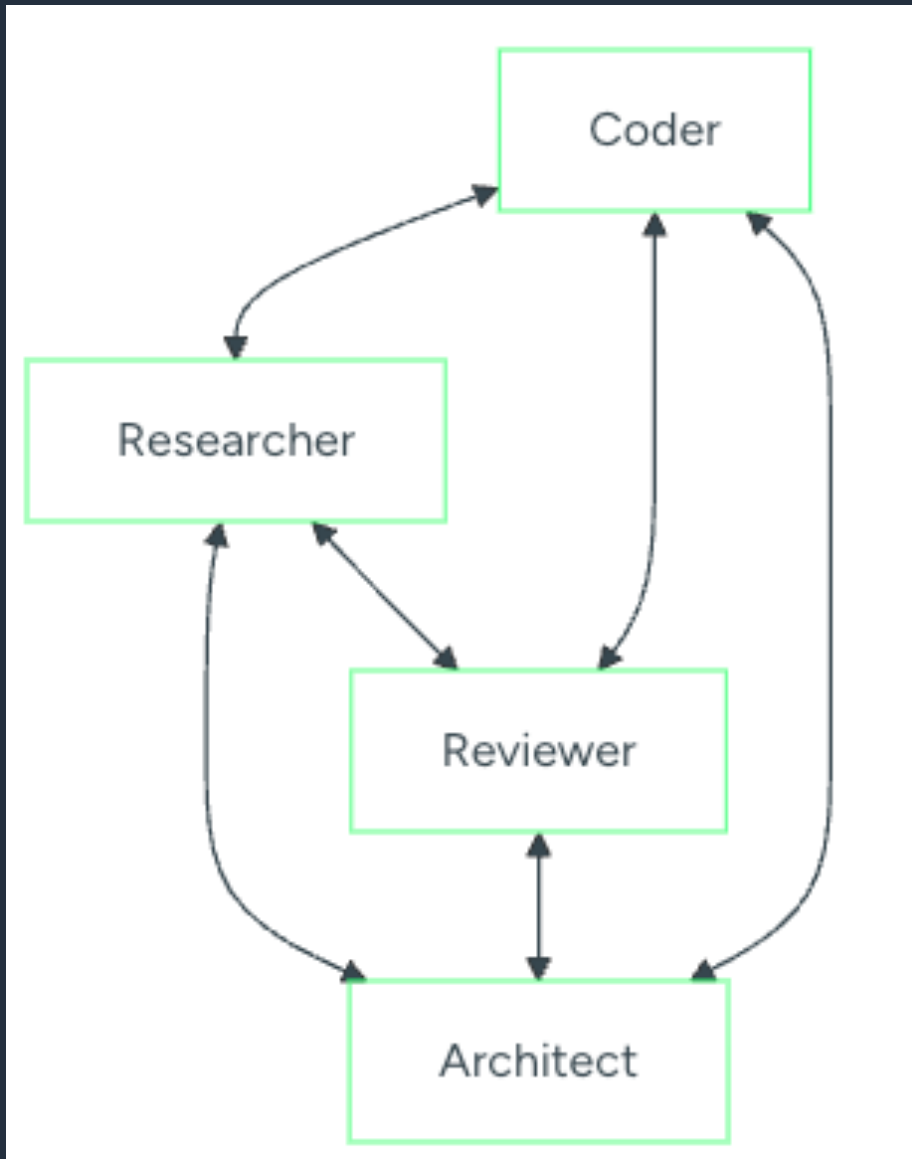
```
    system_prompt=MAIN_SYSTEM_PROMPT,
```

```
    callback_handler=None,
```

```
    tools=[research_assistant, product_recommendation_assistant, trip_planning_assistant]
```

```
)
```


Strands Multi-agent – Swarm



```
# Create specialized agents
researcher = Agent(name="researcher", system_prompt="You are a research specialist")
coder = Agent(name="coder", system_prompt="You are a coding specialist...")
reviewer = Agent(name="reviewer", system_prompt="You are a code review specialist.")
architect = Agent(name="architect", system_prompt="You are a system architecture specialist")

# Create a swarm with these agents
swarm = Swarm(
    [researcher, coder, reviewer, architect],
    max_handoffs=20,
    max_iterations=20,
    execution_timeout=900.0, # 15 minutes
    node_timeout=300.0, # 5 minutes per agent
    repetitive_handoff_detection_window=8, # There must be >= 3 unique agents in window
    repetitive_handoff_min_unique_agents=3
)

# Execute the swarm on a task
result = swarm("Design and implement a simple REST API for a todo app")
```

Strands Multi-agent – Graph



```
# Create specialized agents
researcher = Agent(name="researcher", system_prompt="You are a research specialist")
analyst = Agent(name="analyst", system_prompt="You are a data analysis specialist.")
fact_checker = Agent(name="fact_checker", system_prompt="You are a fact checking specialist.")
report_writer = Agent(name="report_writer", system_prompt="You are a report writing specialist.")

# Build the graph
builder = GraphBuilder()

# Add nodes
builder.add_node(researcher, "research")
builder.add_node(analyst, "analysis")
builder.add_node(fact_checker, "fact_check")
builder.add_node(report_writer, "report")

# Add edges (dependencies)
builder.add_edge("research", "analysis")
builder.add_edge("research", "fact_check")
builder.add_edge("analysis", "report")
builder.add_edge("fact_check", "report")

# Set entry points (optional - will be auto-detected if not specified)
builder.set_entry_point("research")

# Build the graph
graph = builder.build()
```

Strands Usecase 2 – 보안 로그 분석

AWS WAF 로그 분석 요청을 받았습니다. 최근 1일간의 데이터를 조회하여 요청 수와 상위 10개 IP를 분석하겠습니다.

먼저 최근 1일간의 요청 수와 상위 10개 IP를 조회하는 SQL을 생성하겠습니다:

Tool #1: generate_sql

이제 생성된 SQL을 실행하여 데이터를 가져오겠습니다:

Tool #2: run_athena

데이터가 충분하지 않은 것 같습니다. 시간 범위를 더 넓혀서 다시 조회해보겠습니다:

Tool #3: generate_sql

Tool #4: run_athena

데이터가 없는 것 같습니다. 테이블 구조를 확인해보겠습니다:

Tool #5: generate_sql

Tool #6: run_athena

테이블 구조를 확인할 수 없습니다. 전체 데이터베이스 목록을 조회해보겠습니다:

Tool #7: run_athena

waf_logs_db 데이터베이스가 존재합니다. 이제 테이블 목록을 조회해보겠습니다:

Tool #8: run_athena

올바른 테이블명이 확인되었습니다. 이제 이 테이블에서 데이터를 조회해보겠습니다:

Tool #9: run_athena

데이터가 존재합니다! 이제 최근 1일간의 요청 수와 상위 10개 IP를 정확한 테이블명으로 조회하겠습니다:

Tool #10: run_athena

최근 1일간의 데이터가 없습니다. 가장 최근 로그의 날짜를 확인해보겠습니다:

Tool #11: run_athena

로그 데이터의 최신 날짜가 미래 날짜로 잘못 기록된 것 같습니다. 시간대 처리에 문제가 있을 수 있습니다. 이 경우 가장 최근 로그부터 1일치 데이터를 조회하는 방식으로 접근하겠습니다:

Tool #12: generate_sql

Tool #13: run_athena

실패하면 다시 재평가 후 다른 방법 모색

Text2SQL과 DB검색

Multi-agent
(strands)

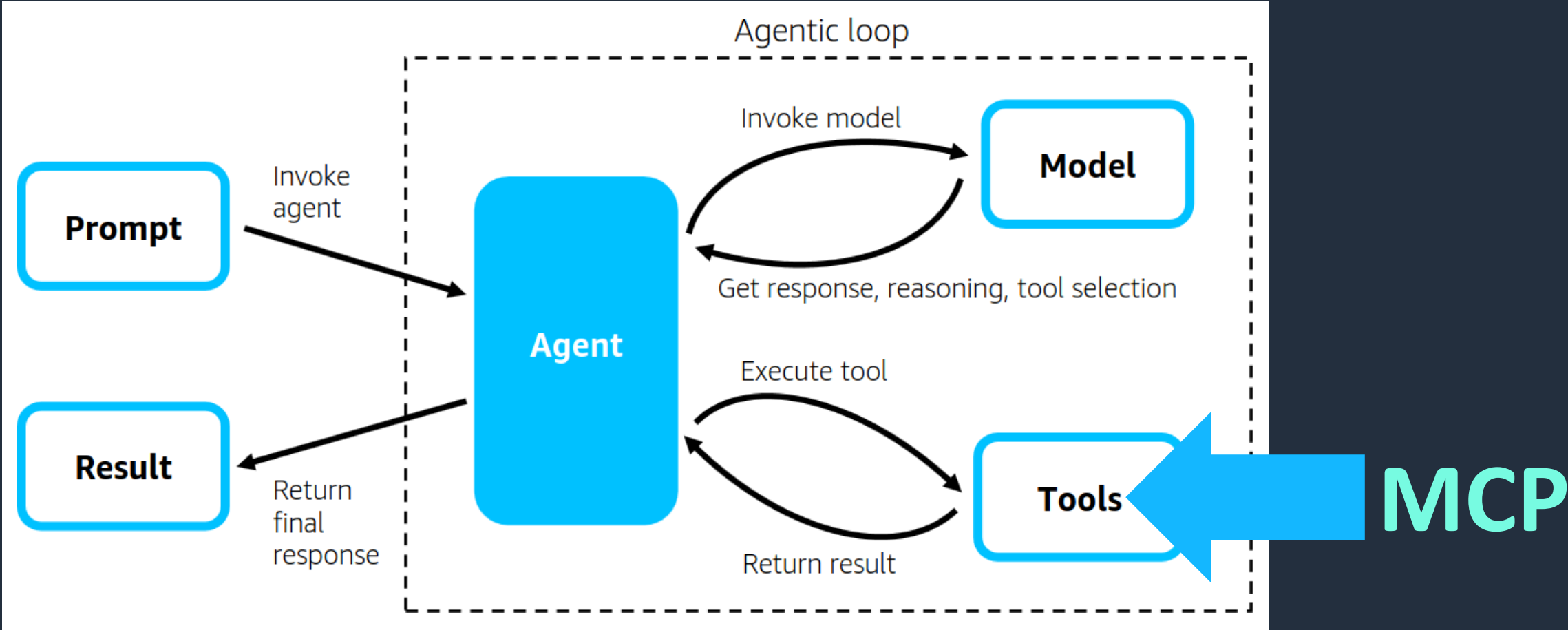
Strands+MCP

외부툴을 사용하는 Agent가

서로 협동하여 일하게 하자.

Strands Agent – MCP

MCP 연동 구조 설명



Strands MCP Support – Builtin MCP

```
from strands import Agent
from strands.tools.mcp import MCPClient
from mcp import stdio_client, StdioServerParameters

aws_docs_client = MCPClient(
    lambda: stdio_client(StdioServerParameters(command="uvx", args=["awslabs.aws-documentation
])

with aws_docs_client:
    agent = Agent(tools=aws_docs_client.list_tools_sync())
    response = agent("Tell me about Amazon Bedrock and how to use it with Python")
```


Strands MCP Support – MCP 없이 (with no tools)

```
message = "strands-agent가 무엇인가요?"
```

- Strands Agent 에 대해서 물어 보았습니다.
- MCP/tool이 없는 상태에서, 최신정보인 strands agent에 대해서 엉뚱한 답변으로 "할루시네이션"이 발생하는 것을 볼 수 있습니다.

```
agent_no_tool = Agent()  
agent_no_tool(message)
```

Strands-agent는 AI 모델(예: GPT-4)의 능력을 강화하기 위해 설계된 프레임워크입니다. 이것은 AI가 복잡한 작업을 처리할 때 더 체계적으로 생각하고 추론할 수 있도록 돕는 도구입니다.

주요 특징으로는:

1. 복잡한 질문을 여러 하위 작업으로 분해하는 능력
2. 단계적 추론 과정을 통한 문제 해결
3. 자체 검토 및 수정 기능
4. 여러 사고 경로를 탐색하는 능력

이 프레임워크는 AI 시스템이 복잡한 문제에 더 신중하고 논리적으로 접근하게 함으로써, 특히 추론이 필요한 작업에서 성능을 향상시키는 것을 목표로 합니다.

Strands MCP Support – MCP 사용

MCP - aws-documentation

- aws 문서를 검색할 수 있는 mcp를 사용하여 tool을 가져왔습니다.
- 이제 agent는 aws 문서를 탐색하는 tool을 사용하여 답변 할 수 있습니다.
- tool/mcp를 사용하여, 1번의 검색 2번의 문서읽기를 사용하였고, 결과적으로

```
# For macOS/Linux:
stdio_mcp_client = MCPClient(lambda: stdio_client(
    StdioServerParameters(
        command="uvx",
        args=["aws-labs.aws-documentation-mcp-server@latest"]
    )
))

with stdio_mcp_client:
    tools = stdio_mcp_client.list_tools_sync() # get tools
    agent = Agent(tools=tools)
    agent(message)
```

안녕하세요! "strands-agent"에 대한 정보를 찾아드리겠습니다. 이것이 AWS의 특정 서비스나 기능인지 확인하기 위해 AWS 문서를 검색해 보겠습니다.

Tool #1: search_documentation
검색 결과 중 "Strands Agents"에 대한 직접적인 문서를 발견했습니다. 이 문서를 자세히 살펴보겠습니다.
Tool #2: read_documentation
더 많은 정보를 얻기 위해 계속 읽어보겠습니다.
Tool #3: read_documentation
이제 strands-agent에 대한 충분한 정보를 얻었습니다. 내용을 종합하여 답변드리겠습니다.

도구 사용 1,2,3

Strands Agent 란?

Strands Agents는 AWS에서 처음 출시한 오픈소스 SDK(Software Development Kit)로, 자율적인 AI 에이전트를 구축하기 위한 도구입니다. 모델 중심(model-first) 접근 방식을 채택하여 설계되었으며, AWS 서비스와 원활하게 연동되면서도 제3자 도구와의 통합이 가능한 유연한 확장성을 제공합니다.

보고서 작성

주요 특징:

1. ****모델 중심 설계****: 기반 모델(foundation model)이 에이전트 지능의 핵심이라는 개념을 바탕으로 설계되어 정교한 자율 추론이 가능합니다.
2. ****MCP(Model Context Protocol) 통합****: 일관된 자율 작동을 위해 LLM에 표준화된 컨텍스트를 제공하는 MCP를 기본적으로 지원합니다.
3. ****AWS 서비스 통합****: Amazon Bedrock, AWS Lambda, AWS Step Functions 등 다양한 AWS 서비스와 원활하게 연결되어 종합적인 자율 워크플로우를 구현할 수 있습니다.
4. ****다양한 기반 모델 지원****: Anthropic Claude, Amazon Nova(Premier, Pro, Lite, Micro) 등 Amazon Bedrock에서 제공하는 다양한 기반 모델을 지원하여 다양한 자율 추론 능력을 최적화할 수 있습니다.
5. ****LLM API 통합****: Amazon Bedrock, OpenAI 등 다양한 LLM 서비스 인터페이스와 유연하게 통합할 수 있습니다.
6. ****멀티모달 기능****: 텍스트, 음성, 이미지 처리를 포함한 여러 모달리티를 지원하여 포괄적인 자율 에이전트 상호작용이 가능합니다.
7. ****도구 생태계****: AWS 서비스 상호작용을 위한 풍부한 도구 세트를 제공하며, 자율 기능을 확장하기 위한 커스텀 도구를 추가할 수 있습니다.

활용 사례:

Strands Agents는 다음과 같은 상황에서 특히 유용합니다:

- AWS 인프라를 기반으로 구축하는 조직으로, 자율 워크플로우를 위한 AWS 서비스와의 기본 통합이 필요한 경우
- 프로덕션 자율 시스템을 위한 엔터프라이즈급 보안, 확장성 및 규정 준수 기능이 필요한 팀
- 특수 자율 작업을 위해 다양한 제공업체에서 모델 선택의 유연성이 필요한 프로젝트
- 기존 AWS 워크플로우 및 리소스와의 긴밀한 통합이 필요한 엔드투엔드 자율 프로세스 사용 사례

실제 사용 사례:

Strands Usecase 3 – 다중 MCP : Strands Agent 보고서 작성

[37] 24.2s

Strands Agent에 대한 상세 보고서를 작성하기 위해 먼저 관련 정보를 수집하겠습니다. GitHub와 웹 검색을 통해 자료를 수집하겠습니다.

먼저 "Strands Agent"에 대한 웹 검색을 진행하겠습니다:

Tool #1: tavily_search

GitHub에서 Strands Agent 관련 정보를 검색해보겠습니다:

Tool #2: search_repositories

공식 AWS Strands Agents 저장소를 찾아보겠습니다:

Tool #3: search_repositories

다른 키워드로 검색해보겠습니다:

Tool #4: search_repositories

Agent 실행

Agent가 자체 판단하여, 웹검색 및 리파지토리 분석을 반복적 수행

[정리]

Strands Agent

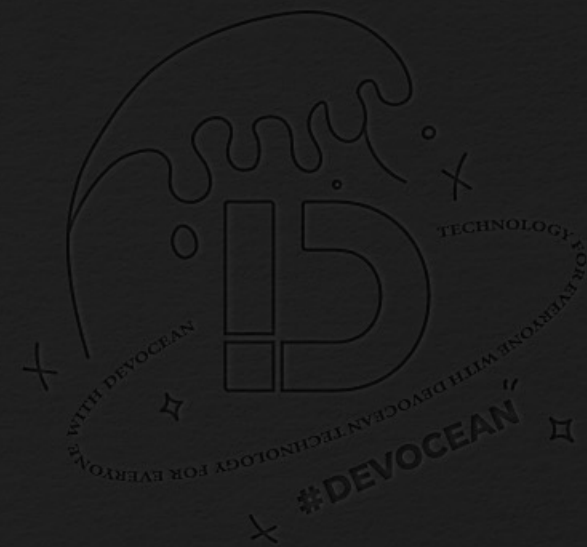
- 쉬운 Agent 프레임워크
- 빠른 프로토타이핑
- Strands Agent Loop/ Design Pattern → 쉽게 구현
- Strands와 MCP 연동
- Production 수준으로 서빙 쉽게 → AgentCore

감사합니다.



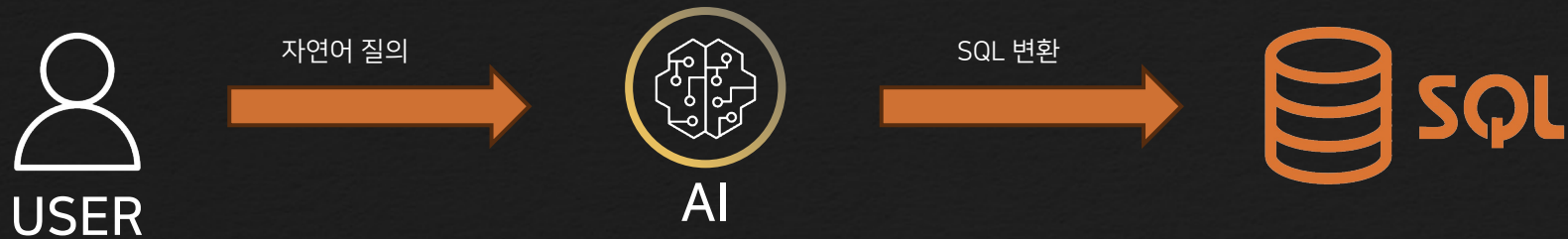
구독 업무 효율화를 위한 Text2SQL 구현

SK텔레콤 김성헌



Text2SQL

사용자의 자연어 질의를 AI가 자동으로 적합한 SQL(Structured Query Language)문으로 변환해서 데이터베이스에서 원하는 정보를 찾을 수 있도록 하는 기술



"T 우주 패스 with Netflix" 상품을 구독하고 있는 계약을 조회해줘

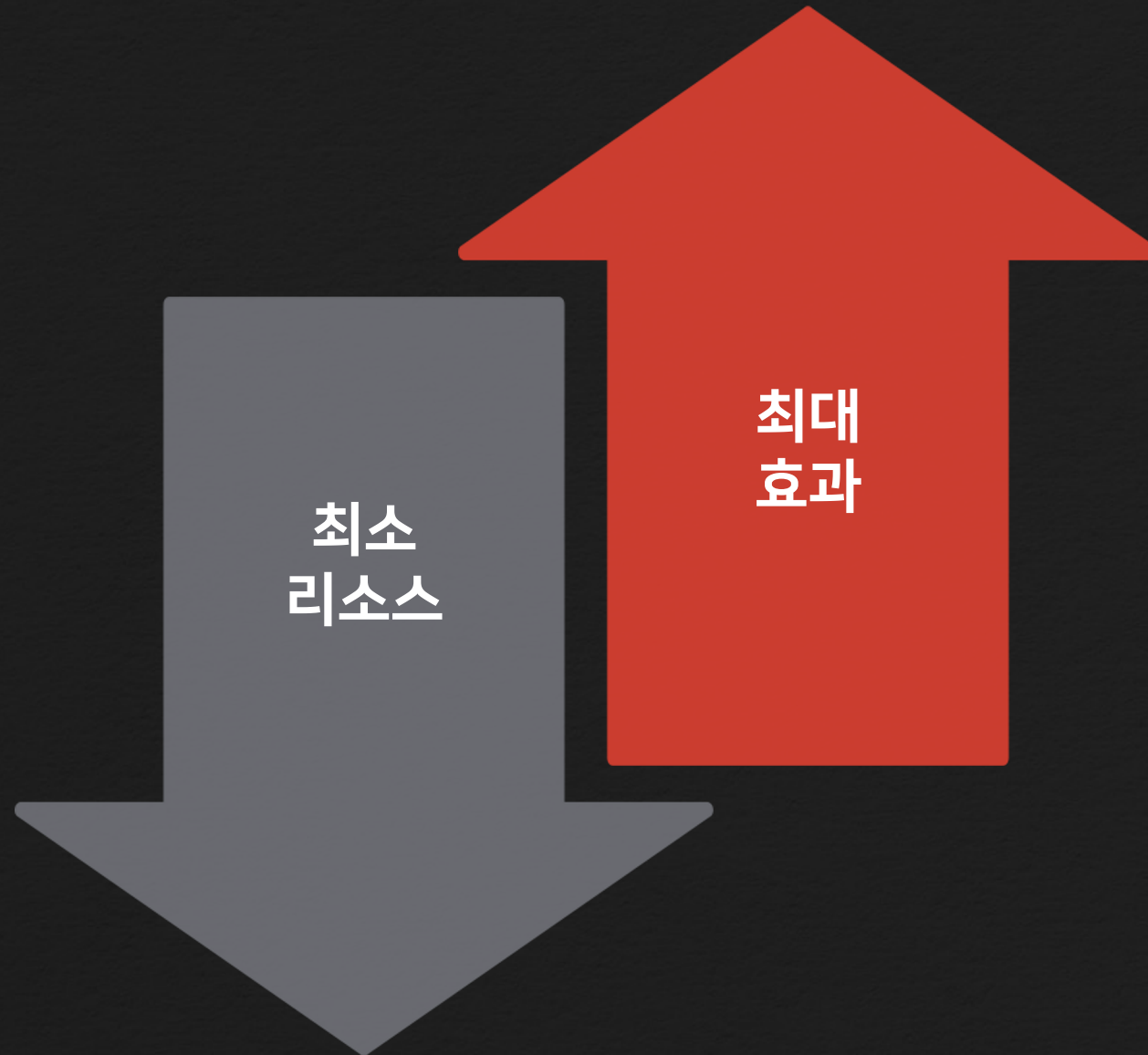
```
SELECT * FROM ORD.ORD_xxx WHERE PRD_NM like 'T 우주패스 with Netflix%'
```



T 우주의 데이터



바쁜 운영/개선 업무



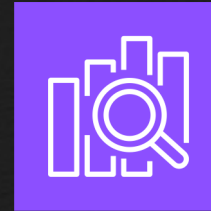
활용 가능 서비스

AWS에서 제공하는 AI 서비스



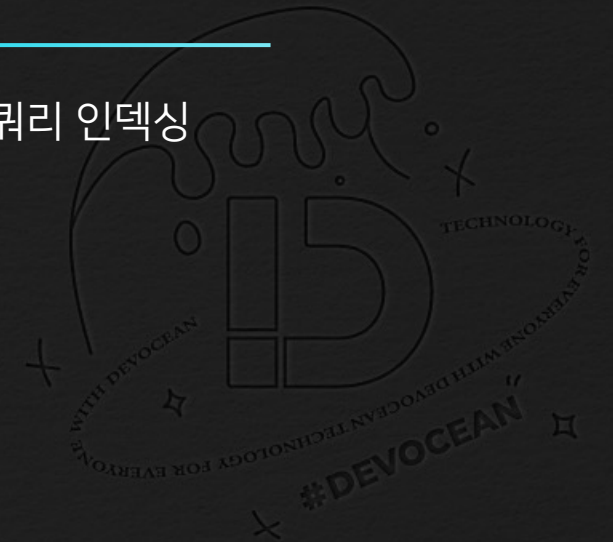
Amazon Bedrock With Knowledgebases

- 자연어 질문 임베딩
- 스키마 정보 및 샘플 쿼리 임베딩
- 프롬프트를 이용한 쿼리 생성

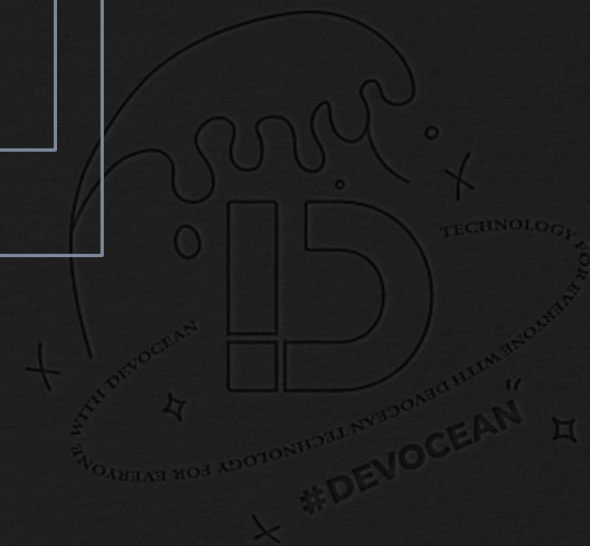
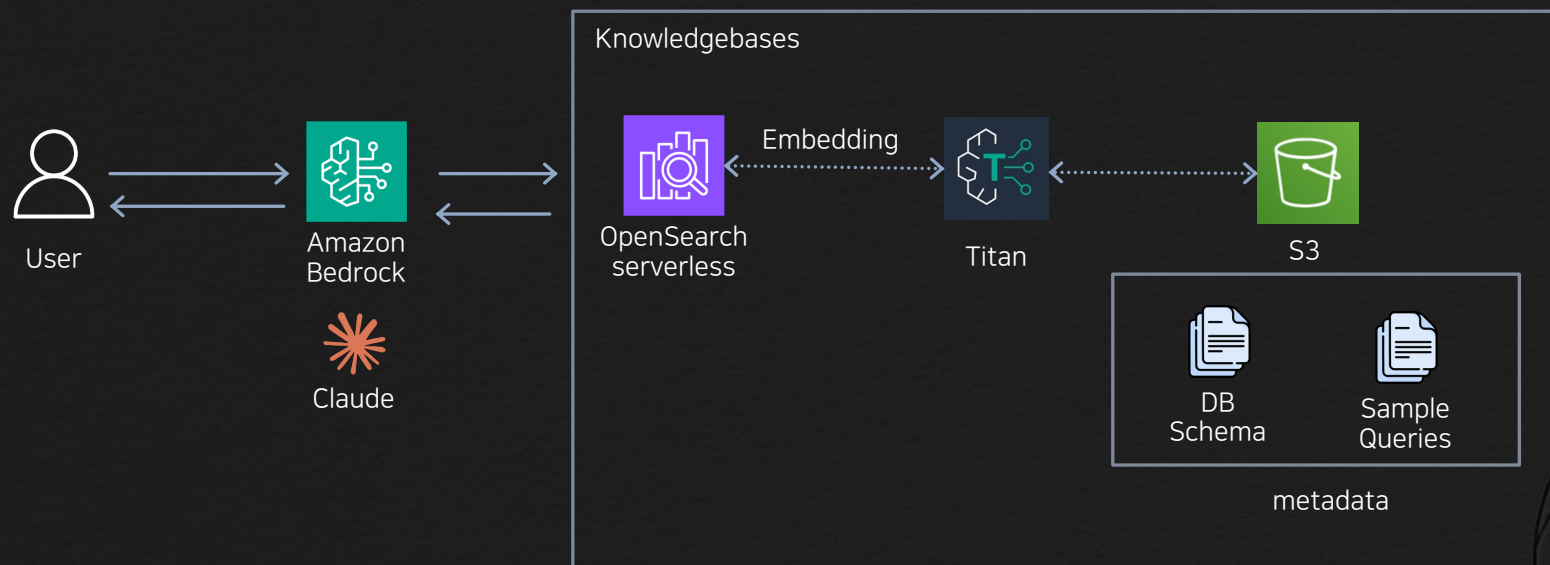


Amazon OpenSearch Serverless

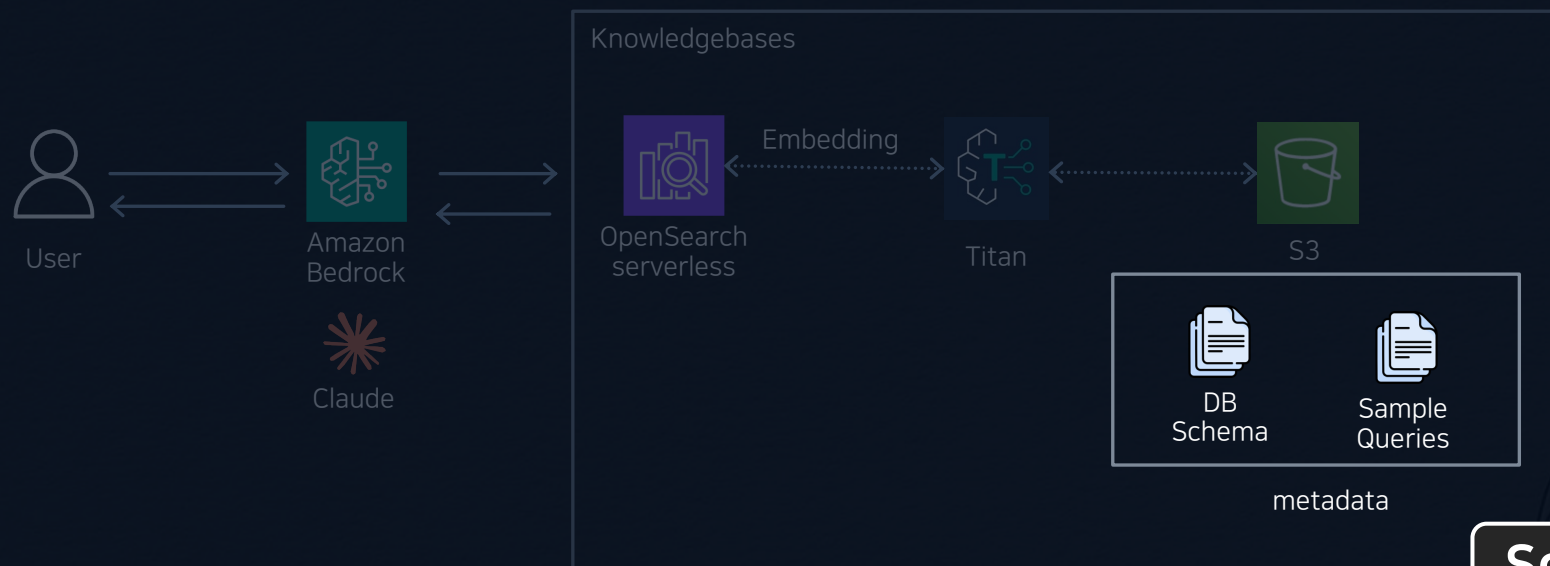
- 스키마 정보 및 샘플 쿼리 인덱싱
- 하이브리드 서치



AWS Bedrock + KnowledgeBases



AWS Bedrock + KnowledgeBases



Schema Linking

Schema Linking



자연어 질의 인식

- "T 우주 패스 with Netflix" 상품 계약 조회



속성 및 개념 추출

- 질의 내 주요 속성 엔티티 파악 (상품, 구독, 계약)



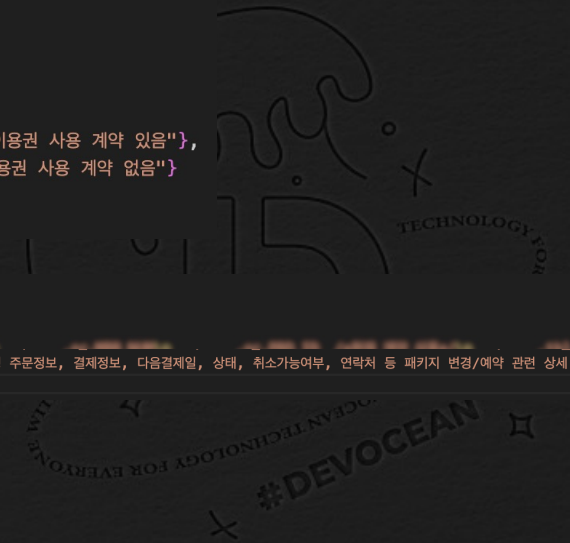
스키마 연결

- PRD_PRD.PRD_NM, ORD_CTR.PRD_ID...

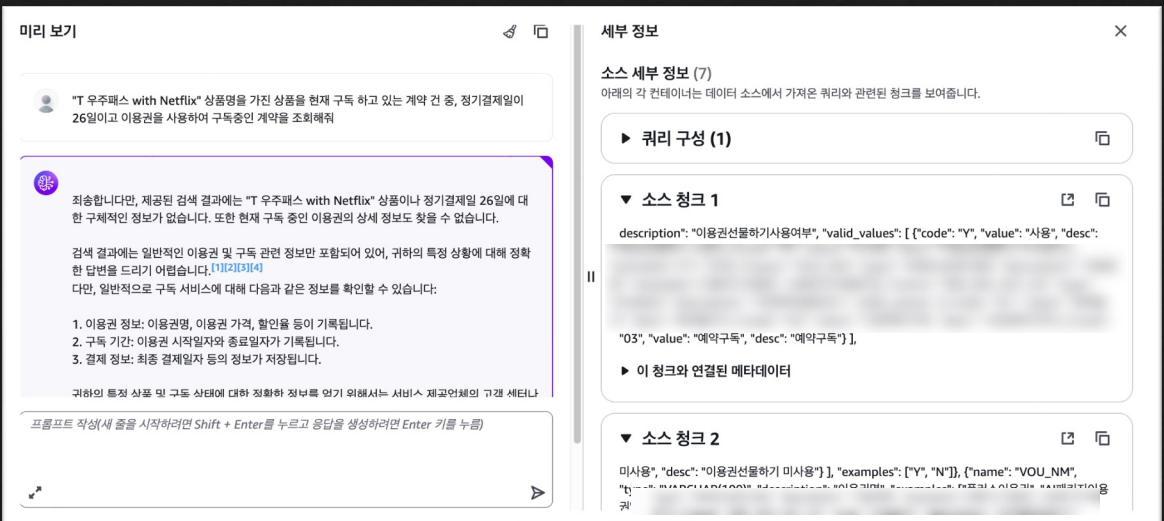
- LLM을 통해 Table, Column, Query 에 대한 description 증강

- 공통코드 데이터 추출하여 Column 과 매핑

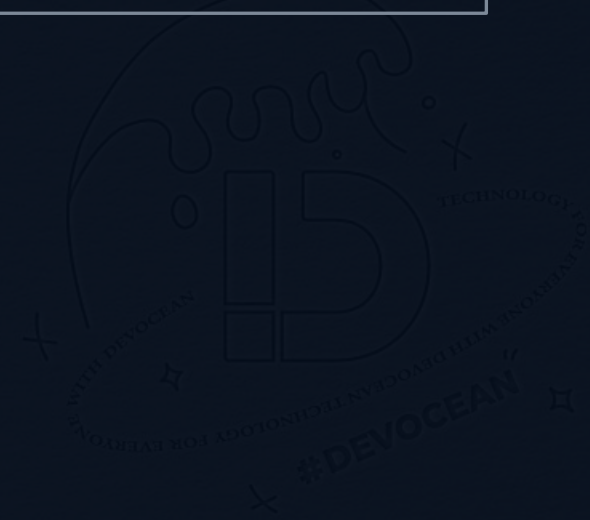
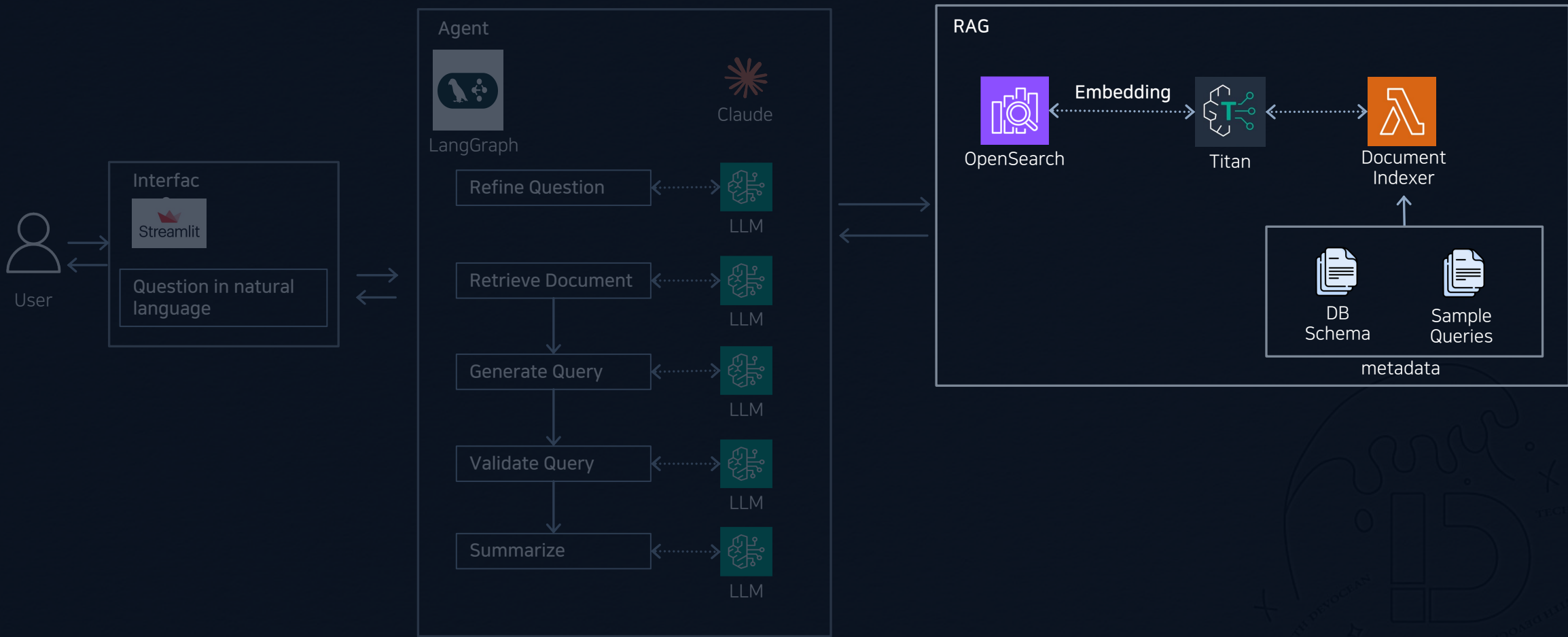
```
"table_name": "ord",
"description": "계약 정보를 담고 있는 테이블",
"columns": [
  {
    "name": "CTR_NUM",
    "type": "DECIMAL(10,0)",
    "description": "계약번호",
    "valid_values": [
      {"code": "Y", "value": "Yes", "desc": "이용권 사용 계약 있음"},
      {"code": "N", "value": "No", "desc": "이용권 사용 계약 없음"}
    ]
  },
  {
    "name": "PRD_ID",
    "type": "VARCHAR(10)",
    "description": "상품 ID",
    "valid_values": [
      {"code": "P001", "value": "P001", "desc": "상품 ID 001"},
      {"code": "P002", "value": "P002", "desc": "상품 ID 002"}
    ]
  }
],
"sample_queries": [
  {
    "query": "PRD_PRD.PRD_NM, ORD_CTR.PRD_ID...",
    "description": "/// 패키지(구독) 계약번호로 현재 계약 및 최근 패키지변경 주문정보, 결제정보, 다음결제일, 상태, 취소가능여부, 연락처 등 패키지 변경/예약 관련 상세 계약"
```



단순한 질의는 OK, 질의가 복잡해지면 아쉬운 결과



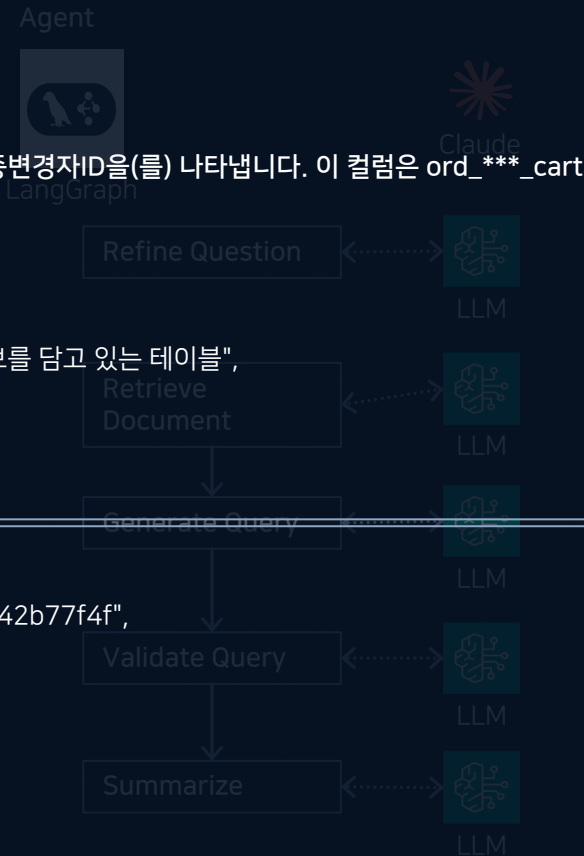
RAG(Retrieval Argument Generation)



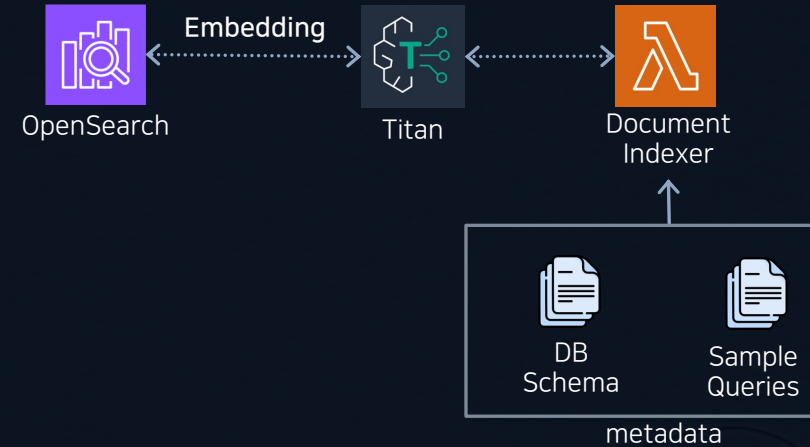
Vector Embedding

```
1. schema index document{
  "_index": "chg-text2sql-schema-index",
  "_id": "58681a10-75c2-4696-8c61-1695994def5b",
  "vector_field": [
    0.0005889780586585402,
    -0.0006467749481089413,
    // (생략) //
  ],
  "text": "컬럼명: AUDIT_ID. 이 컬럼은 최종변경자ID을(를) 나타냅니다. 이 컬럼은 ord_***_cart 테이블에 속해 있습니다.",
  "metadata": {
    "type": "schema",
    "db_name": "general_system",
    "table_name": "ord_***_cart",
    "table_description": "비로그인 카트 정보를 담고 있는 테이블",
    "column_name": "AUDIT_ID",
    "column_type": "VARCHAR(20)",
    ...
  }
}
```

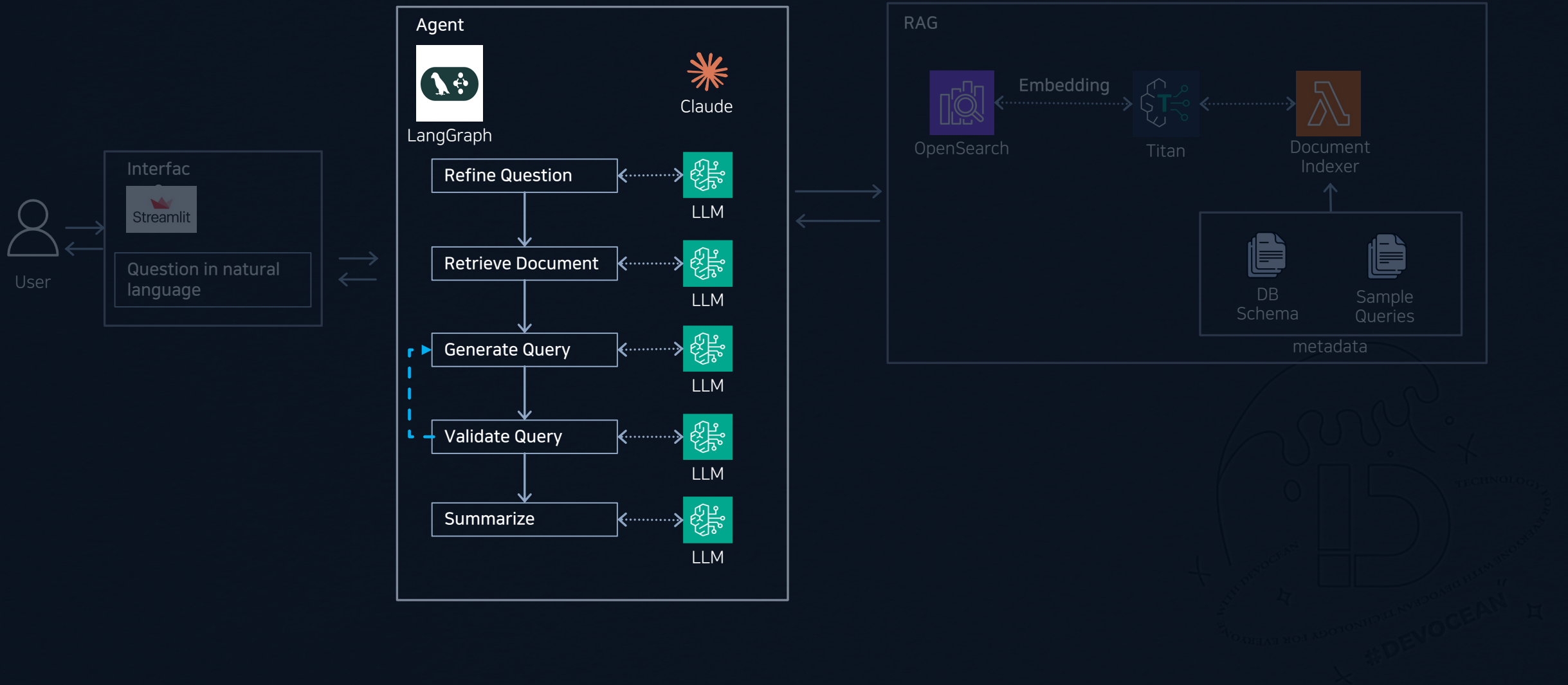
```
2. query index document{
  "_index": "old-text2sql-query-index",
  "_id": "ddcfbda5-cbcf-4ce4-8418-254a42b77f4f",
  "_version": 1,
  "_score": 0,
  "_source": {
    "vector_field": [
      0.011821222491562366,
      0.002364244544878602,
      (생략)
    ],
    "text": "SQL 쿼리: SELECT ord.ord_*** oc, rsv_***_yn, ***_dtmWFROMWn WHERE CTR_NUM = #{***Num}Wn AND EFF_END_DTM = '99991231235959'Wn설명: // 계약(ORD_***)의 예약존재여부(RSV_***_YN), 변경일시를 계약번호 기준으로 가져옵니다.Wn",
    "metadata": {}
  },
  ...
}
```



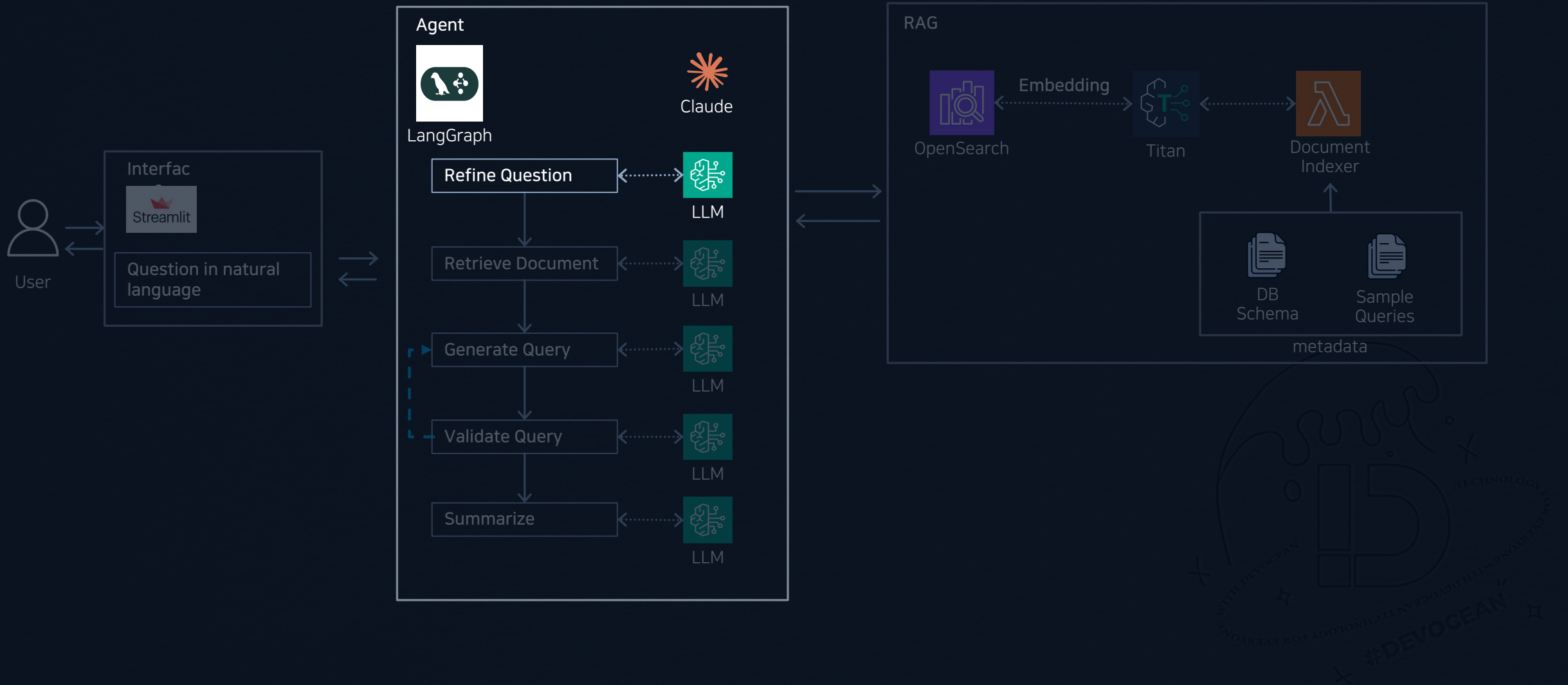
RAG



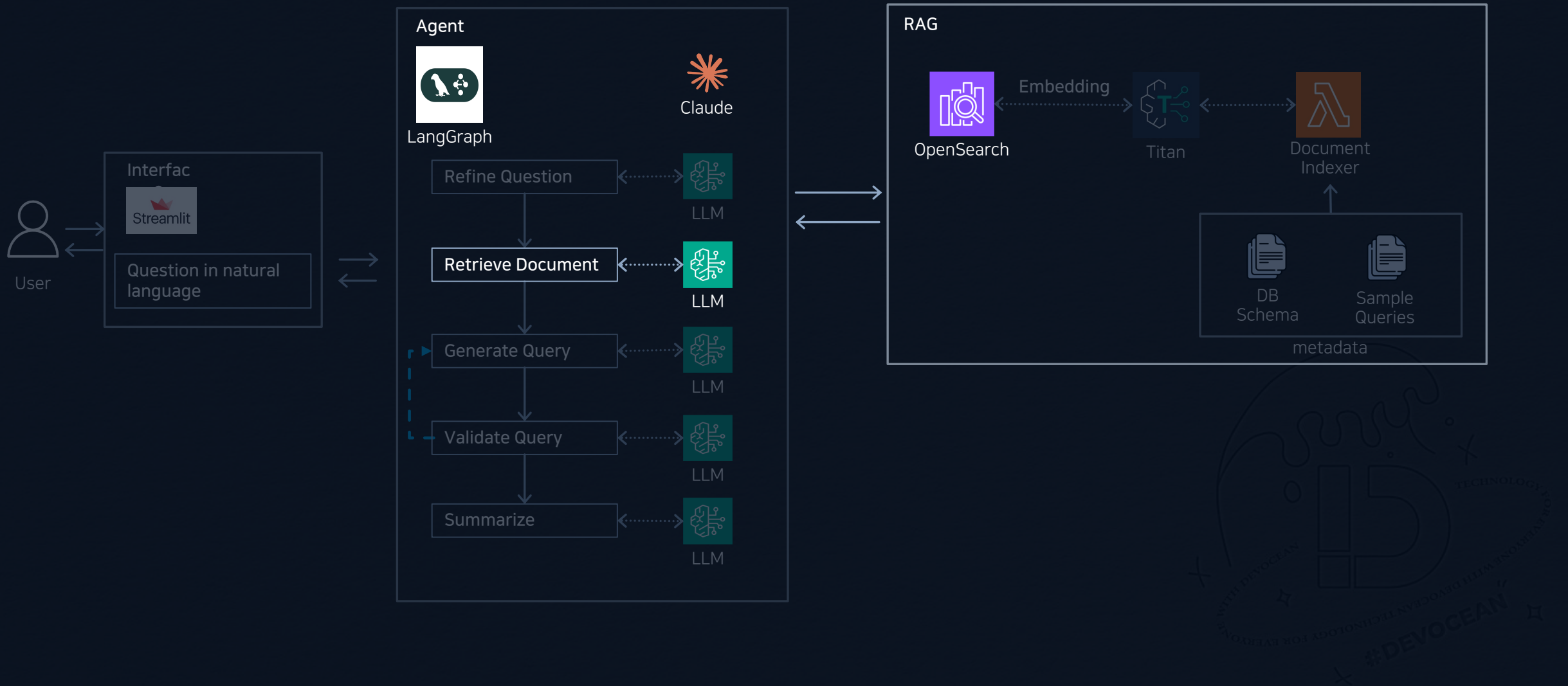
Agent With LangGraph



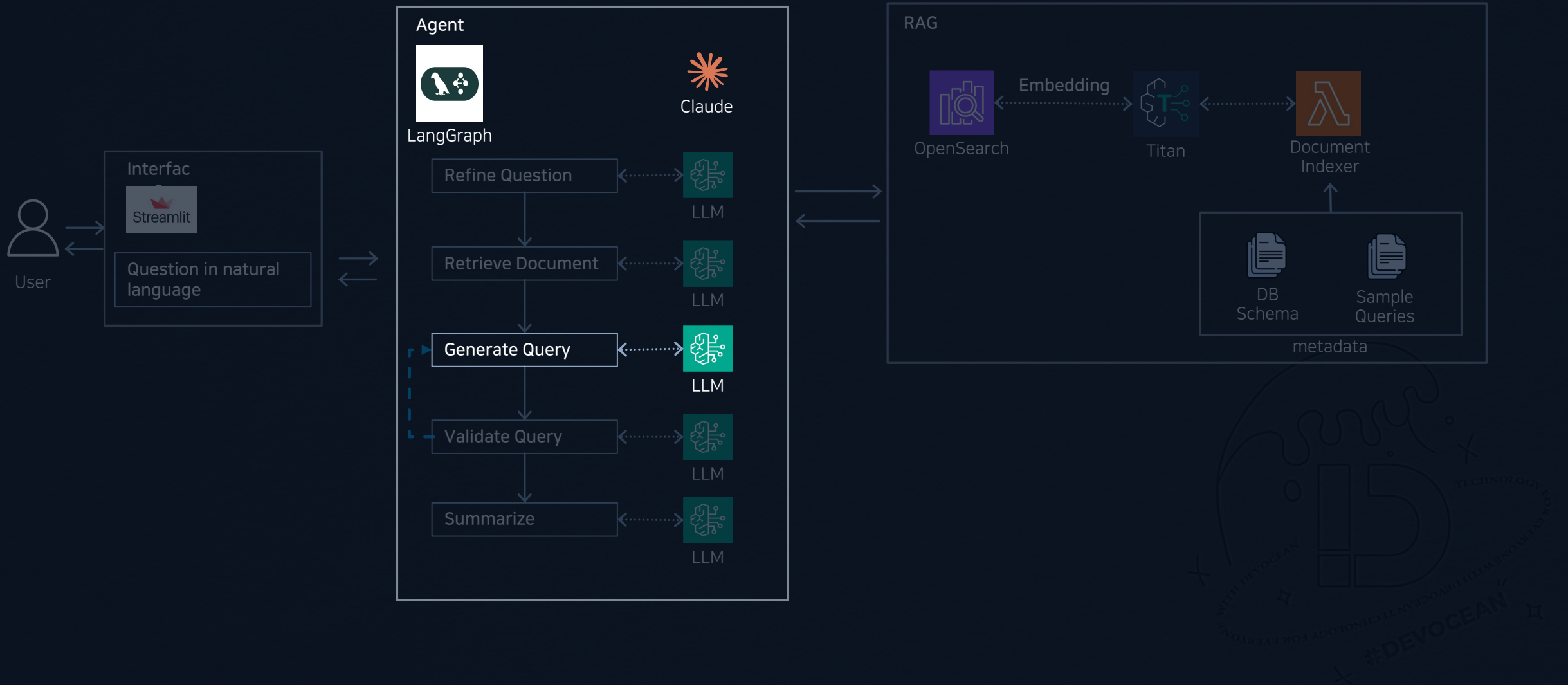
Agent With LangGraph



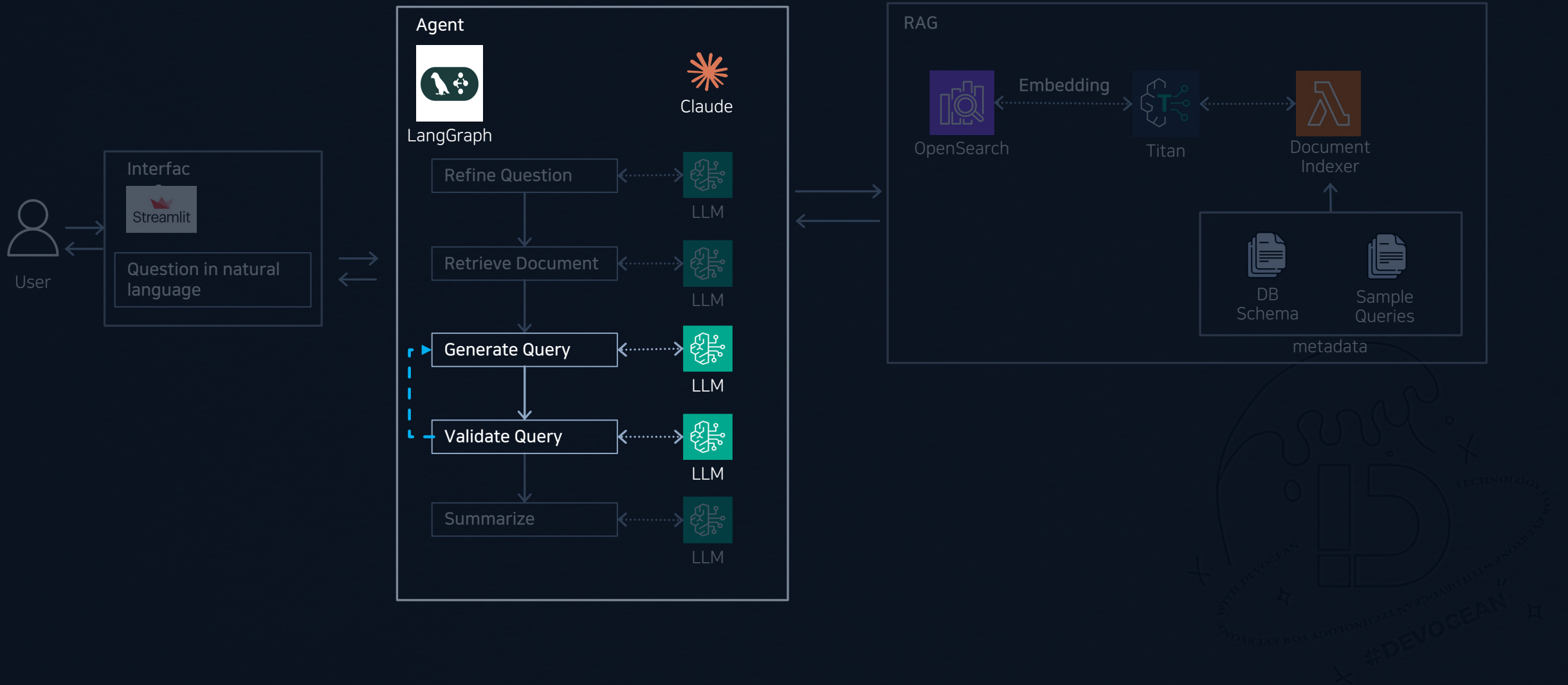
Agent With LangGraph



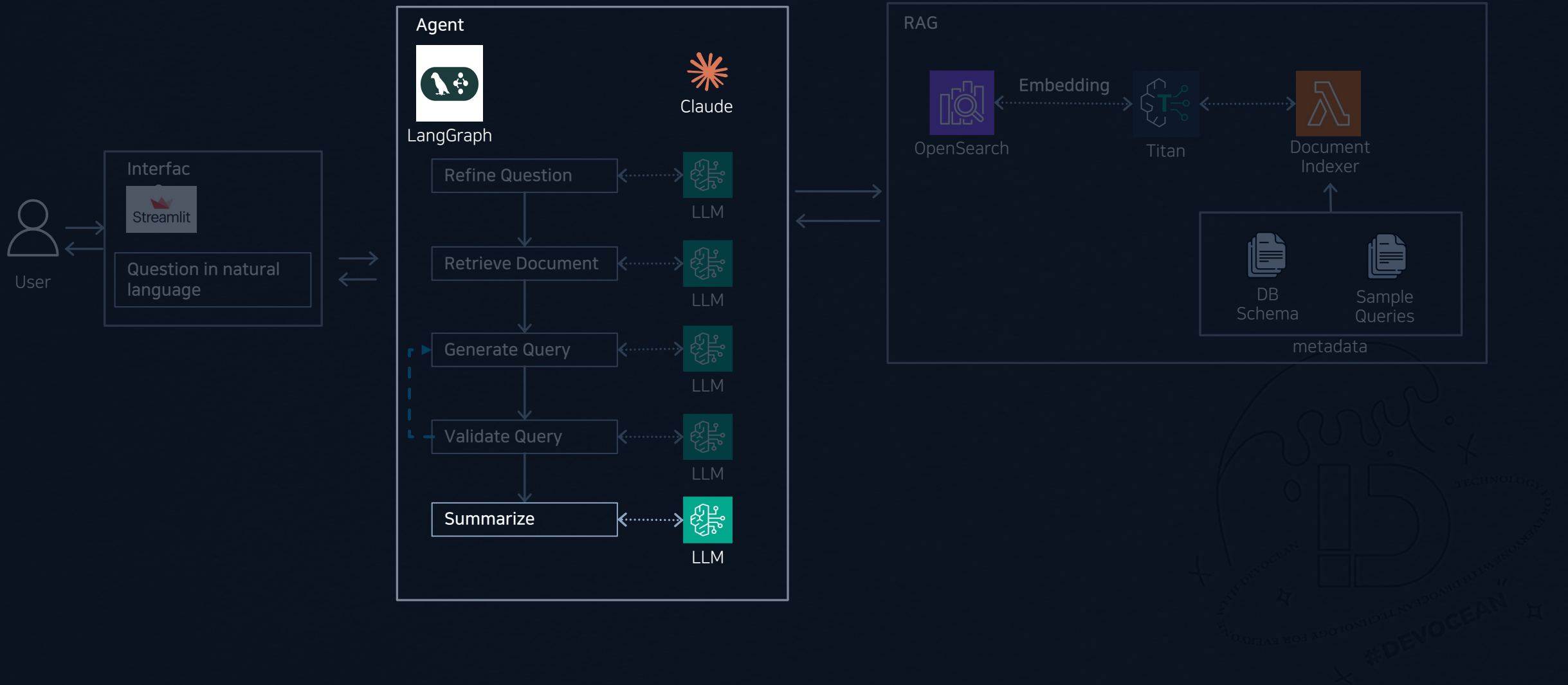
Agent With LangGraph



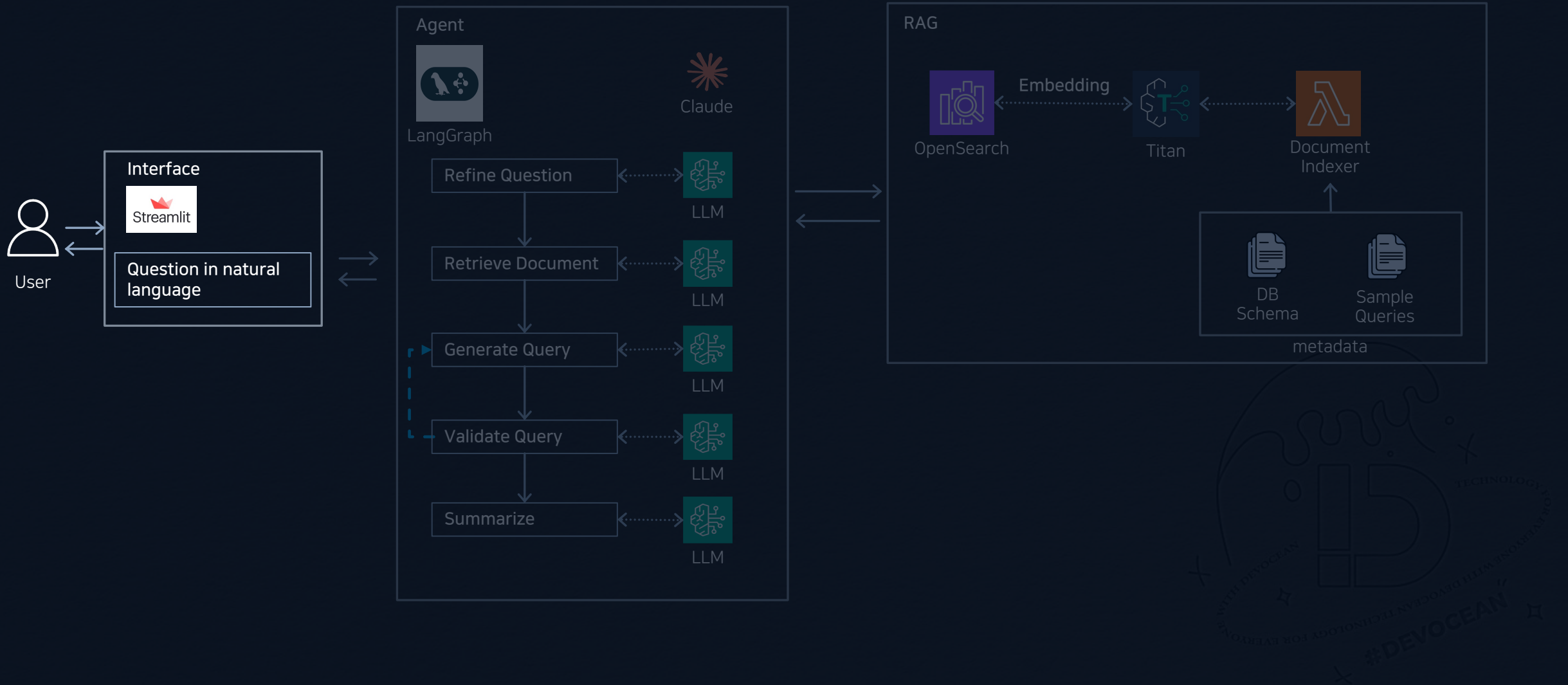
Agent With LangGraph



Agent With LangGraph



시스템 구성



보완된 결과

아직 해소되지 않은 문제들

Text-to-SQL with Feedback

SQL 질의

SQL 생성을 위한 질의를 입력하세요:

"배달의 민족" 상품명을 가진 상품을 구성 상품으로 가입중인 계약을 조회하고 "배달의 민족" 상품을 제외하고 구성상품을 1개 이상 추가로 가입하고 있는 계약을 조회해줘

SQL 생성

요청 분석 | 참조 문서 | **생성된 SQL** | 피드백

생성된 SQL

사용자의 요구사항을 반영하여 다음과 같은 SQL 쿼리를 작성했습니다:

```
```sql
SELECT
 ORD_CTR.CTR_NUM,
 ORD_CTR.CTR_ST_CD,
 ORD_CTR.EFF_END_DTM AS CTR_EFF_END_DTM,
 ORD_CTR_COMP.CTR_COMP_NUM,
 ORD_CTR_COMP.CTR_COMP_ST_CD,
 ORD_CTR_COMP.EFF_END_DTM AS COMP_EFF_END_DTM,
 PPR.PRD_NM
FROM ORD_CTR
JOIN ORD_CTR_COMP ON ORD_CTR.CTR_NUM = ORD_CTR_COMP.CTR_NUM
JOIN PPR ON ORD_CTR_COMP.CTR_COMP_NUM = PPR.CTR_COMP_NUM
GROUP BY
 ORD_CTR.CTR_NUM
HAVING
 COUNT(CASE WHEN PPR.PRD_NM <> '배달의 민족' THEN 1 END) > 0
```
```

이 쿼리는 다음과 같은 작업을 수행합니다:

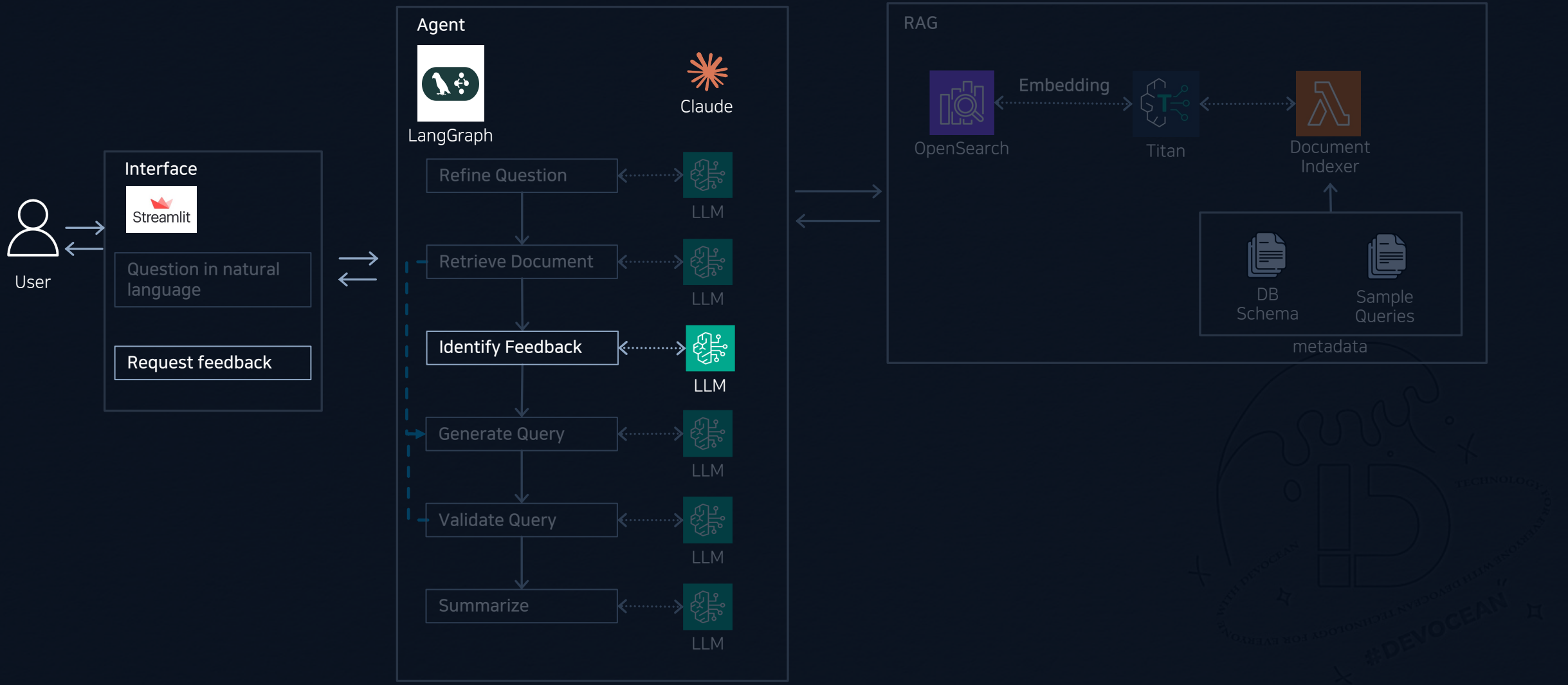
1. 'ORD.ORD_CTR', 'ORD.ORD_CTR_COMP', 'PRD.PRD_PRD' 테이블을 조인하여 계약, 계약구성, 상품 정보를 가져옵니다.
2. 'WHERE' 절에서 계약 상태가 '사용 중' 이고, 계약구성 상태로 '사용 중'인 경우만 선택합니다.
3. 'WHERE' 절에서 상품명에 '배달의 민족'이 포함된 상품만 선택합니다.
4. 'GROUP BY' 절에서 계약 ID를 그룹화합니다.
5. 'HAVING' 절에서 '배달의 민족' 상품 외에 추가로 1개 이상의 상품이 가입된 계약 ID를 필터링합니다.

이 쿼리는 "배달의 민족" 상품을 구성 상품으로 가지고 있으면서, 추가로 다른 상품을 1개 이상 가입하고 있는 계약 ID를 반환합니다.

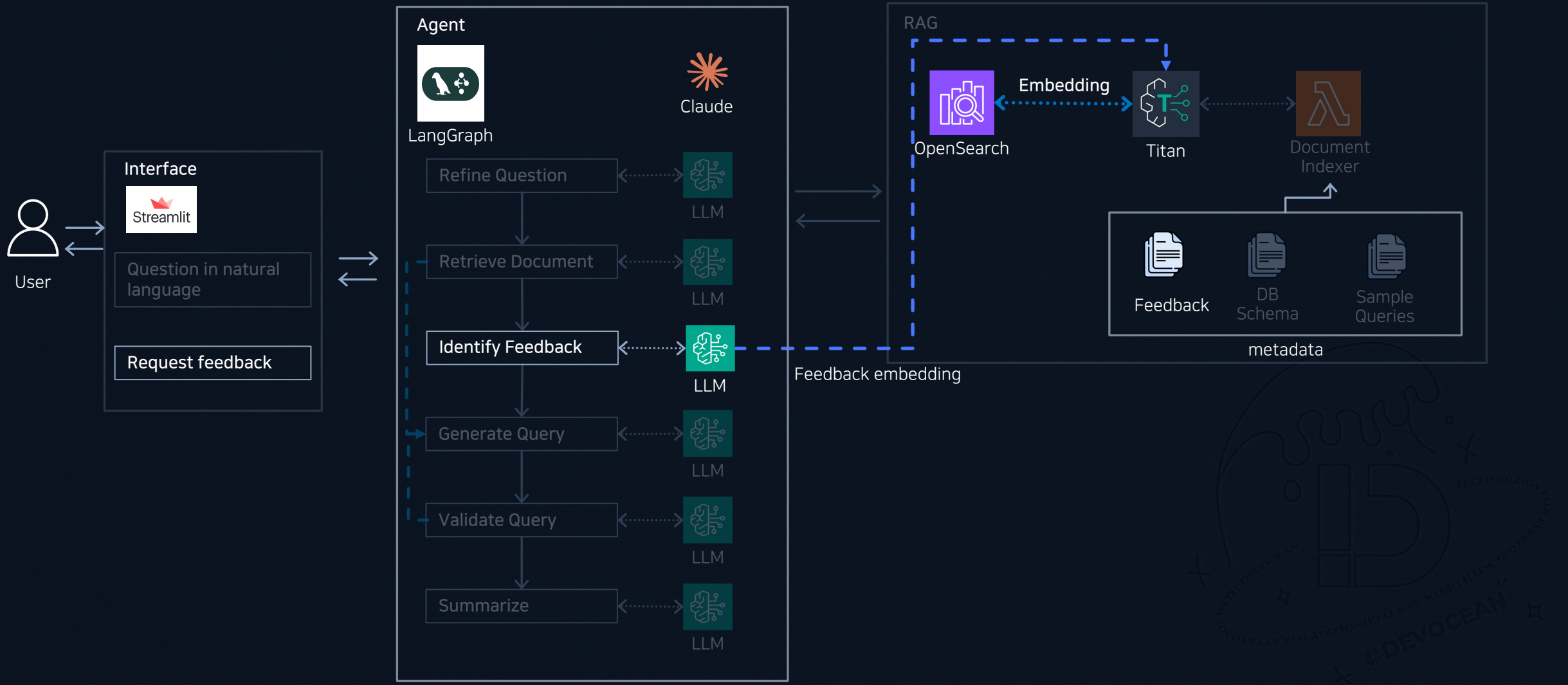
주의사항:

- 참고 문서에는 '공유되지 않은 계약'을 판단할 수 있는 필드가 없어서 이 조건은 제외하였습니다.
- 참고 문서에는 '배달의 민족' 상품을 이용하고 있는 사용자'의 계약을 판단할 수 있는 필드가 없어서 이 조건은 제외하였습니다.

Feedback 기반 데이터 증강



Feedback 기반 데이터 증강



Feedback 기반 데이터 증강

Feedback을 통한 강화학습으로 결과 향상

요청 분석

질조 문서

생성된 SQL

피드백

SQL 개선을 위한 피드백

피드백 유형을 선택하세요:

수정

구체적인 피드백을 입력하세요:

구성상품을 JOIN 할때는 PRD_ID가 아닌 ORD_CTR_COMP.COMP_PRD_ID로 조인을 해야돼

피드백 적용

피드백 지우기

피드백 히스토리

피드백 1: 수정

내용: 구성상품을 JOIN 할때는 PRD_ID가 아닌 ORD_CTR_COMP.COMP_PRD_ID로 조인을 해야돼

사용자의 피드백을 반영하여 다음과 같이 SQL 쿼리를 수정하였습니다:

```
```sql
SELECT
 ORD_CTR.CTR_NUM,
 ORD_CTR.CTR_ST_CD,
 ORD_CTR.EFF_END_DTM AS CTR_EFF_END_DTM,

 ...

 AND ORD_CTR_COMP.CTR_COMP_ST_CD = 'AC'
 AND ORD_CTR_COMP.EFF_END_DTM = '99991231235959'
 AND PPR.PRД_NM LIKE '%배달의 민족%'
GROUP BY
 ORD_CTR.CTR_NUM
HAVING
 COUNT(CASE WHEN PPR.PRД_NM <> '%배달의 민족%' THEN 1 END) > 0
...
```
```

변경 사항:

1. 'ORD_ODR_COMP_PRD' 테이블을 주가르 조인하여 'COMP_PRD_ID' 컬럼을 사용하도록 수정하였습니다.

2. 'ORD_ODR_COMP_PRD,COMP_PRD_ID' 와 'PPR,PRD_ID'를 조인하여 상품 정보를 가져오도록 변경하였습니다.

이 쿼리는 "배달의 민족" 상품을 구입 상품으로 가지고 있으면서, 주가르 다른 상품을 1개 이상 가입하고 있는 계약 ID를 반환합니다.

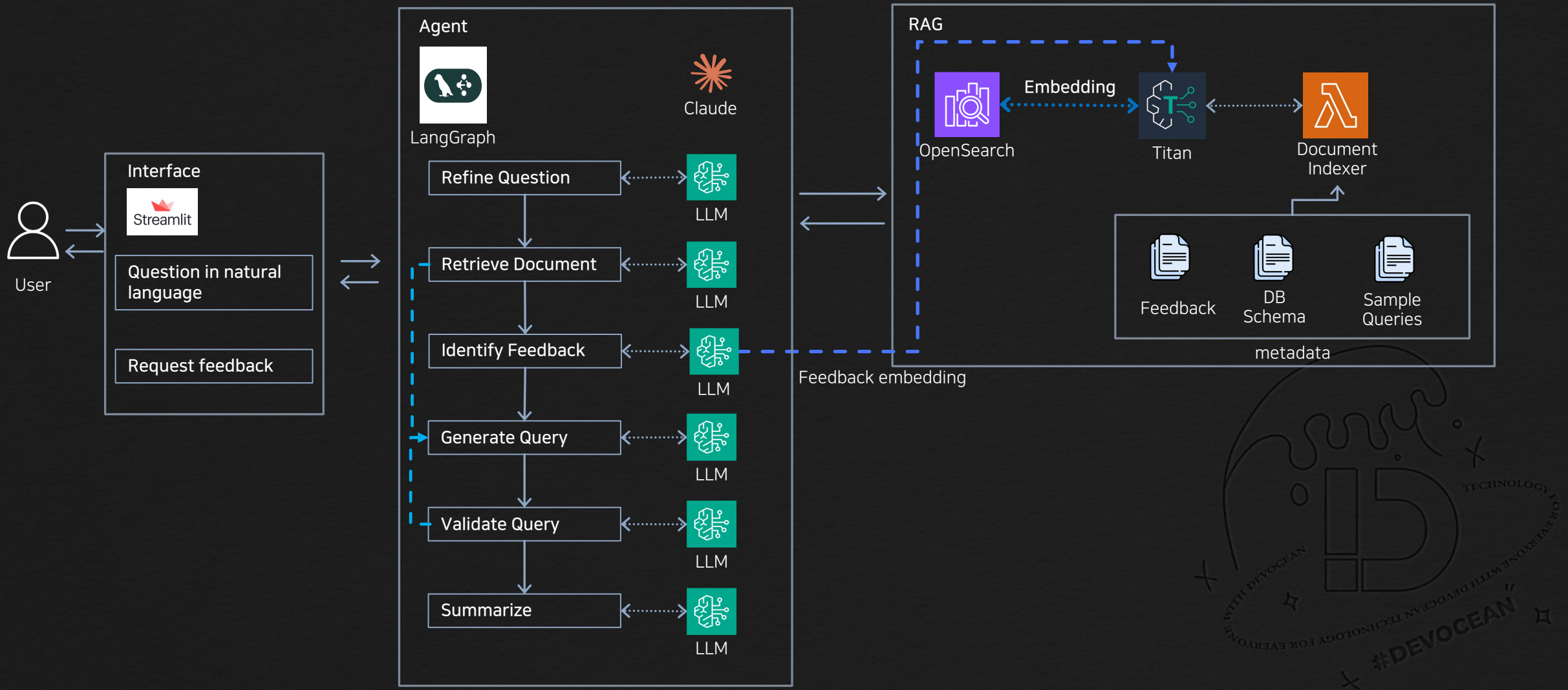
주의사항:

- 옳고 못서에는 '공유되지 않은 계약'을 판단할 수 있는 컬럼이 없어서 이 조건은 제외하였습니다.

- 옳고 못서에는 '배달의 민족 상품을 이용하고 있는 사용자'의 계약을 판단할 수 있는 컬럼이 없어서 이 조건은 제외하였습니다.

피드백 결과 저장

Architecture



Lesson Learned

비즈니스 도메인

동일 의미 다양한 용어 사용 및 유사한 명칭의 컬럼 다수 존재
용어사전 활용

결과 정확도

하나의 질의에도 여러개의 유사쿼리로 응답오는 상황
Reranker를 활용하여 순위

변화 관리

DB스키마 지속적인 현행화 필요
파이프라인 을 구성하여 강화

Hallucination

존재하지 않는 테이블 및 컬럼을 참조하여 결과에 반영
프롬프트를 강화하고, 피드백 기능을 활용



활용 방안



구독 챗봇 연계

질의 답변 쿼리 생성
통합테스트 데이터 생성
정산 데이터 생성



통계 데이터 활용

정형화 된 통계데이터 생성
매출 데이터
가입자 추이



비즈니스 로직 설계

신규 기능 초안 생성
검증 쿼리 생성



감사합니다





Agentic Ops 도입사례

(AI와 함께하는 AWS 클라우드 운영)

이중호

구독플랫폼개발팀 Manager

SK Telecom



목차

01 Problem Statement

02 Why AWS Bedrock?

03 Use Cases of Bedrock

04 Retrospective

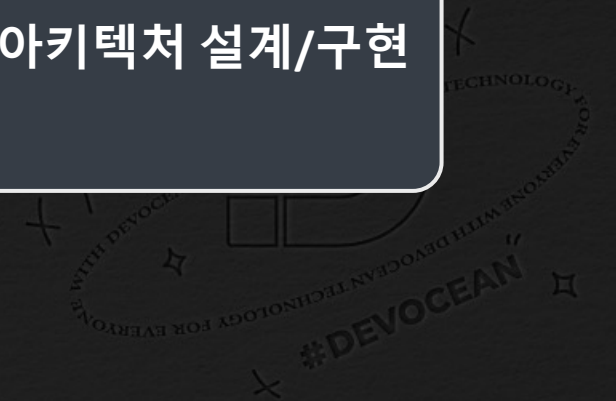
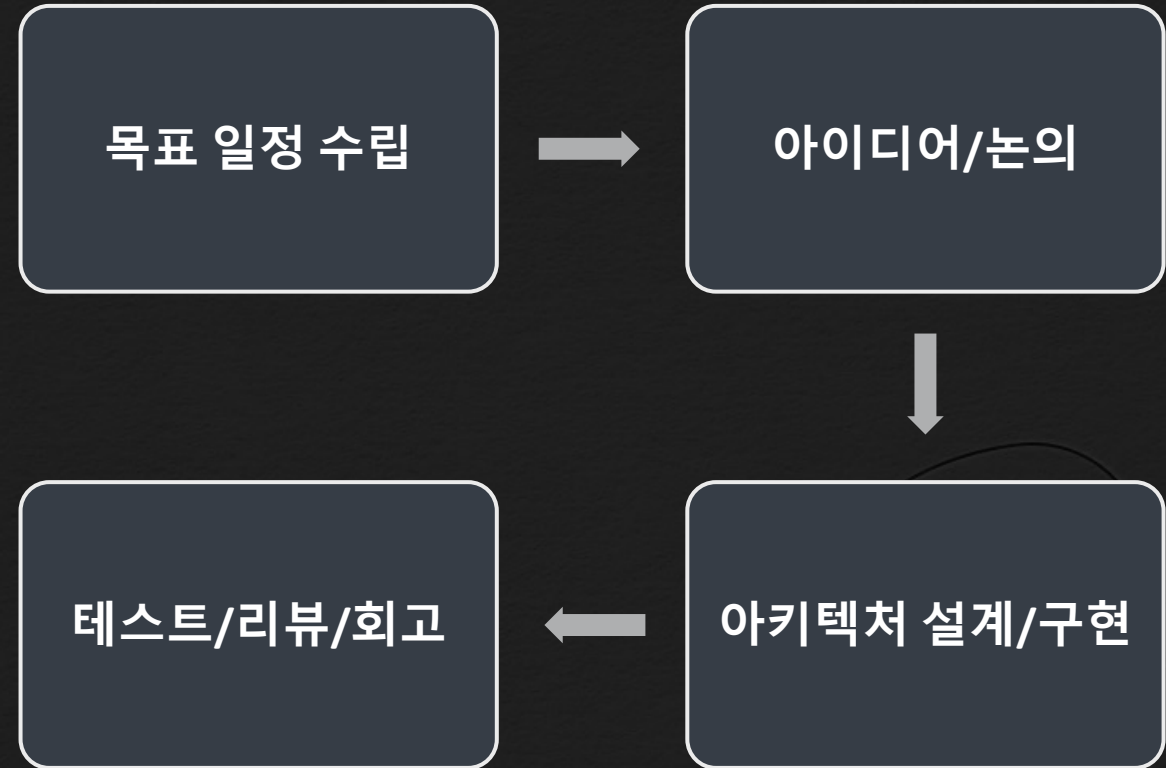


01 Problem Statement



이상적인 Project 수행 방식

명확한 목표, 주도적인 업무 수행 방식



Cloud 관리자의 실제 업무

요청에 의한 이슈 처리 및 문의 대응 비중이 높음



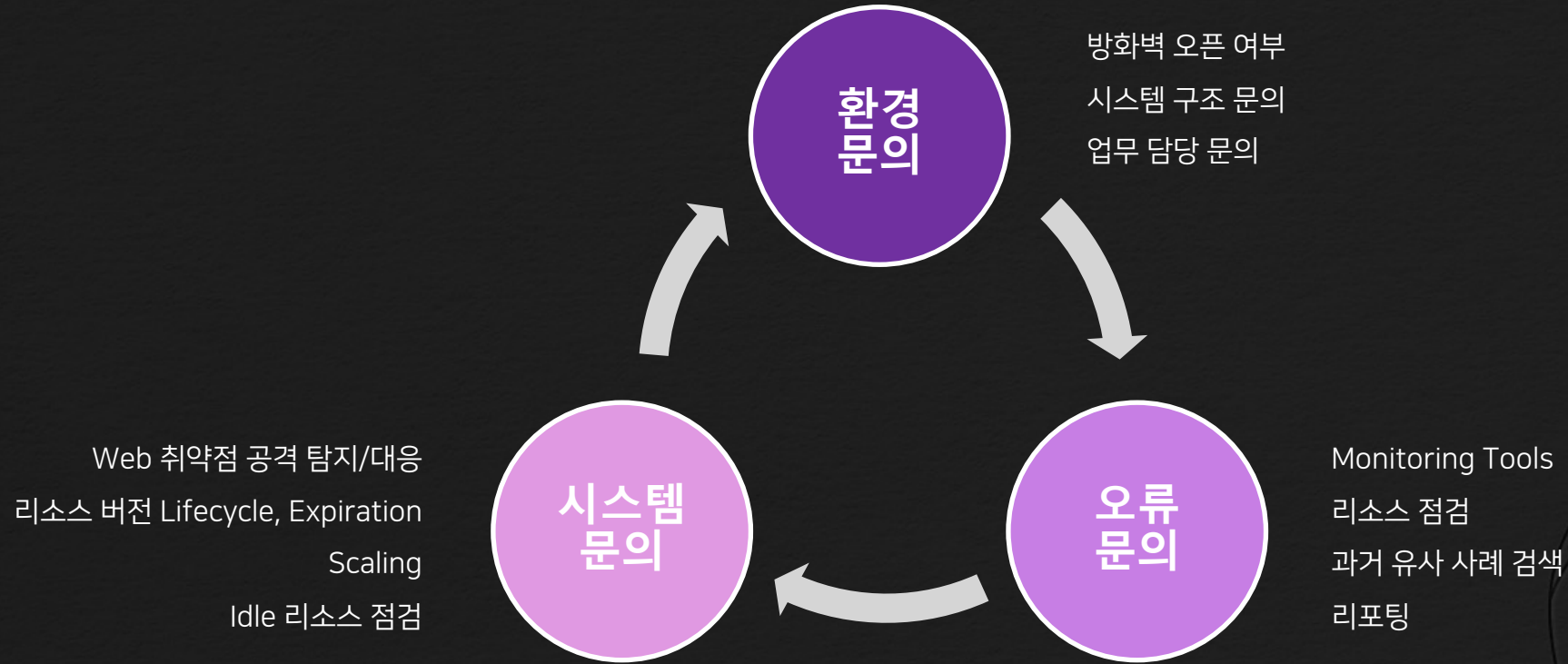
Cloud 관리자

외부 요청에 의한 업무가 대부분을 차지해서
주도적으로 업무할 시간이 부족



운영 업무의 패턴

소모적이고 반복적인 업무의 비중 높음



02 Why Bedrock?



Amazon Bedrock을 선택한 이유

AWS 사용 시 생성형 AI 애플리케이션을 위한 가장 쉬운 플랫폼

Meta

amazon

ANTHROPIC

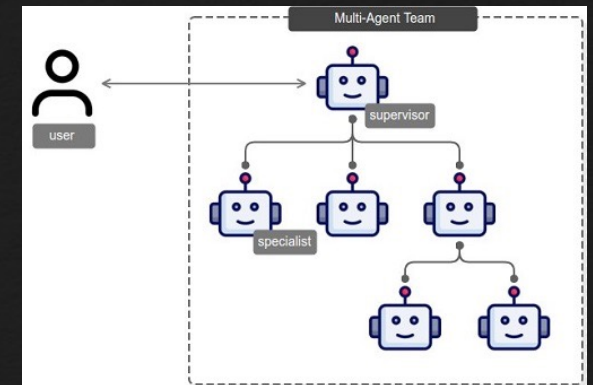
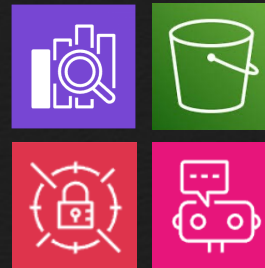
deepseek

```
A developer's guide to Bedrock's new Converse API

const conversation = [{
  role: "user",
  content: [{ text: userMessage }],
}];

const command = new ConverseCommand({
  modelId: "anthropic.claude-3-haiku-20240307-v1:0",
  messages: conversation
});

const response = await client.send(command);
```



1

폭 넓은 모델 선택

2

단일 API 사용

3

AWS 서비스와
손쉬운 통합

4

에이전트로 손쉬운
워크플로우 구현

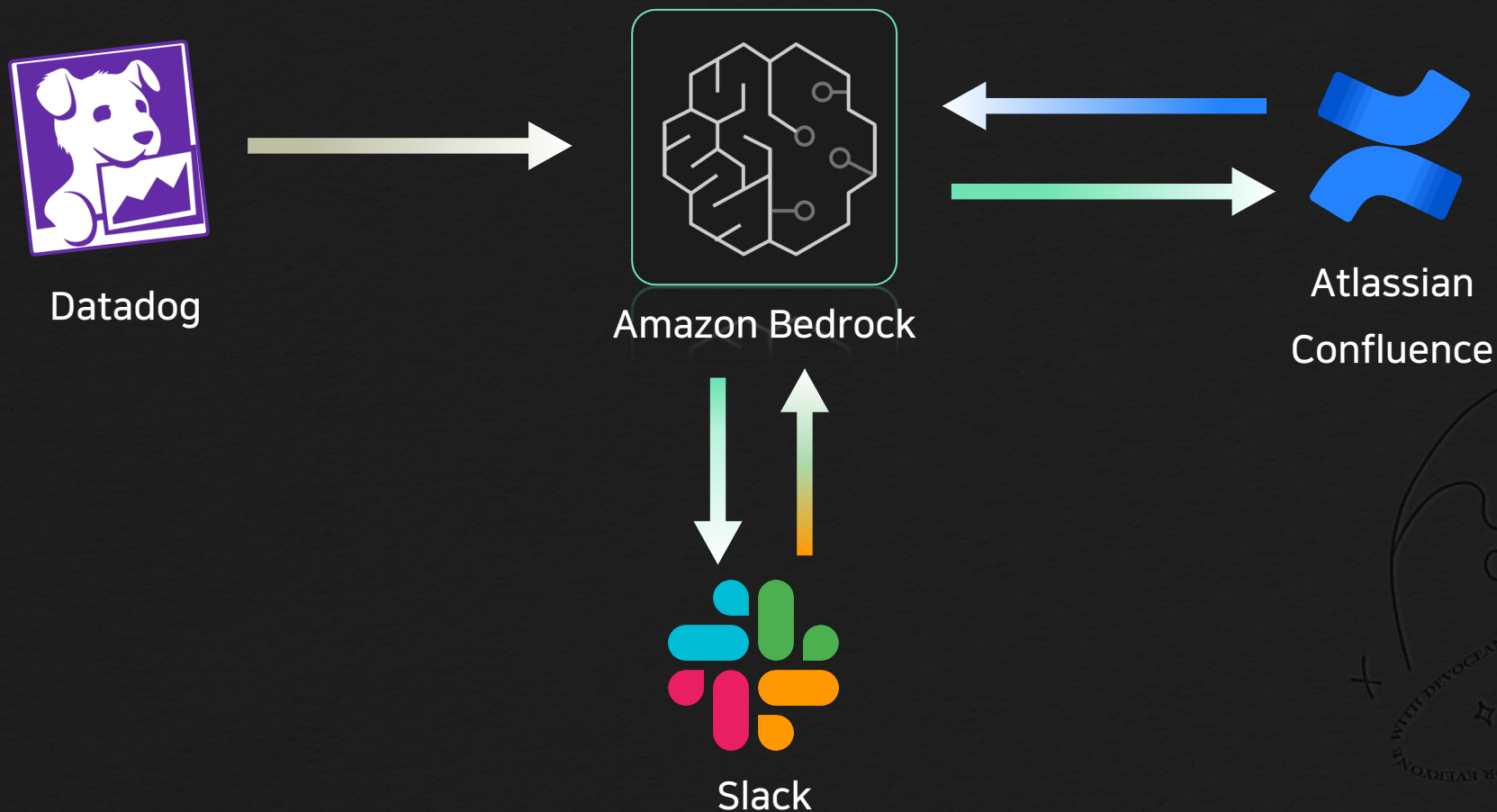


03 Use Cases of Bedrock



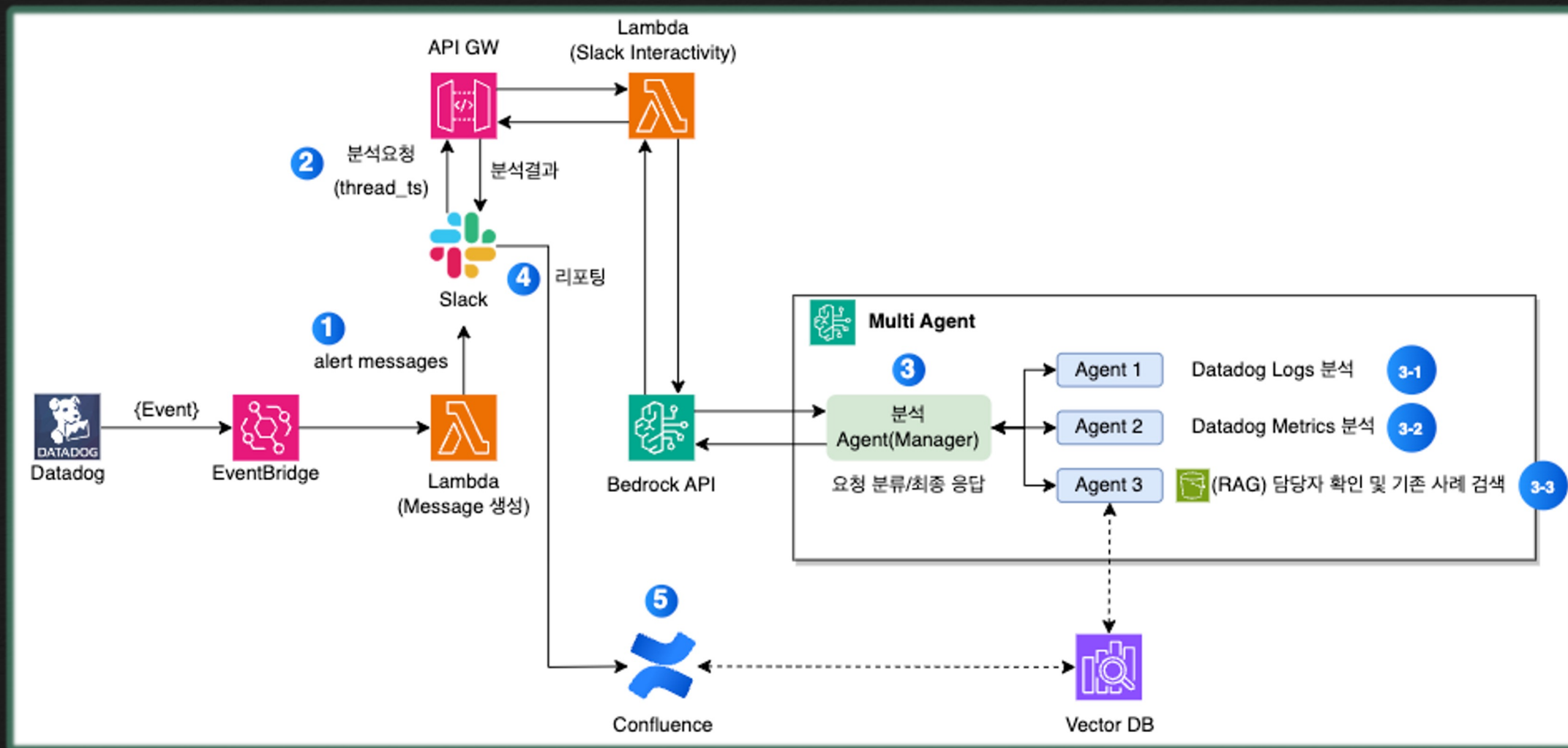
Case1. LLM 기반 이슈 관리 Process

업무에 주로 활용되는 협업도구, 모니터링 툴 그리고 Amazon Bedrock의 결합



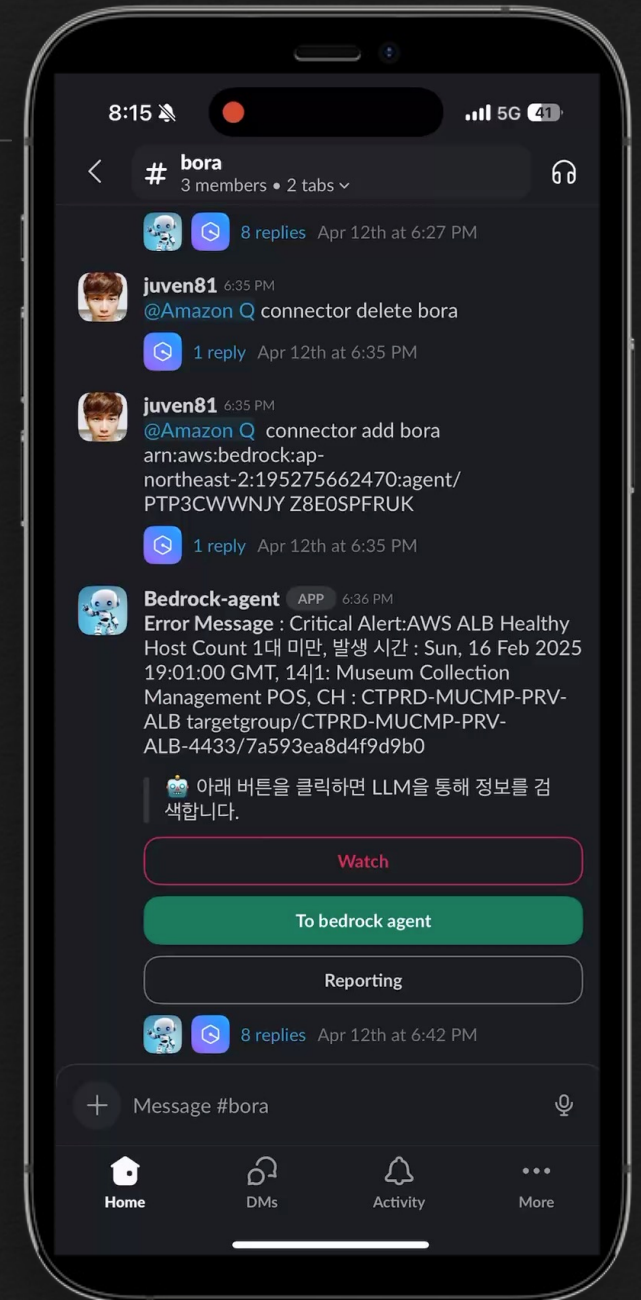
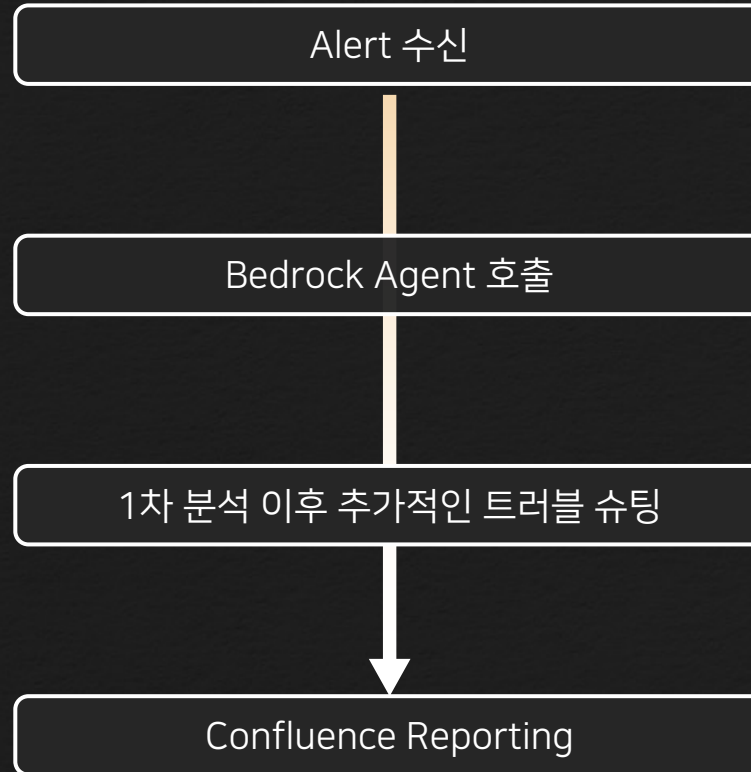
Architecture: Datadog– Slack – Bedrock – Confluence 연동

Multi Agent 기반으로 더욱 다양하고 신속한 분석, 그리고 사례에 기반한 정확한 해결책 제시 가능



운영자 Interactivity 최소화

Slack의 Thread에서 Bedrock Agent에게 이슈 분석 요청

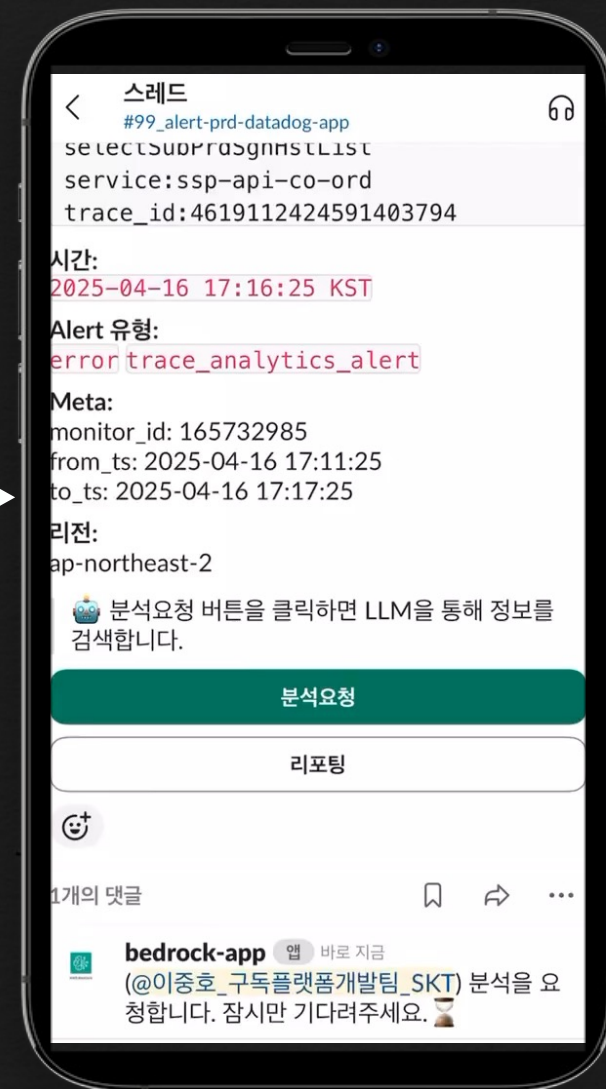
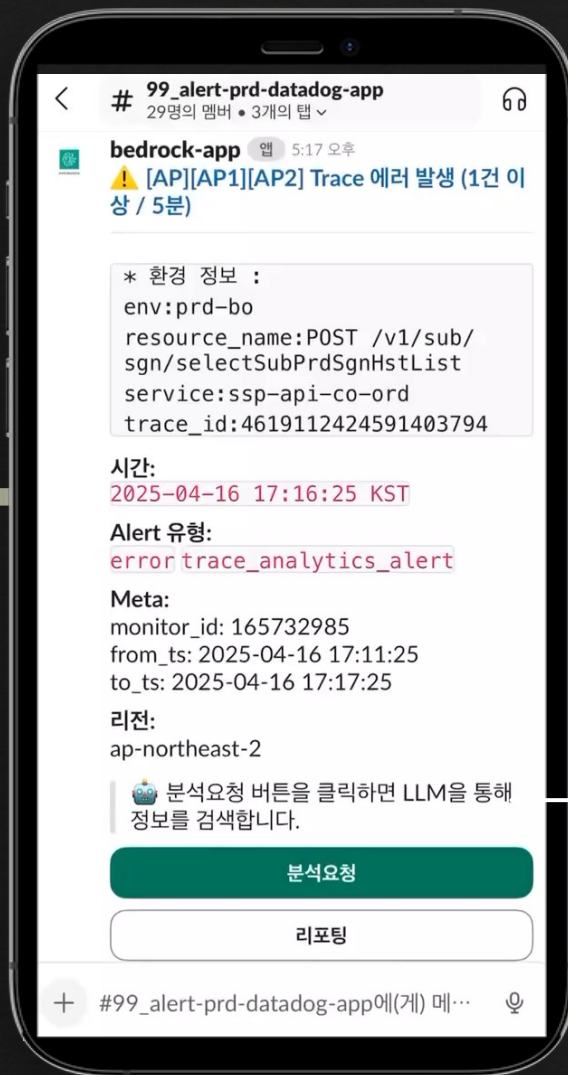


Multi Agent 활용

Thread Request 후 Bedrock API > Multi Agent를 통해 3가지 Type의 Action 수행 후 최종 응답



Data Dog



Confluence 자동 Reporting

Slack의 Thread를 Prompt로 요약 후 Confluence 게시하기

2025/04/07 16:05 (ssp-bff-fo)

작성자 : 이종호 구독플랫폼개발팀 - 2025 04 07,

SmartBee 요약

새로고침 아이콘을 클릭하여 요약하세요 ↺ ⓘ

원문 링크 표기

(Bedrock에 의해 요약된 내용입니다.)

관련 슬랙 메시지: <https://newssphq.slack.com/archives/C08HAELFPE2/p1743815998425119>

이슈 타이틀, 원인, 해결여부

| 항목 | 내용 |
|---------------|--|
| 1. 문제 | ssp-bff-fo 서비스에서 ModelMapper 오류 발생 |
| 2. 원인 | ModelMapper가 java.util.ArrayList를 java.util.List로 변환하는 데 실패했습니다. 오류 코드는 COM_9999 (Internal Error)입니다. 4월 4일 17:20부터 평소보다 많이 발생하기 시작했으며, 어플리케이션 변경 없이 특정 시점부터 발생했습니다. |
| 3. 해결여부 | 미해결 (추가 확인 중) |
| 4. 참여자 및 조치사항 | |
| 4.1 | FE팀 (서비스 담당자) 조치사항 <ul style="list-style-type: none">- 오류 발생 상황 모니터링 및 보고- 서비스 전반적인 영향 없음을 확인- ModelMapper 소스와 타겟이 동일함에도 간헐적으로 발생하는 현상에 대해 추가 확인 진행 중 |
| 4.2 | BE팀 조치사항 <ul style="list-style-type: none">- 오류 내용 확인 및 코드 검토 필요성 언급- 서비스 영향 없음 확인 |
| 4.3 | BE팀 조치사항 <ul style="list-style-type: none">- 가입가능여부조회 호출과 관련된 오류 분석- 자동인증 제휴사 응답 문제 여부 확인- 한컴독스 관련 SKU 정보 수정- VoC 모니터링 계획 수립 |
| 4.4 | 김플랫폼팀 (서비스 담당자) 조치사항 <ul style="list-style-type: none">- 별도 조치사항 없음 (대화 내용에 구체적인 조치 언급 없음) |

참여자들과 조치사항 요약 게시

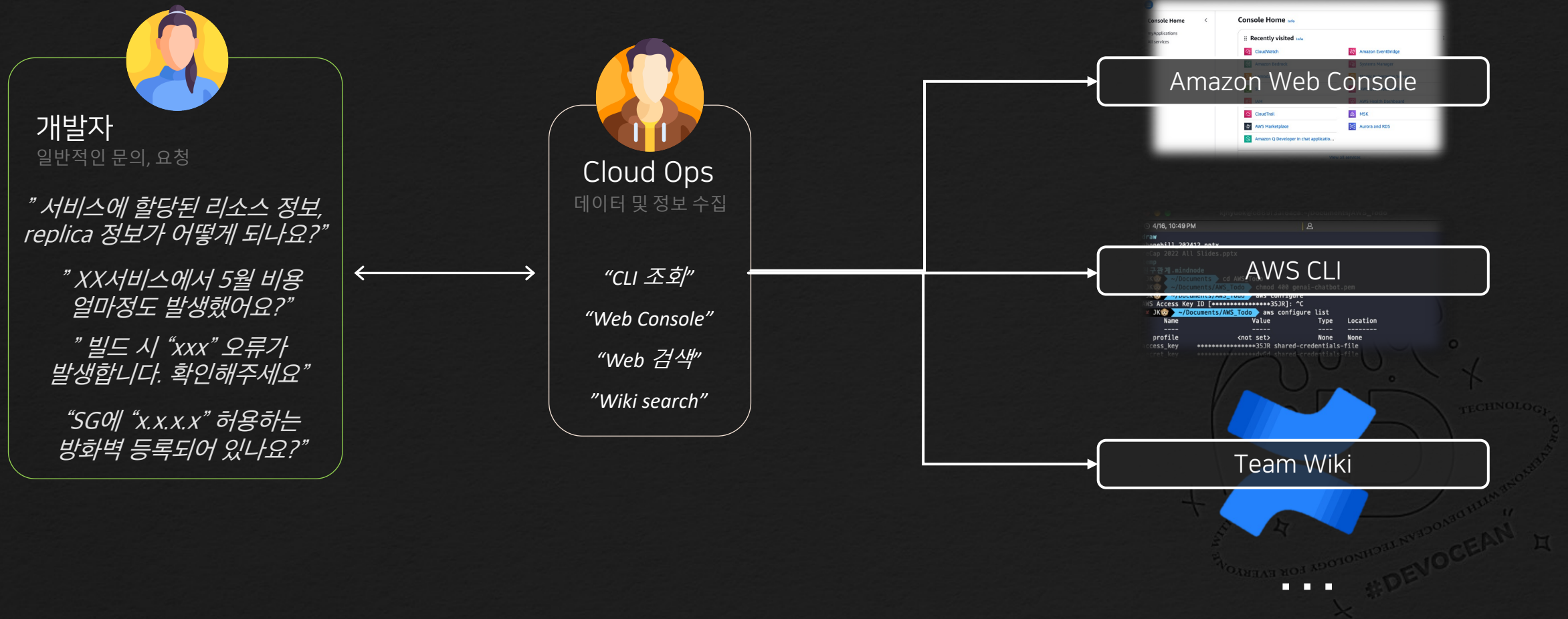
참여자

김플랫폼팀 (서비스 담당자) 조치사항

김플랫폼팀 (서비스 담당자) 조치사항

Case2. LLM 기반 문의, 요청 대응, 정보 검색

유관 담당자로부터 단순 문의, 반복적인 요청 처리에 많은 Effort 사용



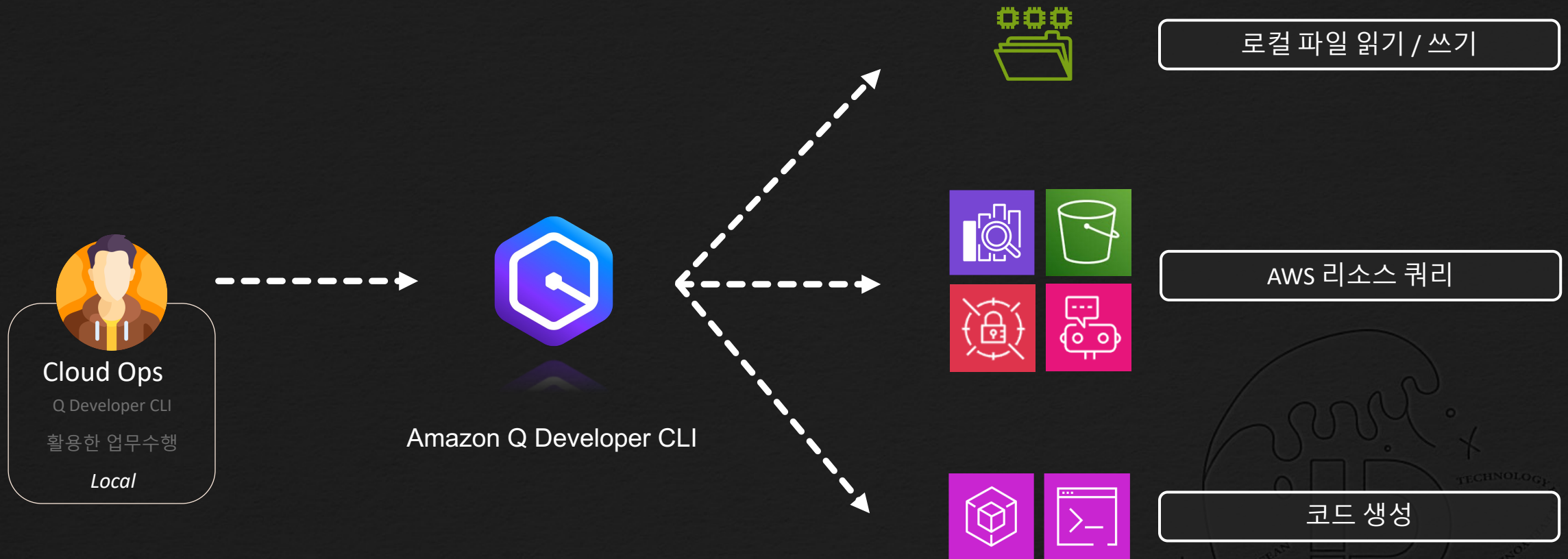
Slack 기반 Agentic Ops의 구현

단순, 반복적인 요청 처리를 협업도구와 Amazon Bedrock 으로 경감



더욱 강력한 도구 - Amazon Q Developer CLI

CLI를 통한 파일 읽기 쓰기, AWS 리소스 쿼리, 코드 생성



Amazon Q Developer CLI - Operation

정보 수집, 처리, 실행 능력이 뛰어난 운영 도구



Amazon Q Developer CLI

MCP

필요 정보 수집
외부 데이터 연동
"AWS 리소스"
"Documents"
"Slack"
"Knowledge Base"



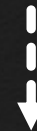
CLI



```
> B0 EKS 클러스터의 bff pod의 replica와 리소스 할당 정보를 요약해서 슬랙 채널 (q-test) 채널로 전송해줘.
> B0 EKS 클러스터의 bff pod 정보를 조회하고 슬랙 채널로 전송해드리겠습니다.

먼저 EKS 클러스터의 bff pod 정보를 확인해보겠습니다.

✂ Using tool: list_k8s_resources from mcp server awslabs.eks-mcp-server
...
● Running list_k8s_resources with the param:
{
  "name": "list_k8s_resources",
  "arguments": {
    "cluster_name": "B0",
    "kind": "Pod",
    "api_version": "v1",
    "label_selector": "app=bff"
  }
}
```



Slack

Built-in

정보 수집
파일 읽기/쓰기/실행
"fs_read"
"fs_write"
...



Replica 설정

- 현재 Replica 수: 1개
- 사용 가능한 Replica: 1개
- 준비된 Replica: 1개
- 배포 전략: Canary (maxSurge: 1, maxUnavailable: 0)

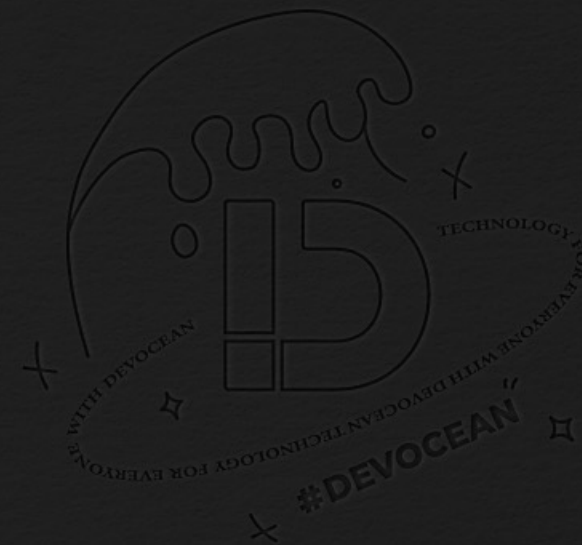


리소스 할당

메인 컨테이너 (service-mesh-app)

- CPU 요청: 300m (0.3 코어)
- 메모리 요청: 933Mi
- 메모리 제한: 933Mi
- 포트: 9000 (HTTP)

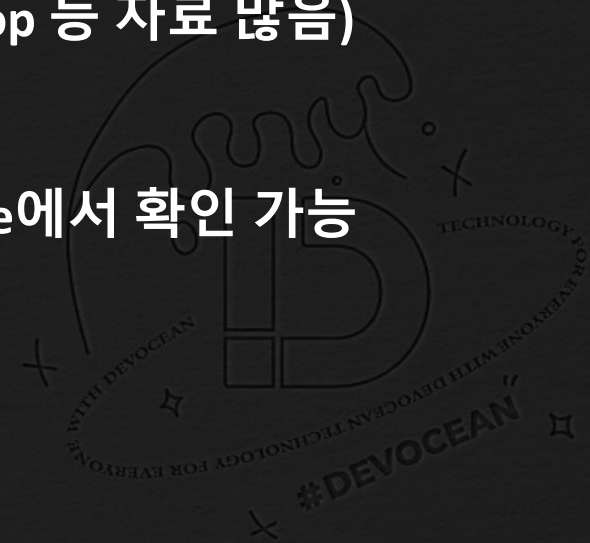
04 Retrospective



도입과정의 Lessons Learned #1

도입의 용이성

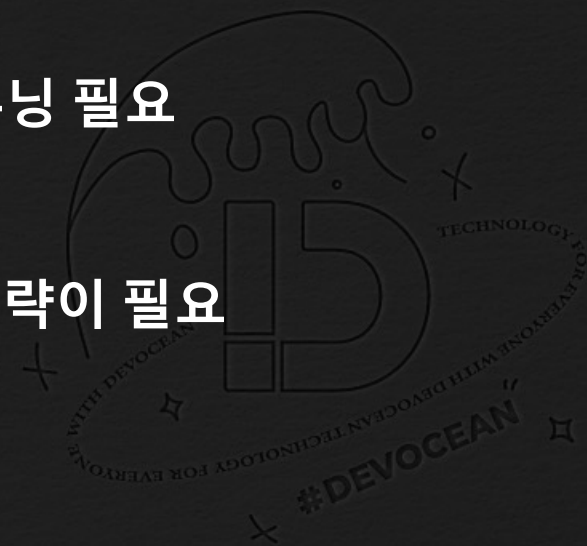
- 1 Amazon Bedrock 은 단일 API를 통해서 다양한 모델 쉽게 사용이 가능
- 2 Lambda를 통해 간편하게 필요한 API(Bedrock/Data Dog/Slack/Atlassian) 로 구현 가능
- 3 익숙한 서비스로 접근하여 GenAI환경 구성가능(AWS Tech Blog, Workshop 등 자료 많음)
(S3, OpenSearch 등으로 Knowledge Bases 구축)
- 4 기 보유한 팀의 Asset을 활용해서 빠르게 구현해 볼 수 있고, Web Console에서 확인 가능



도입과정의 Lessons Learned #2

고려사항

- 1 On-Prem 설치형 Confluence는 파일 변환 필요 > s3로 업로드 위한 Task 필요
- 2 정확한 데이터 탐색을 위해서는 불필요한 지식 자료는 정리 필요
(임시 파일들이나 현행화 되지 않은 정보들)
- 3 Multi Agent 방식활용 시 성능에 민감한 애플리케이션 구현 시 정교한 튜닝 필요
(Main Agent가 의도분류, Sub Agent와 소통하며 각각 응답을 조합하는 구조, 병렬처리 고려)
- 4 응답의 일관성 및 데이터 누락방지를 위한 문서 자산의 적절한 Chunk 전략이 필요



도입의 목적

1. 시스템 모니터링 및 관리
 - 경고에 따른 대응 및 예방조치
 - 장애 및 오류 트러블 슈팅
2. 반복적인 IT 운영 업무의 자동화 및 효율성 극대화
 - 인프라 환경 관리
 - 필요 기능 자동 구현

도입 결과

1. 협업도구와 Amazon Bedrock 연동으로 IT서비스 운영 자동화 실현
2. 인시던트 대응 속도 향상 및 Confluence 자동 리포팅으로 업무 가시성 강화
3. 요청에 의한 단순 확인 업무 대응의 민첩성을 구현 중
4. 추가적인 업무의 자동화 기능 확대 검토 중



감사합니다



내가 만든 에이전트, Bedrock AgentCore로 빠르게 배포하기

하남봉

Solutions Architect, AWS



NEW

Amazon Bedrock AgentCore

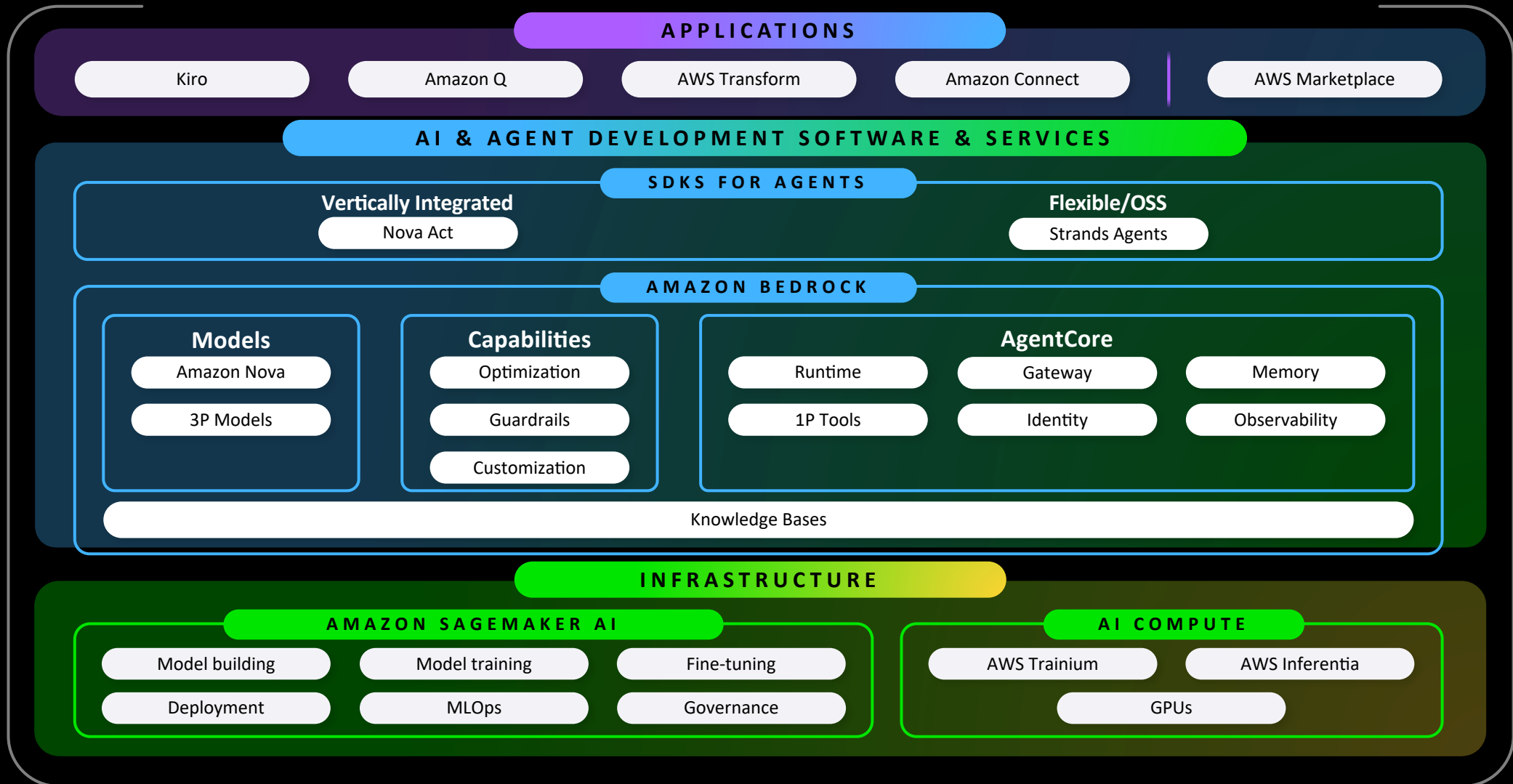
모든 프레임워크와 모델에 적용 가능한, 안전하고 확장 가능한 Agent 배포 및 운영을 위한 종합 서비스

PREVIEW

AWS AI Stack

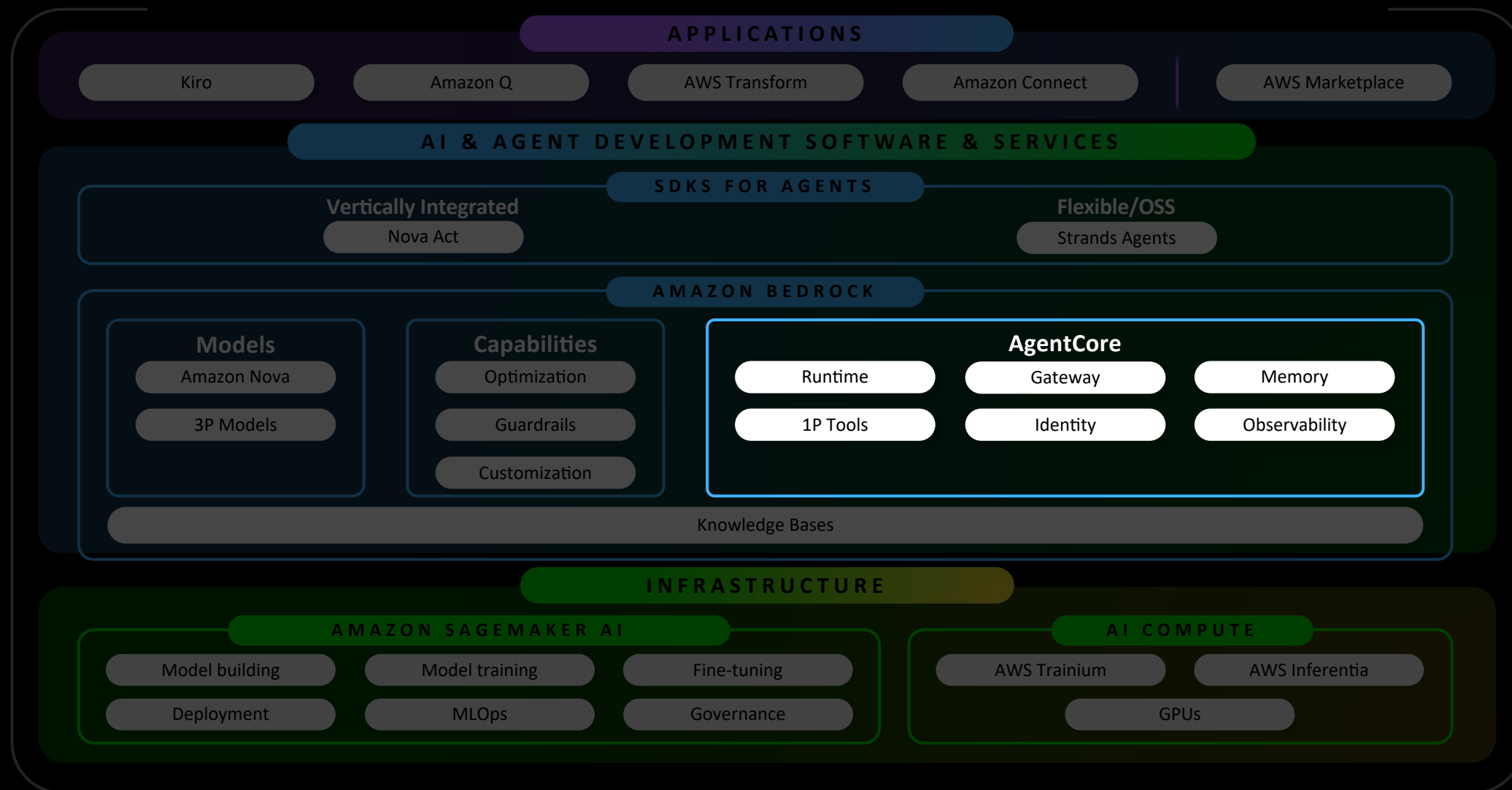
INTERFACES & PROTOCOLS
(MCP/A2A)

DATA



AWS AI Stack

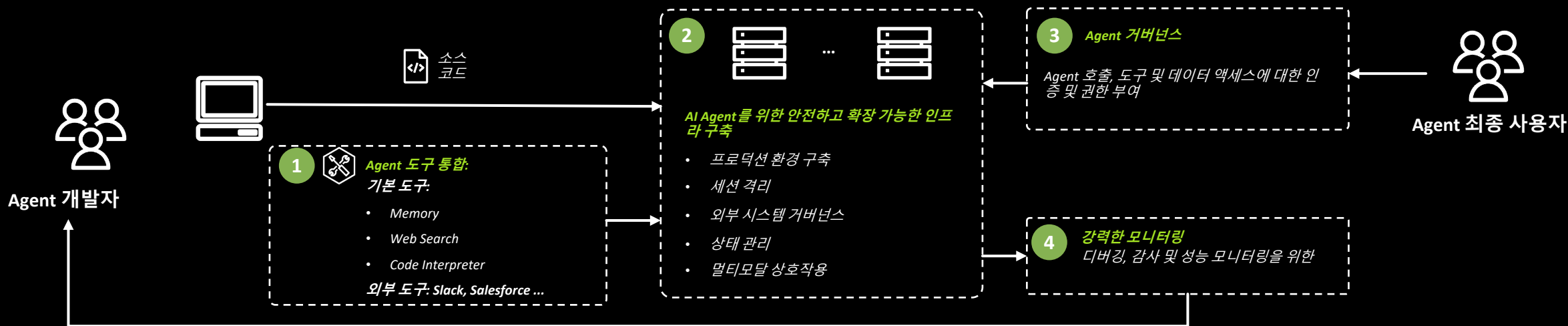
INTERFACES & PROTOCOLS
(MCP/A2A)



DATA



Agent 개발자가 직면한 문제점



개발 단계

배포 단계

1

문제점 1

기본 도구와 외부 도구를 지능형 에이전트에 통합하면 개발 속도가 느려질 수 있습니다

2

문제점 2

Agent 기반 인프라 구축은 개발자가 전문적으로 다루는 영역이 아닙니다

3

문제점 3

제품화를 위해서는 팀 간 보안 검증이 필요하며, 이는 시간이 많이 소요되고 노동 집약적입니다

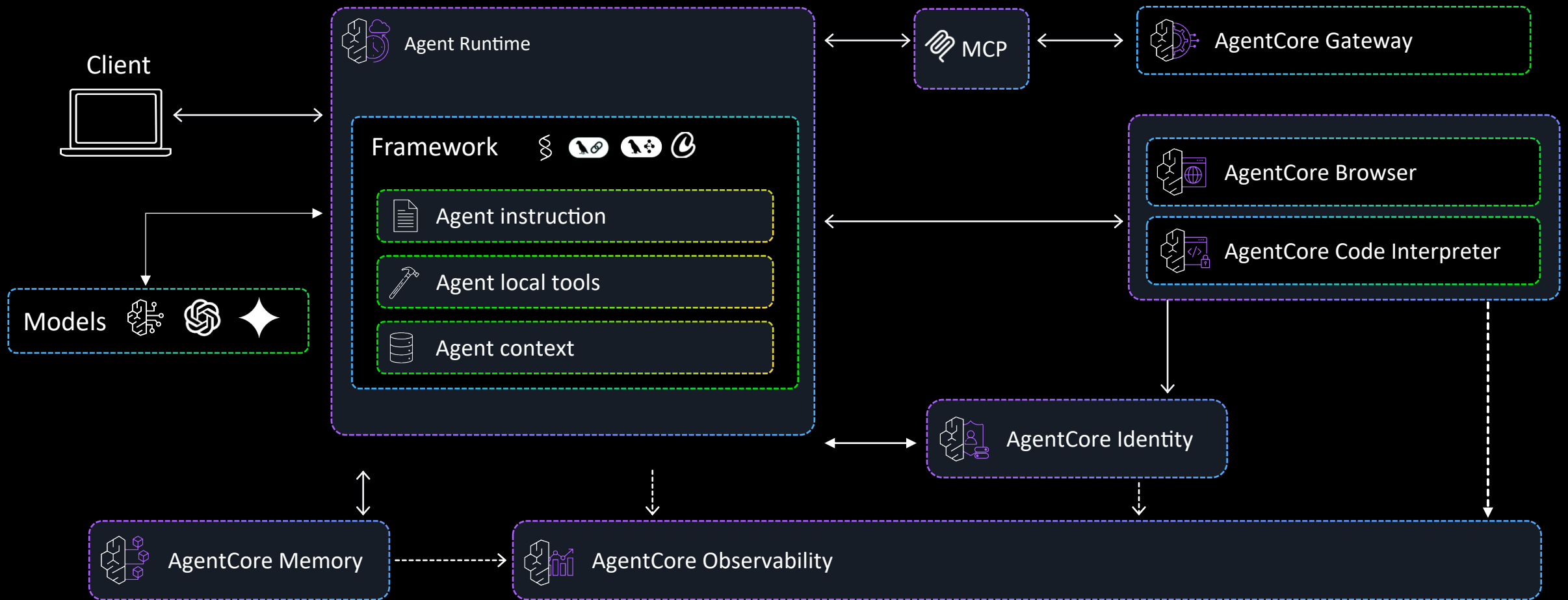
4

문제점 4

비결정적 애플리케이션은 세밀한 엔드투엔드 모니터링이 필요합니다

오늘날 Agent 개발자의 요구: 그들은 이러한 유지보수 및 운영 작업에서 벗어나 지능형 에이전트의 성능 개발과 비즈니스 반복에 더 집중하기를 원합니다

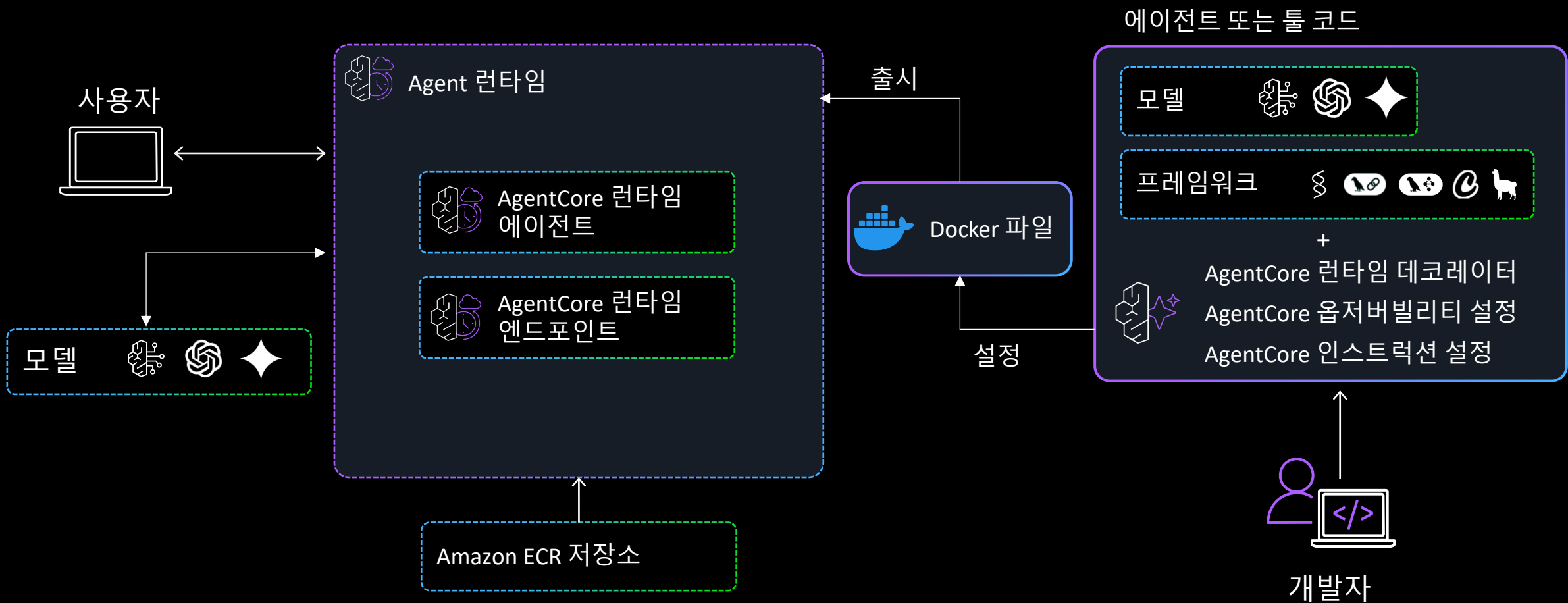
AgentCore의 대규모 에이전트 지원 기능



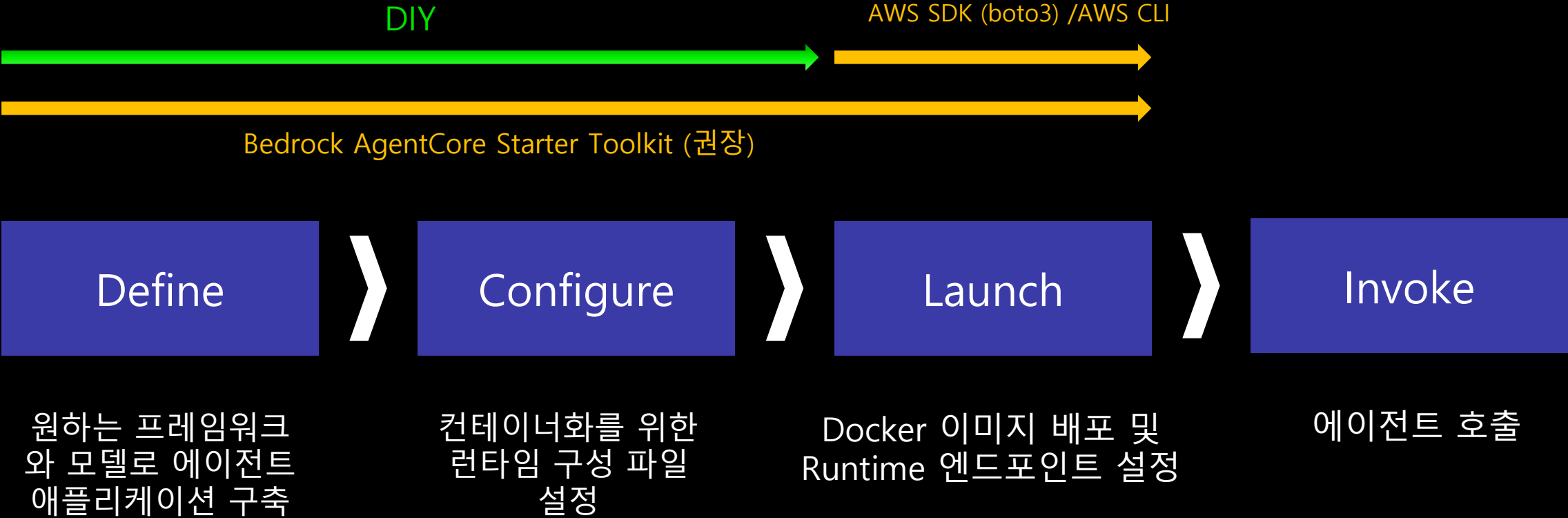


Amazon Bedrock AgentCore Runtime

AgentCore 런타임



AgentCore Runtime 배포 3단계



Define: 기존 Agent 코드 변환

Define

Configure

Launch

Invoke

BedrockAgentCoreApp 인스턴스 초기화

```
from bedrock_agentcore import BedrockAgentCoreApp

app = BedrockAgentCoreApp()
```

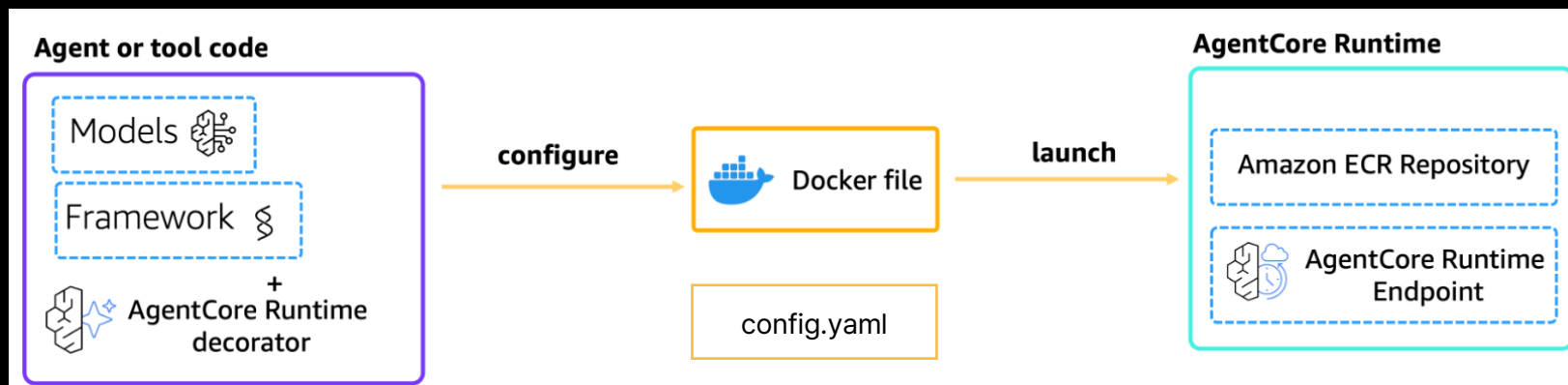
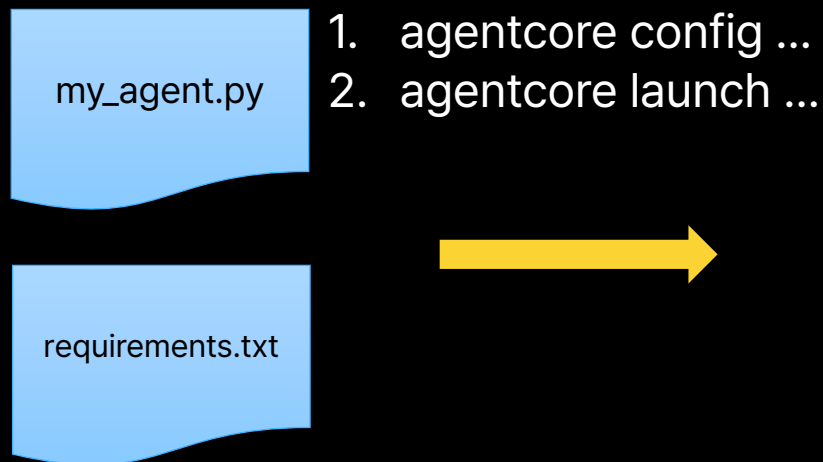
@app.entrypoint 데코레이터를 추가하고, 매개변수를 payload로 설정

```
@app.entrypoint
async def agent_invocation(payload):
    """Handler for agent invocation"""
    user_message = payload.get(
        "prompt", "No prompt found in input, please provide a prompt."
    )
    stream = agent.stream_async(user_message)
    async for event in stream:
        print(event)
        yield (event)
```

파일 마지막에 `app.run()`을 추가하면, 기존 agent 코드가 http 서비스를 제공하는 웹 서버로 변환됨

```
if __name__ == "__main__":
    app.run()
```

AgentCore Runtime 배포

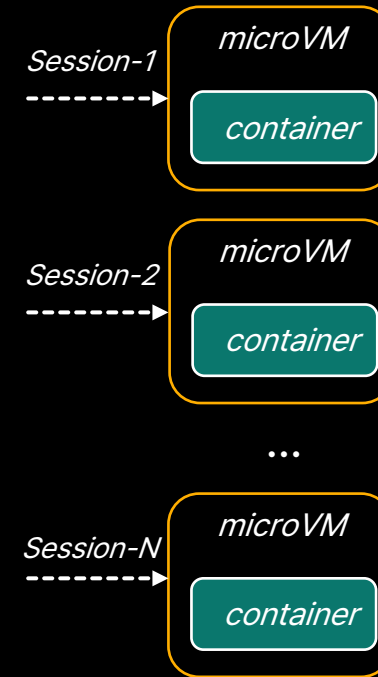
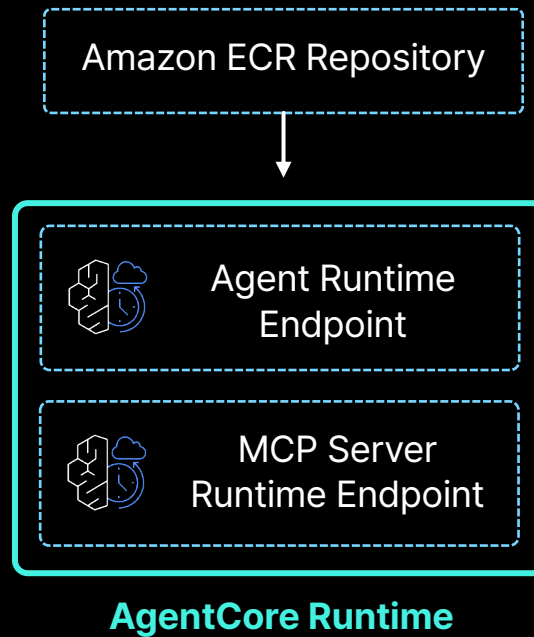
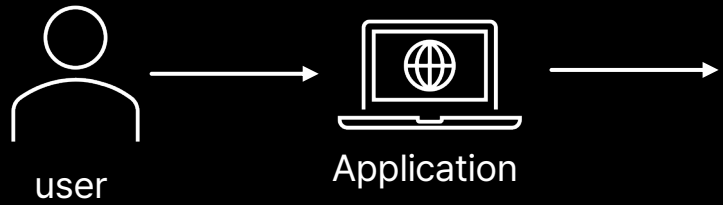


1. 준비: **agentcore configure** --entrypoint my_agent.py -er <IAM_ROLE_ARN> --protocol HTTP 명령을 사용하여 Dockerfile과 yaml 구성 파일 생성
2. 로컬 모드 배포: **agentcore launch -l** 옵션으로 컨테이너를 로컬에서 실행하여 테스트
3. AWS 배포: **agentcore launch**로 Amazon ECR에 이미지 푸시하고 Bedrock AgentCore 런타임 생성

AgentCore Runtime Invoke



```
agentcore_client.invoke_agent_runtime(  
    agentRuntimeArn=agent_arn,  
    runtimeSessionId=session_id,  
    payload=payload  
)
```



Session Lifecycle

- Fast Start : 200 ms
- Auto Cleanup : 15 min idle / 8 hour maximum

AgentCore Runtime Configuration 예시

agentcore configure --entrypoint hello.py

```
> uv run agentcore configure --entrypoint hello.py
Configuring Bedrock AgentCore...
Entrypoint parsed: file=/Users/daekeun/Github/amazon-bedrock-agentcore-samples/01-tutorials/01-AgentCore-runtime/hello.py, bedrock_agentcore.yaml
Agent name: hello
```

🔑 Execution Role

Press Enter to auto-create execution role, or provide execution role ARN/name to use existing
Execution role ARN/name (or press Enter to auto-create): arn:aws:iam::[REDACTED]:role/agentcore-agentcore_strands-role
✓ Using existing execution role: arn:aws:iam::[REDACTED]:role/agentcore-agentcore_strands-role

📁 ECR Repository

Press Enter to auto-create ECR repository, or provide ECR Repository URI to use existing
ECR Repository URI (or press Enter to auto-create): [REDACTED].dkr.ecr.us-east-1.amazonaws.com/bedrock-agentcore-agentcore_strands
✓ Using existing ECR repository: [REDACTED].dkr.ecr.us-east-1.amazonaws.com/bedrock-agentcore-agentcore_strands

🔍 Detected dependency file: requirements.txt

Press Enter to use this file, or type a different path (use Tab for autocomplete):
Path or Press Enter to use detected dependency file:
✓ Using detected file: requirements.txt

🔑 Authorization Configuration

By default, Bedrock AgentCore uses IAM authorization.
Configure OAuth authorizer instead? (yes/no) [no]: no
✓ Using default IAM authorization

Configuring BedrockAgentCore agent: hello

Generated .dockerignore

Generated Dockerfile: /Users/daekeun/Github/amazon-bedrock-agentcore-samples/01-tutorials/01-AgentCore-runtime/Dockerfile

Generated .dockerignore: /Users/daekeun/Github/amazon-bedrock-agentcore-samples/01-tutorials/01-AgentCore-runtime/.dockerignore

Setting 'hello' as default agent

Bedrock AgentCore Configured

Configuration Summary

Name: hello
Runtime: Docker
Region: us-east-1
Account: [REDACTED]
Execution Role: arn:aws:iam::[REDACTED]:role/agentcore-agentcore_strands-role
ECR: [REDACTED].dkr.ecr.us-east-1.amazonaws.com/bedrock-agentcore-agentcore_strands
Authorization: IAM (default)

Configuration saved to: /Users/daekeun/Github/amazon-bedrock-agentcore-samples/01-tutorials/01-AgentCore-runtime/.bedrock_agentcore.yaml

Dockerfile (자동으로 생성되며, .dockerignore도 같이 생성됨)

```
FROM public.ecr.aws/docker/library/python:3.12-slim
WORKDIR /app
```

```
COPY requirements.txt requirements.txt
```

```
# Install from requirements file
```

```
RUN pip install -r requirements.txt
```

```
RUN pip install aws-opentelemetry-distro>=0.10.0
```

```
# Set AWS region environment variable
```

```
ENV AWS_REGION=us-east-1
```

```
ENV AWS_DEFAULT_REGION=us-east-1
```

```
# Signal that this is running in Docker for host binding logic
```

```
ENV DOCKER_CONTAINER=1
```

```
# Create non-root user
```

```
RUN useradd -m -u 1000 bedrock_agentcore
```

```
USER bedrock_agentcore
```

```
EXPOSE 8080
```

```
EXPOSE 8000
```

```
# Copy entire project (respecting .dockerignore)
```

```
COPY . .
```

```
# Use the full module path
```

```
CMD ["opentelemetry-instrument", "python", "-m", "hello"]
```



AgentCore Runtime Launch 예시

AWS 배포 (agentcore launch)

```
Launching Bedrock AgentCore...The push refers to repository [REDACTED].dkr.ecr.us-east-1.amazonaws.com/bedrock-
e456c82aae6: Pushed
f0c5a077d8bb: Pushed
f4fa42b5194c: Layer already exists
f287b50300e3: Layer already exists
f7b1542b9257: Layer already exists
fa24f528c4f6: Layer already exists
f46e0ba021a5: Layer already exists
fec35beaf88: Layer already exists
f3407f3b5b5b: Layer already exists
f4a4c6c9b699: Layer already exists
f6b50848edfe: Layer already exists
Launching Bedrock AgentCore...latest: digest: [REDACTED]f8:
Launching Bedrock AgentCore...Image pushed successfully
Image uploaded to ECR: [REDACTED].dkr.ecr.us-east-1.amazonaws.com/bedrock-agentcore-agentcore_strands
Deploying to Bedrock AgentCore...
Launching Bedrock AgentCore...✔ Agent created/updated: arn:aws:bedrock-agentcore:us-east-1:[REDACTED]:runtime/
Polling for endpoint to be ready...
Launching Bedrock AgentCore...Agent endpoint: arn:aws:bedrock-agentcore:us-east-1:[REDACTED]:runtime/hello-SABM
Image pushed to ECR: [REDACTED].dkr.ecr.us-east-1.amazonaws.com/bedrock-agentcore-agentcore_strands:latest
Bedrock AgentCore Deployed —
Deployment Successful!

Agent Name: hello
Agent ARN: arn:aws:bedrock-agentcore:us-east-1:[REDACTED]:runtime/hello-SABMWW83Jy
ECR URI: [REDACTED].dkr.ecr.us-east-1.amazonaws.com/bedrock-agentcore-agentcore_strands

You can now check the status of your Bedrock AgentCore endpoint with:
agentcore status

You can now invoke your Bedrock AgentCore endpoint with:
agentcore invoke '{"prompt": "Hello"}'

📁 Agent logs available at:
/aws/bedrock-agentcore/runtimes/hello-SABMWW83Jy-DEFAULT
/aws/bedrock-agentcore/runtimes/hello-SABMWW83Jy-DEFAULT/runtime-logs

💡 Tail logs with:
aws logs tail /aws/bedrock-agentcore/runtimes/hello-SABMWW83Jy-DEFAULT --follow
aws logs tail /aws/bedrock-agentcore/runtimes/hello-SABMWW83Jy-DEFAULT --since 1h
```


AgentCore Runtime Invoke 예시

로컬 모드 테스트 결과

```
> uv run agentcore invoke -l '{"prompt": "Hello"}'
Payload:
{
  "prompt": "Hello"
}
Invoking BedrockAgentCore agent 'hello' locally
Workload not detected, creating...
Created workload workload-dd41ac86
Session ID: 1ea30c13-c9d9-49bc-9223-51543ef9c24d

Response:
{
  "response": "\"Hello! How are you doing today? Is there anything I can help"
}

> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
73f02423b354   bedrock_agentcore-hello:latest      "opentelemetry-instr..." 2 minutes ago
```

엔드포인트 배포 후 테스트 결과

```
> uv run agentcore invoke '{"prompt": "Hello"}'
Payload:
{
  "prompt": "Hello"
}
Invoking BedrockAgentCore agent 'hello' via cloud endpoint
Session ID: e9d195ce-5afd-40c1-8d40-759efdd263f6

Response:
{
  "ResponseMetadata": {
    "RequestId": "d2989c30-2dd8-4d8f-9121-eb9e1e7e2a45",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Sun, 27 Jul 2025 05:25:43 GMT",
      "content-type": "application/json",
      "transfer-encoding": "chunked",
      "connection": "keep-alive",
      "x-amzn-requestid": "d2989c30-2dd8-4d8f-9121-eb9e1e7e2a45",
      "baggage": "Self=1-6885b855-7898d88258d9b1bb06eda6c3,session.id=e9d195ce-5afd-40c1-8d40-759efdd263f6",
      "x-amzn-bedrock-agentcore-runtime-session-id": "e9d195ce-5afd-40c1-8d40-759efdd263f6",
      "x-amzn-trace-id": "Root=1-6885b855-37cb731700a957a21637d085;Self=1-6885b855-7898d88258d9b1bb06eda6c3"
    },
    "RetryAttempts": 0
  },
  "runtimeSessionId": "e9d195ce-5afd-40c1-8d40-759efdd263f6",
  "traceId": "Root=1-6885b855-37cb731700a957a21637d085;Self=1-6885b855-7898d88258d9b1bb06eda6c3",
  "baggage": "Self=1-6885b855-7898d88258d9b1bb06eda6c3,session.id=e9d195ce-5afd-40c1-8d40-759efdd263f6",
  "contentType": "application/json",
  "statusCode": 200,
  "response": [
    "b\"Hello! How are you doing today? Is there anything I can help you with?\\n\\n\""
  ]
}
```



Amazon Bedrock AgentCore Gateway

AgentCore Gateway

간소화된 도구 개발 및 통합



- API, Lambda 함수 및 기존 서비스를 MCP 호환 도구로 전환
- 표준화된 인터페이스를 통해 수천 개의 도구 액세스

안전하고 통합된 액세스



- 단일 보안 엔드포인트를 통해 도구 발견 및 사용
- 여러 도구 소스를 하나의 통합 인터페이스로 결합

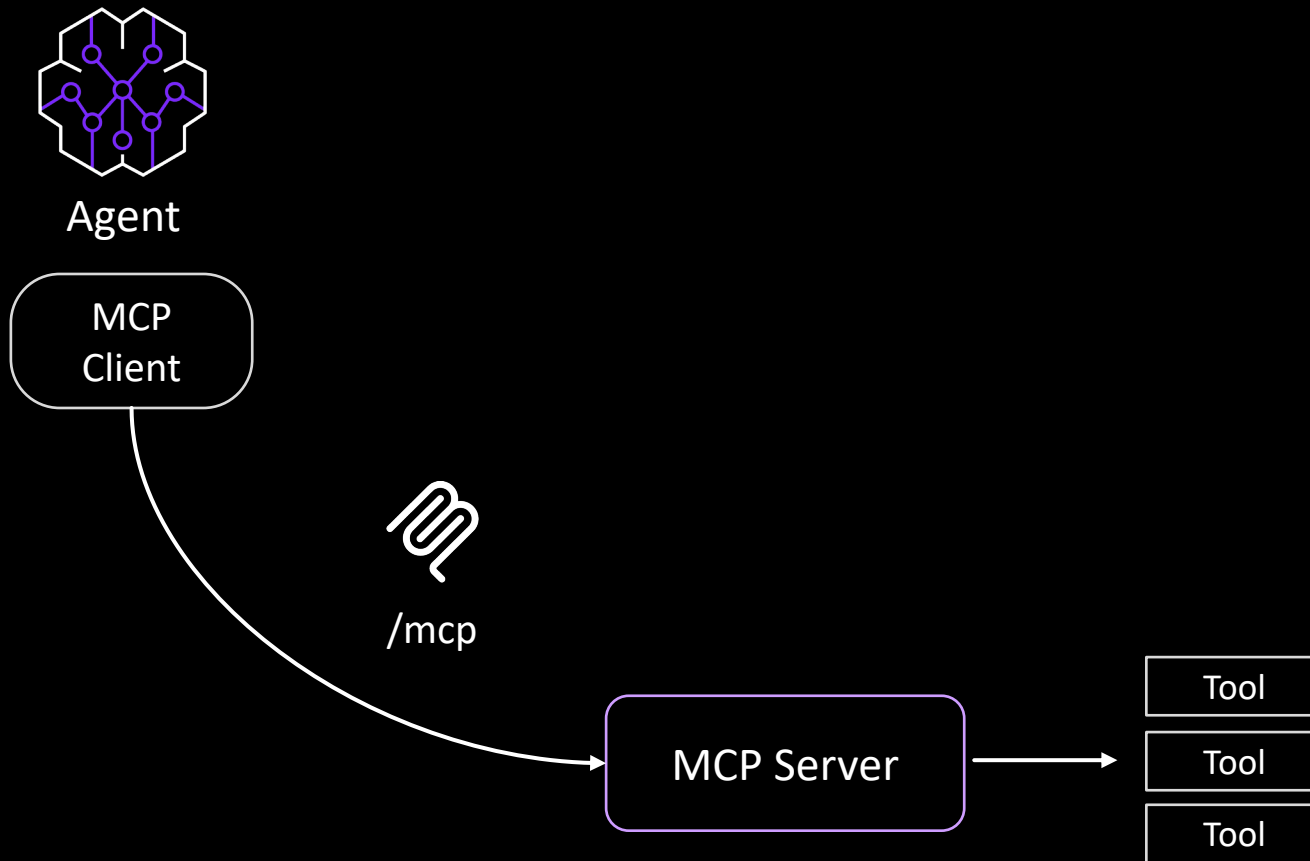
지능형 도구 검색



- 상황 인식 발견으로 에이전트가 적절한 도구를 찾을 수 있도록 지원
- 세분화된 권한을 가진 선별된 도구 컬렉션

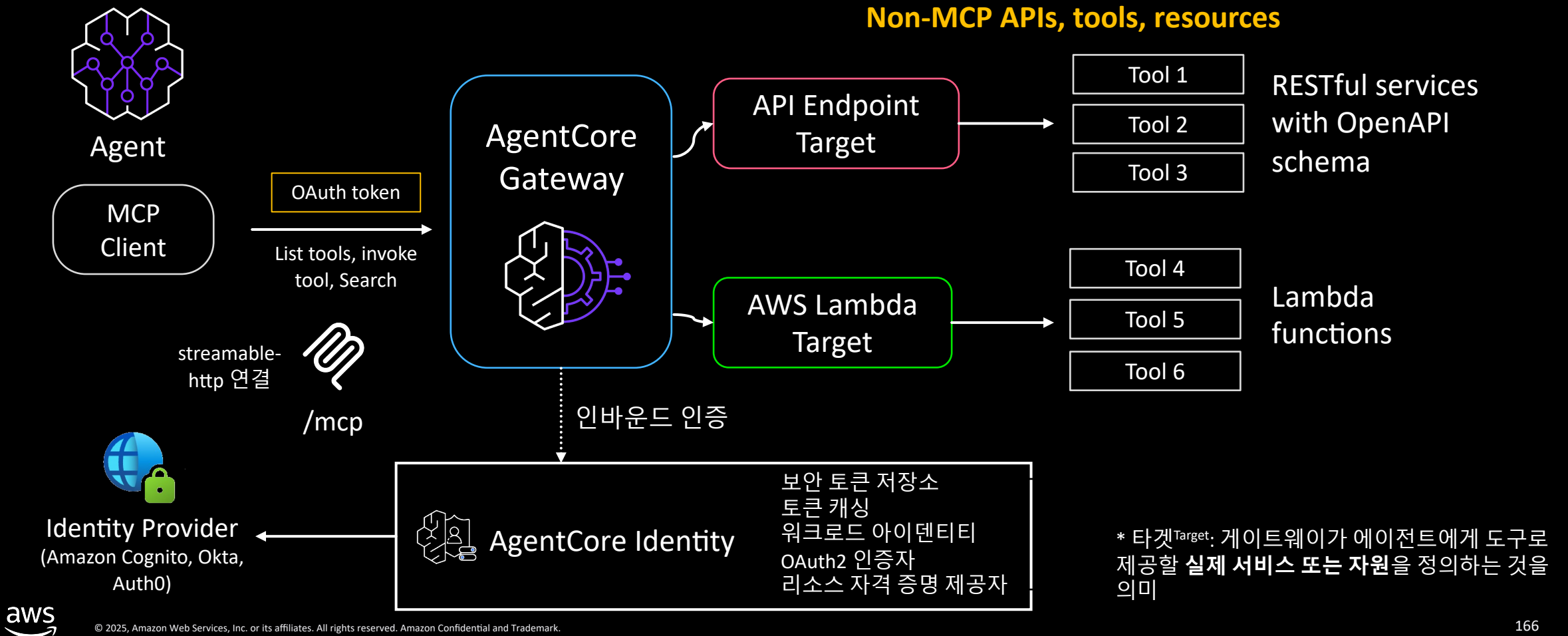
Without AgentCore Gateway

- 각각의 도구를 개별적으로 관리해야 했기 때문에, 일관된 방식으로 도구들을 정의하기 어려움.
- 대규모의 도구를 관리할 경우, 복잡도가 기하급수적으로 증가함.



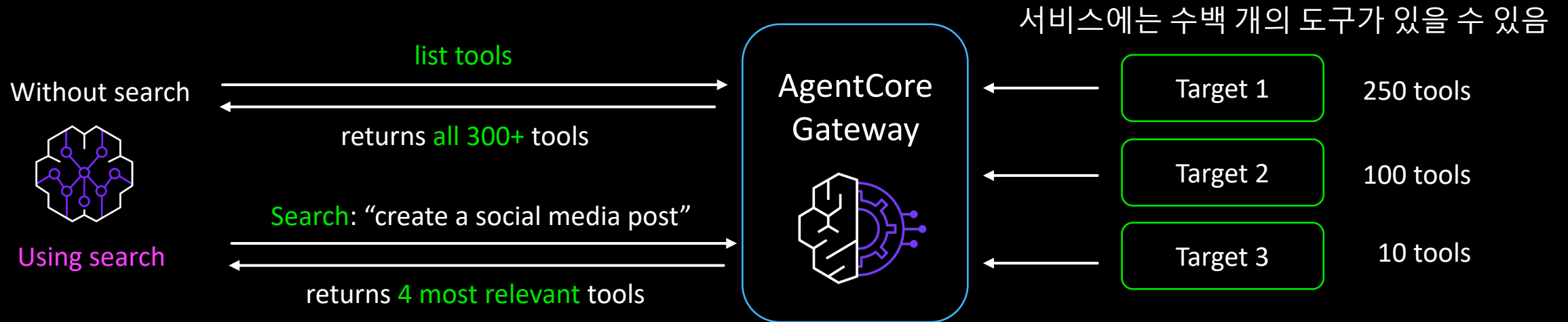
AgentCore Gateway

- API 스펙(OpenAPI/Smithy 등)이나 Lambda 함수를 AgentCore Gateway에 등록하면, Gateway가 이를 에이전트가 이해하고 호출할 수 있는 MCP 기반의 '도구^{tool}'로 바꿔주는 역할을 합니다.



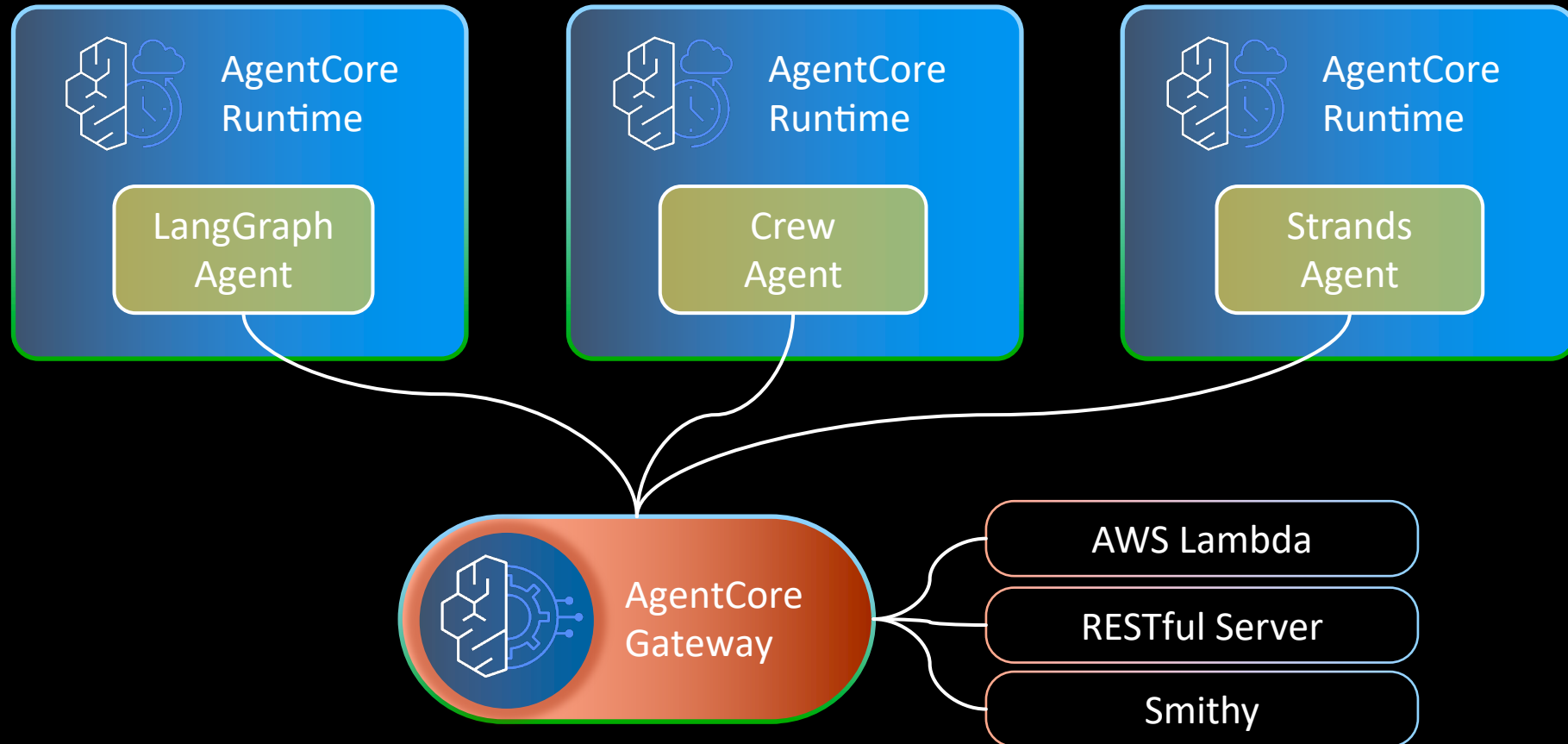
AgentCore Gateway semantic search

- AgentCore Gateway가 자동으로 도구를 인덱싱하고 서버리스 의미 검색을 제공
- 에이전트의 LLM에 전달되는 컨텍스트를 줄여 정확성, 속도 및 비용 개선
- 에이전트가 주어진 작업에 관련된 도구에 집중할 수 있도록 지원



프레임워크 중립적 실행 환경과 통합 Gateway

- AgentCore Runtime은 프레임워크 중립적 실행 환경 (LangGraph, Crew, Strands 등 모두 사용 가능)
- 각 Agent는 Bedrock Agent Core Gateway를 통해 통합된 도구를 이용할 수 있음



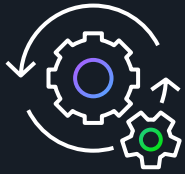


Amazon Bedrock AgentCore Memory



AgentCore Memory

메모리 관리 단순화



- 메모리 인프라 추상화
- 서버리스 아키텍처로 자동 확장
- 세션 간 에이전트 컨텍스트 자동 저장 및 관리

기업용



- 각 고객을 위한 전용 스토리지로 완벽한 데이터 프라이버시 보장
- 암호화 및 지역별 데이터 스토리지 옵션을 통한 기업 수준의 보안

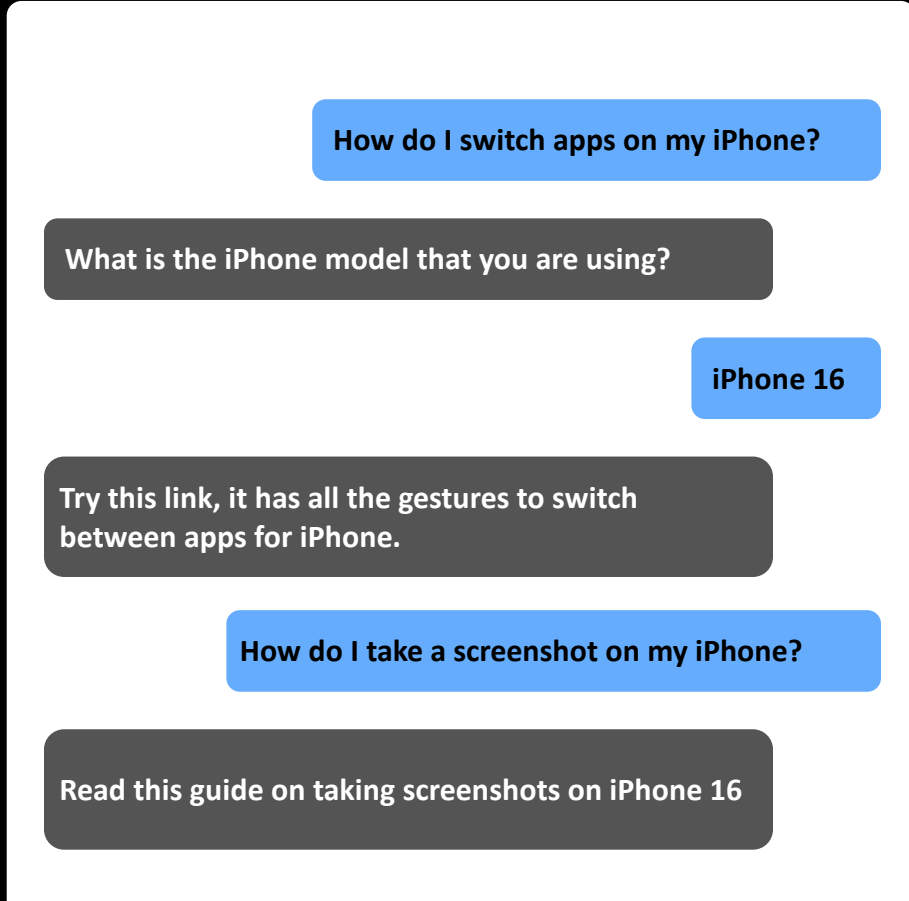
심층 커스터마이제이션



- 사용 사례에 따른 메모리 패턴 정의
- 추출 규칙 구성
- 메모리 추출을 위한 모델 선택 및 프롬프트 커스터마이징

AgentCore Memory - Short Term Memory

즉시 문맥을 파악할 수 있도록 대화 내용 저장

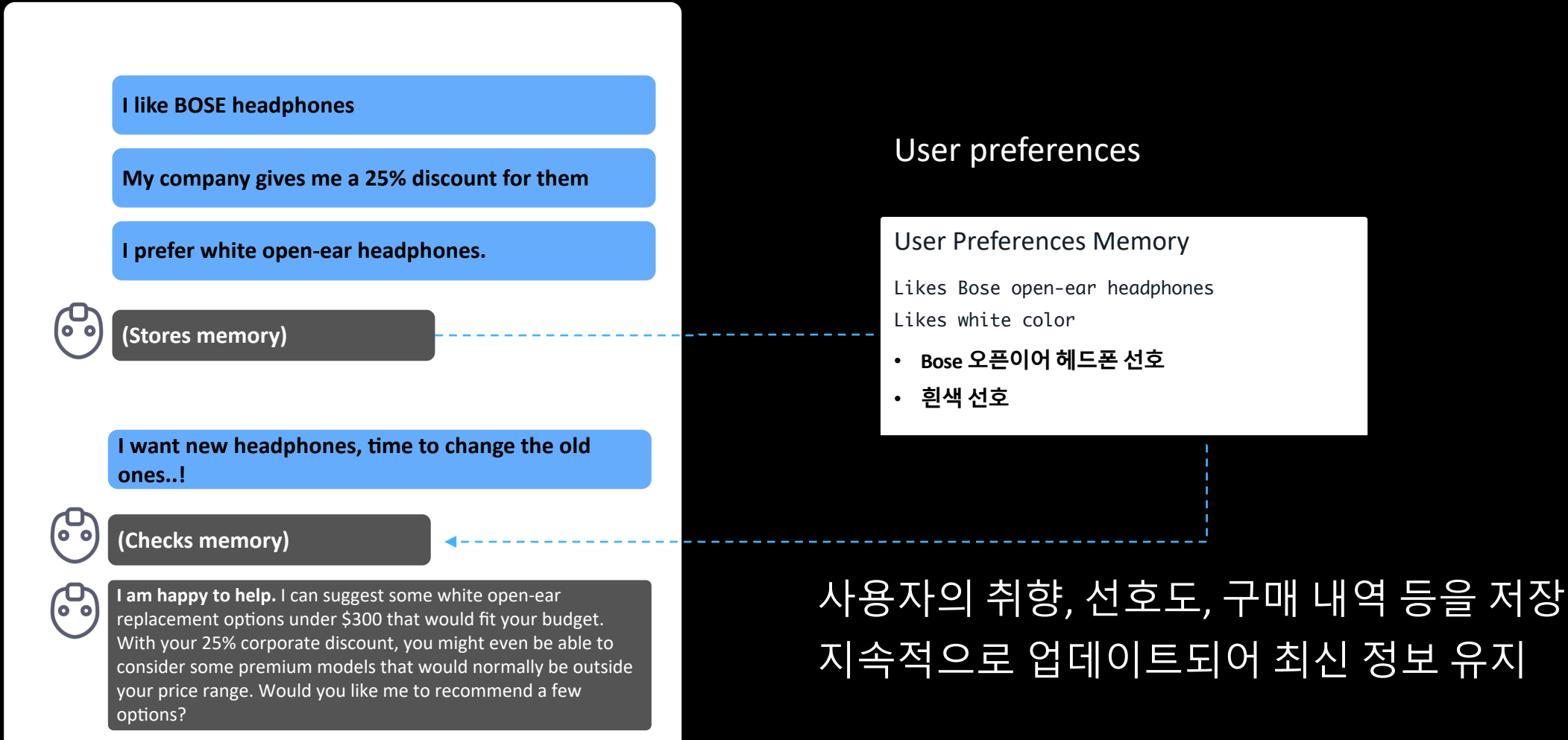


단기 메모리 사용 목적:

- 대화에 지속적으로 참여
- 즉각적인 요구사항 이해
- 중단했던 지점에서 정확히 재개
- 함께 발견한 내용 기억
- 공유하는 내용과 원활하게 연동

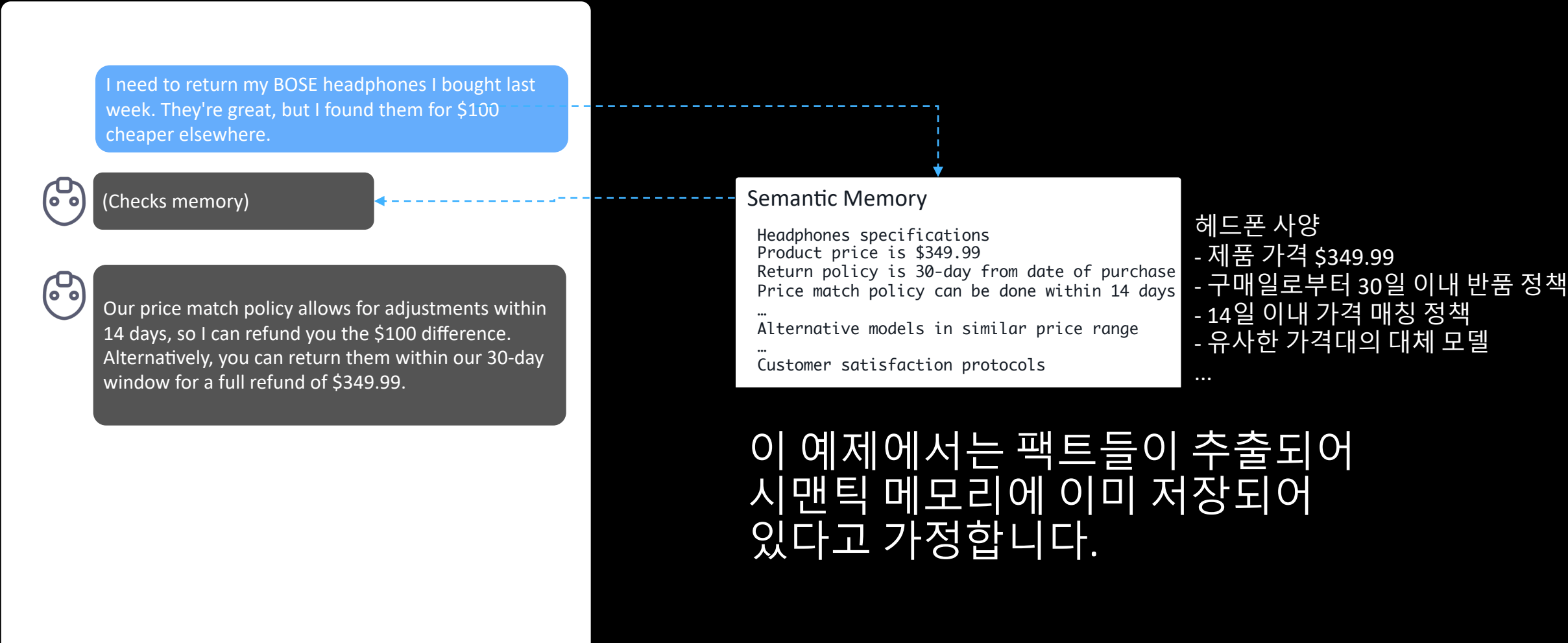
AgentCore Memory - Long Term Memory

저장 전략: User Preferences (유저 선호)



AgentCore Memory - Long Term Memory

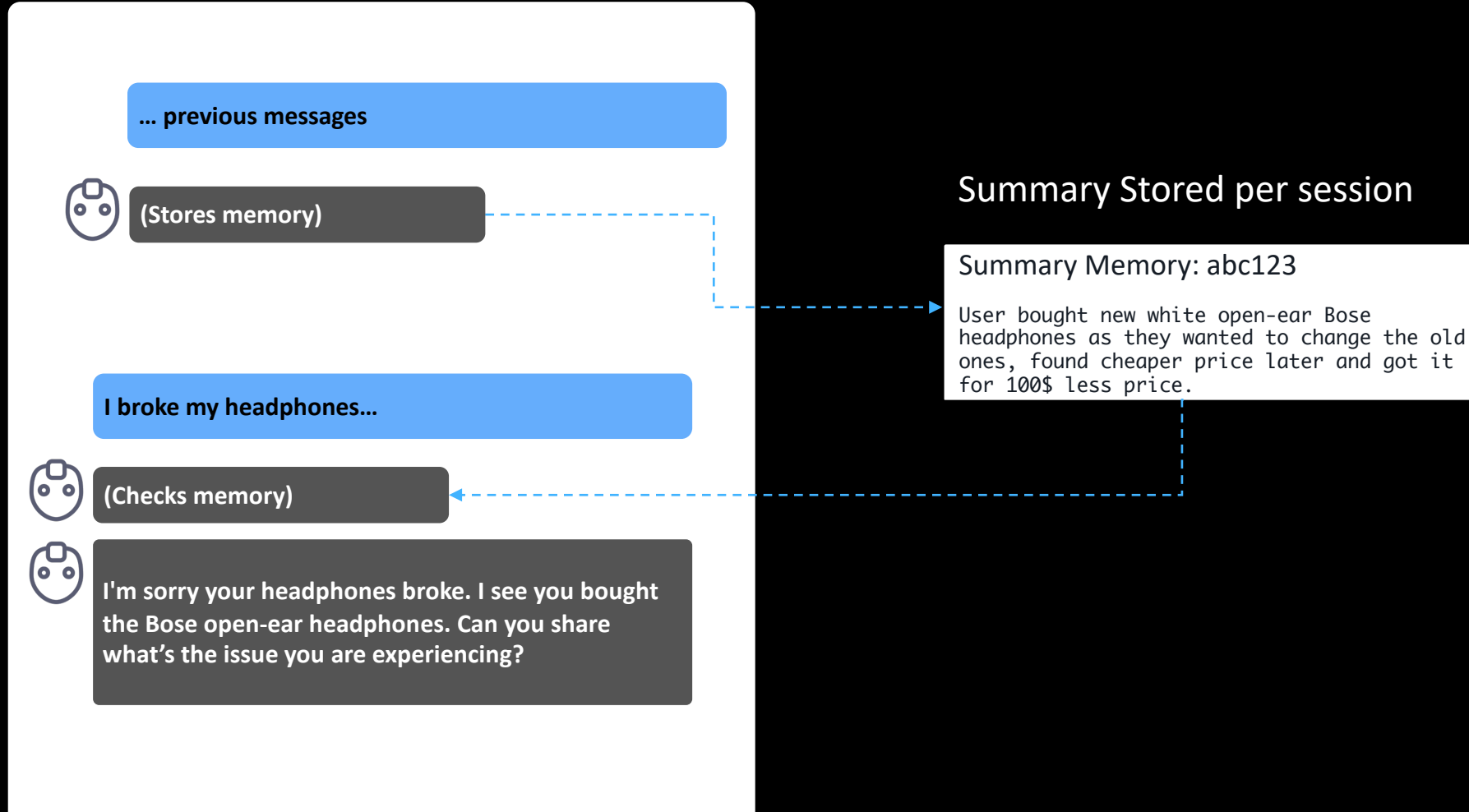
저장 전략: Semantic (Fact 저장)



이 예제에서는 팩트들이 추출되어
시맨틱 메모리에 이미 저장되어
있다고 가정합니다.

AgentCore Memory - Long Term Memory

저장 전략: Summary (요약)

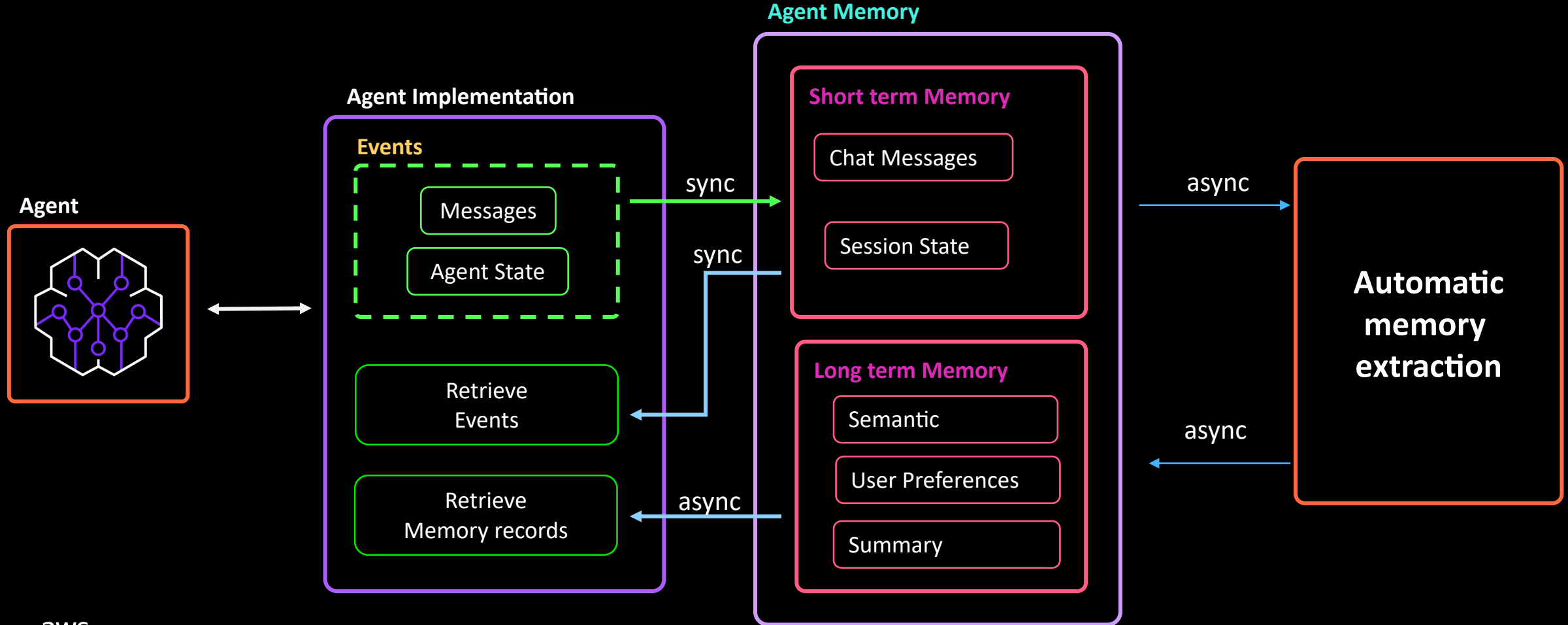


사용자는 오래된
헤드폰을 교체하기 위해
흰색 Bose 오픈이어
헤드폰을 새로 구입했고,

이후 더 저렴한 가격을
찾아 100달러 싸게 구매할
수 있었습니다.

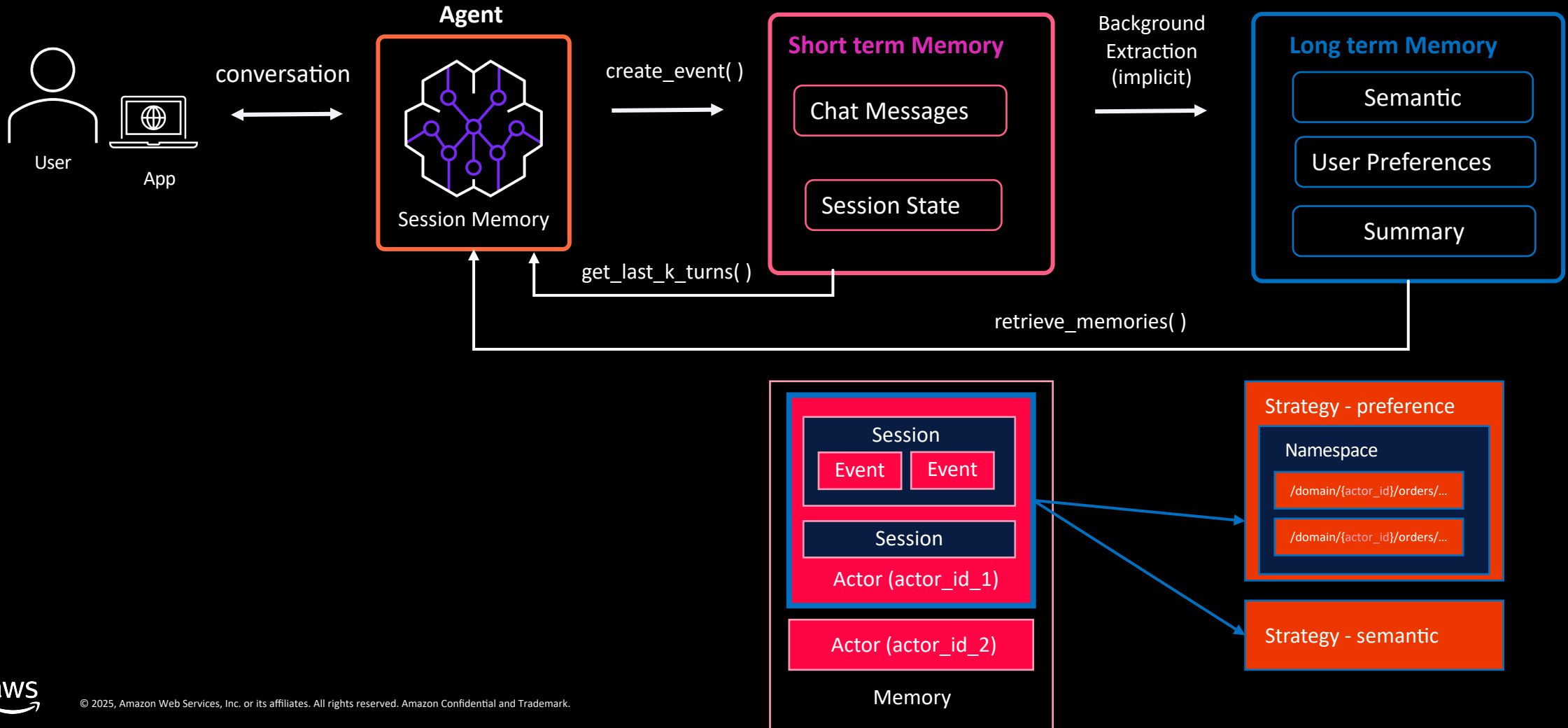
Short-term and long-term memory capabilities

- Agent는 Agent Implementation을 통해 메모리 시스템과 상호작용
- 비동기적 자동 추출 기능이 단기에서 장기 메모리로 중요 정보를 이동시켜 효율적인 메모리 관리

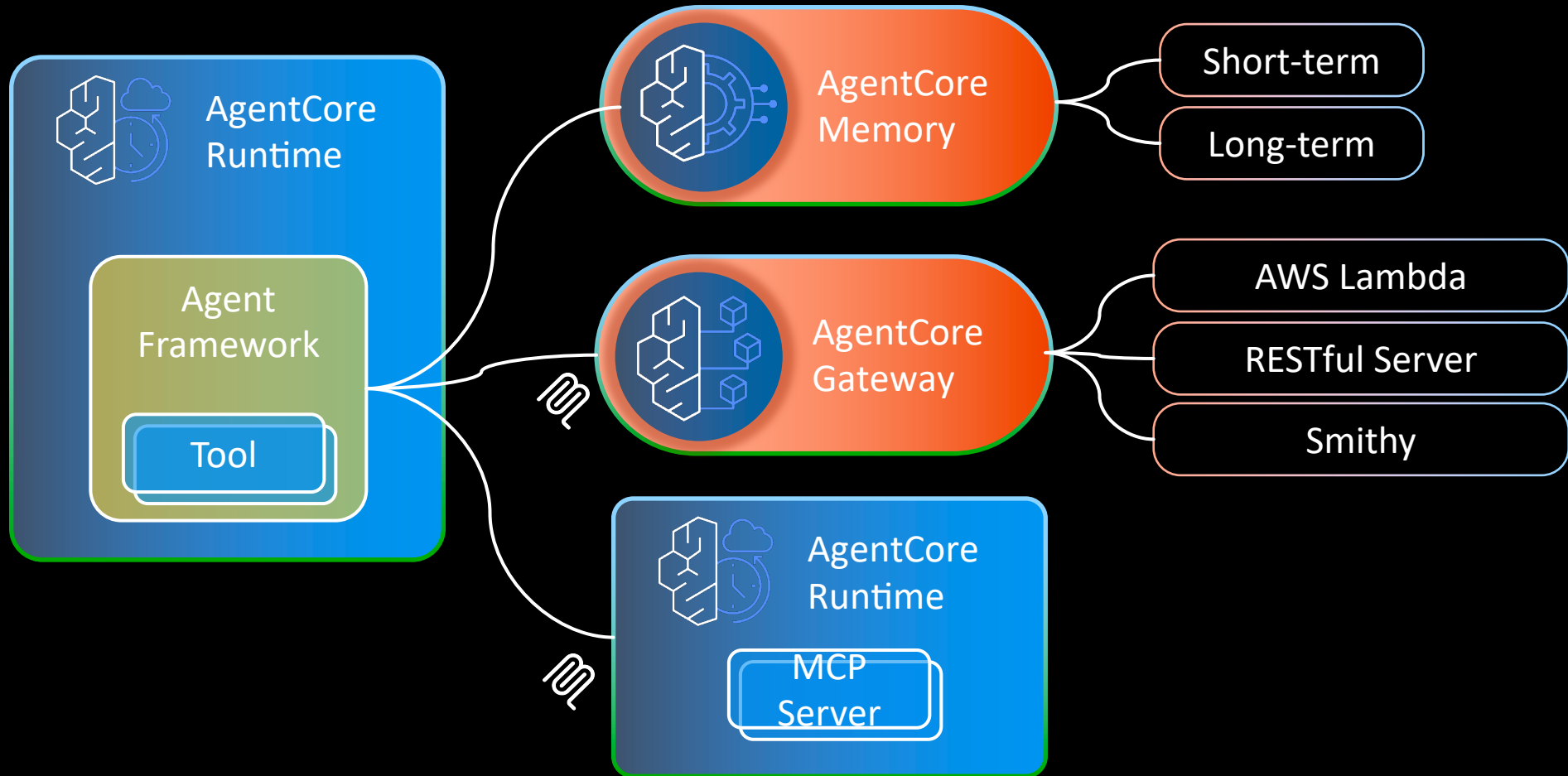


Long Term Memory – Data Flow

- 단기에서 장기 메모리로 추출 전략은 Preference와 Semantic, Summary로 나뉘어 네이스페이스별로 관리



Gateway와 Memory를 활용하는 Runtime 기반 에이전트





Amazon Bedrock AgentCore Identity

AgentCore Identity

AI 에이전트를 위한 안전한 권한 위임 액세스



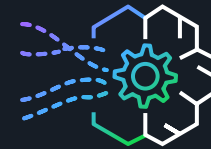
- AI 에이전트가 GitHub, Google, Salesforce, Slack과 같은 AWS 리소스 및 타사 도구에 안전하게 액세스할 수 있도록 지원
- 최소 권한 액세스와 안전한 권한 위임을 통한 강력한 액세스 제어

간소화된 AI 에이전트 구축



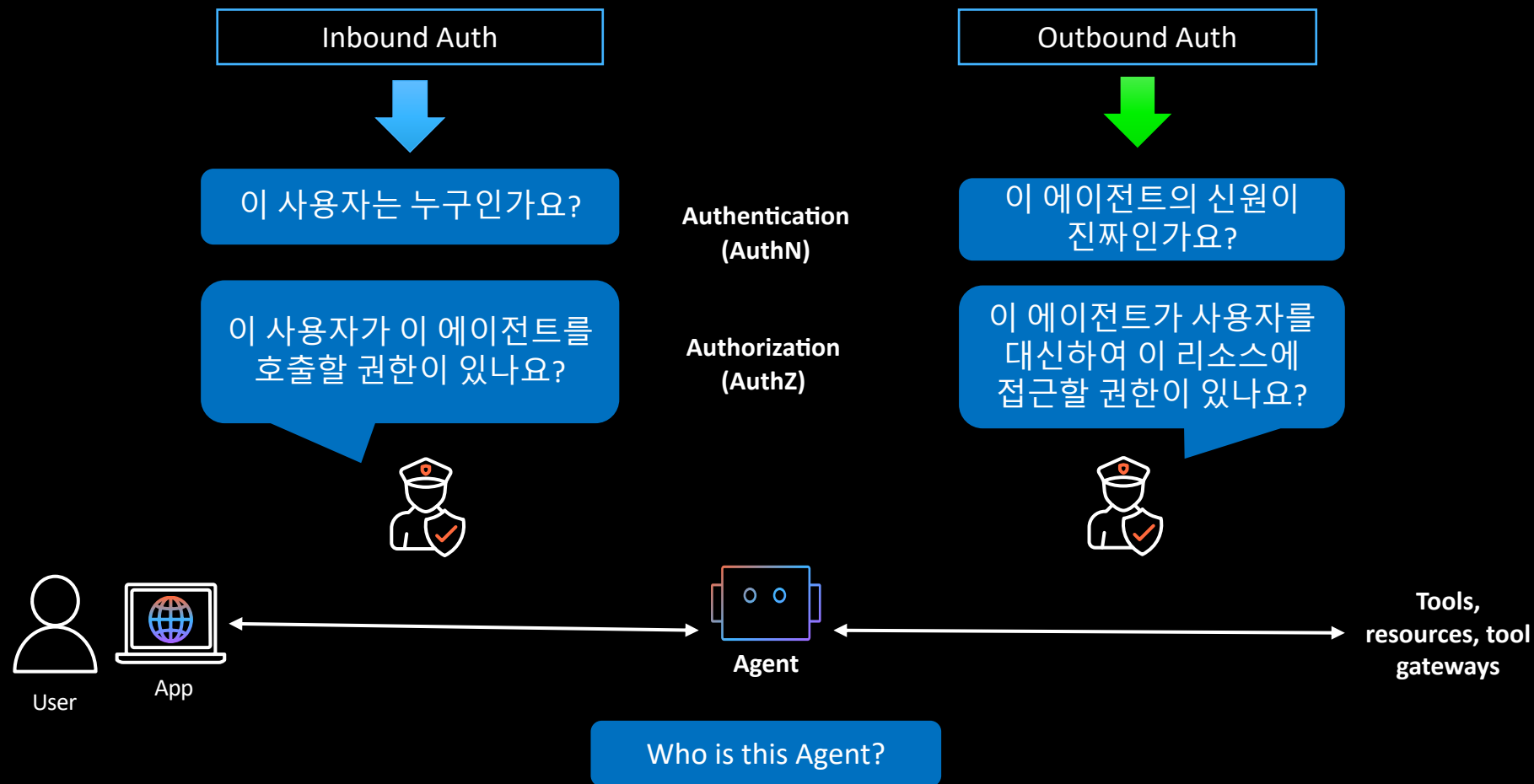
- 보안 토큰 저장소로 동의 피로도 최소화
- 인증 흐름 간소화

가속화된 AI 에이전트 개발

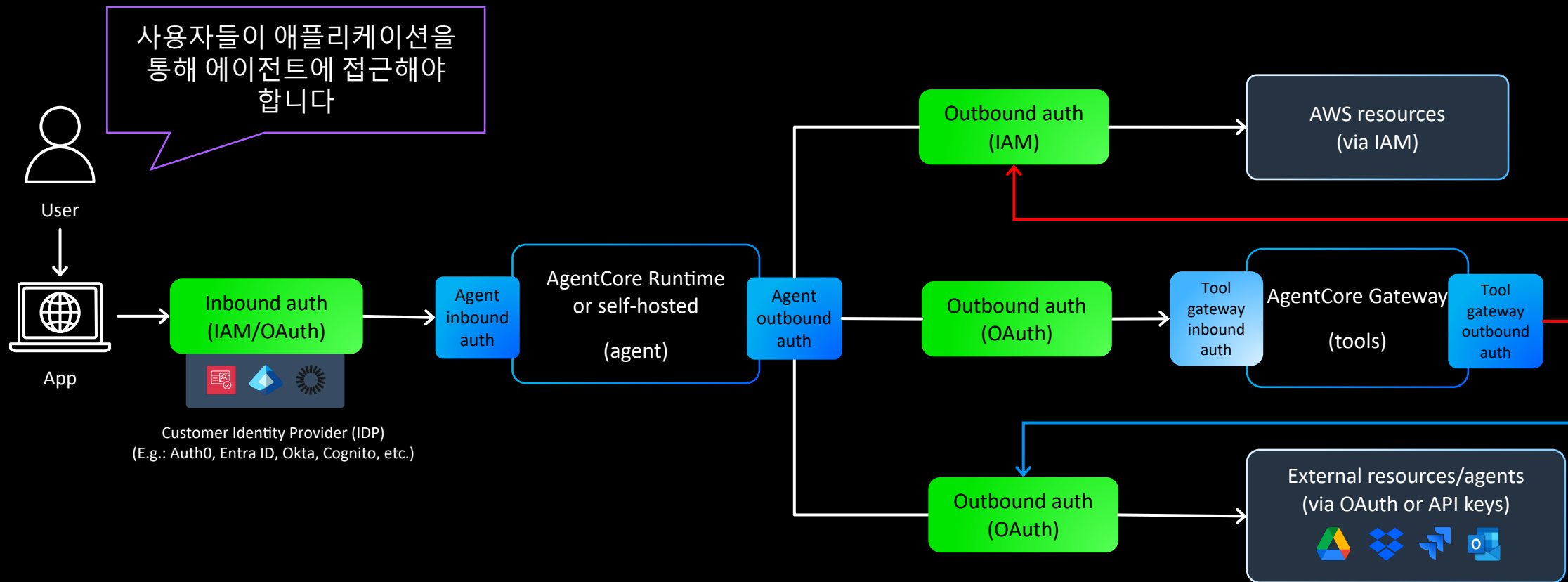


- Okta, Azure AD 또는 Amazon Cognito와 같은 기존 ID 시스템 유지
- 사용자 마이그레이션이나 인증 흐름 재구축 없이 커스텀 개발 노력 감소

Agentic AI Auth Basics



Auth with AgentCore Identity





Amazon Bedrock AgentCore Observability

AgentCore Observability

품질과 신뢰 유지



- 에이전트 동작 전반에 대한 완전한 엔드투엔드 가시성 제공
- 빠른 디버깅과 품질 감사 수행
- 문제를 신속하게 감지하고 성능 추세를 평가

시장 출시 시간 단축



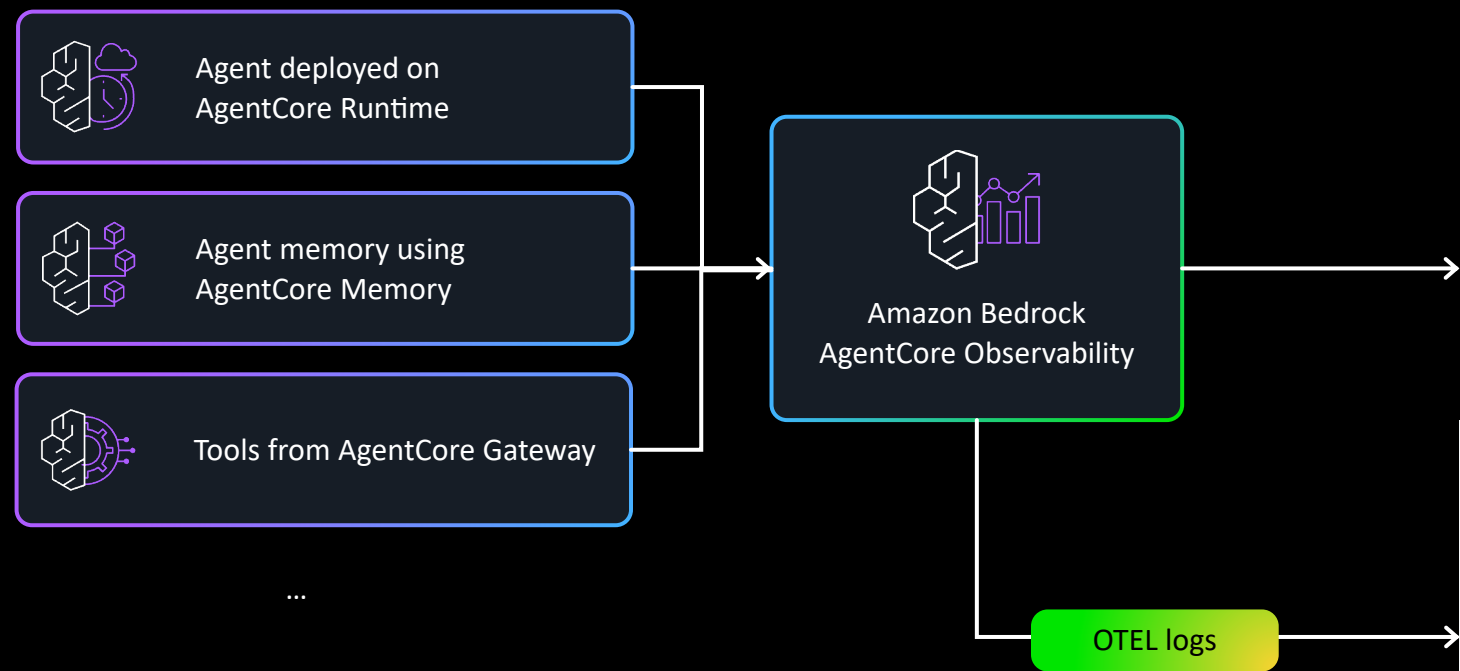
- CloudWatch로 구동되는 대시보드가 개발자의 시간을 절약
- 에이전트의 운영 상태를 단일 화면에서 확인 가능
- 다양한 소스의 데이터를 수동으로 통합하는 작업 불필요

서드파티 관찰가능성 도구와 통합

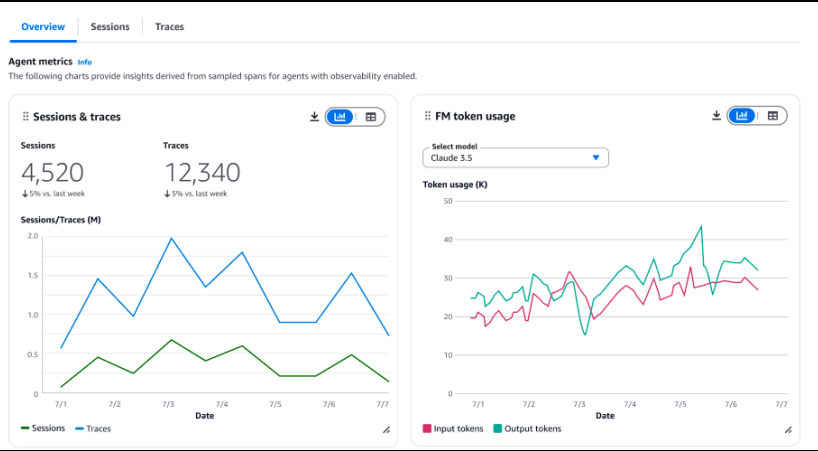


- CloudWatch를 비롯한 다양한 모니터링 및 관찰가능성 도구와의 통합 지원
- 이미 구축된 관찰가능성 스택을 그대로 활용 가능한 유연성 제공

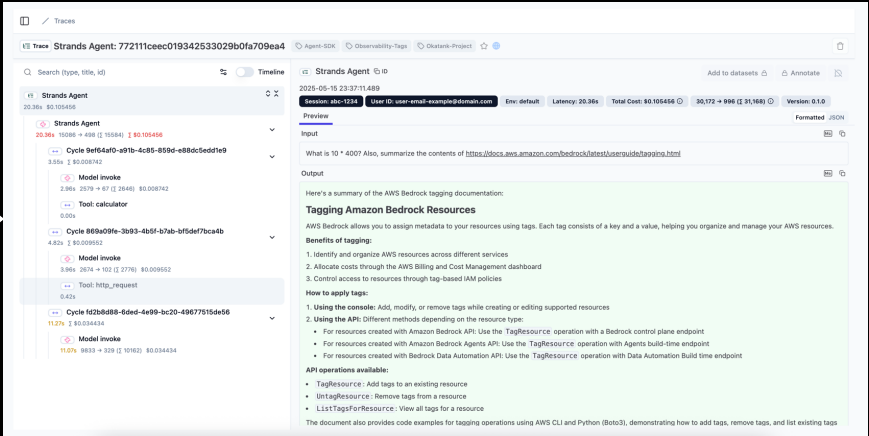
AgentCore Observability



AgentCore Observability dashboards

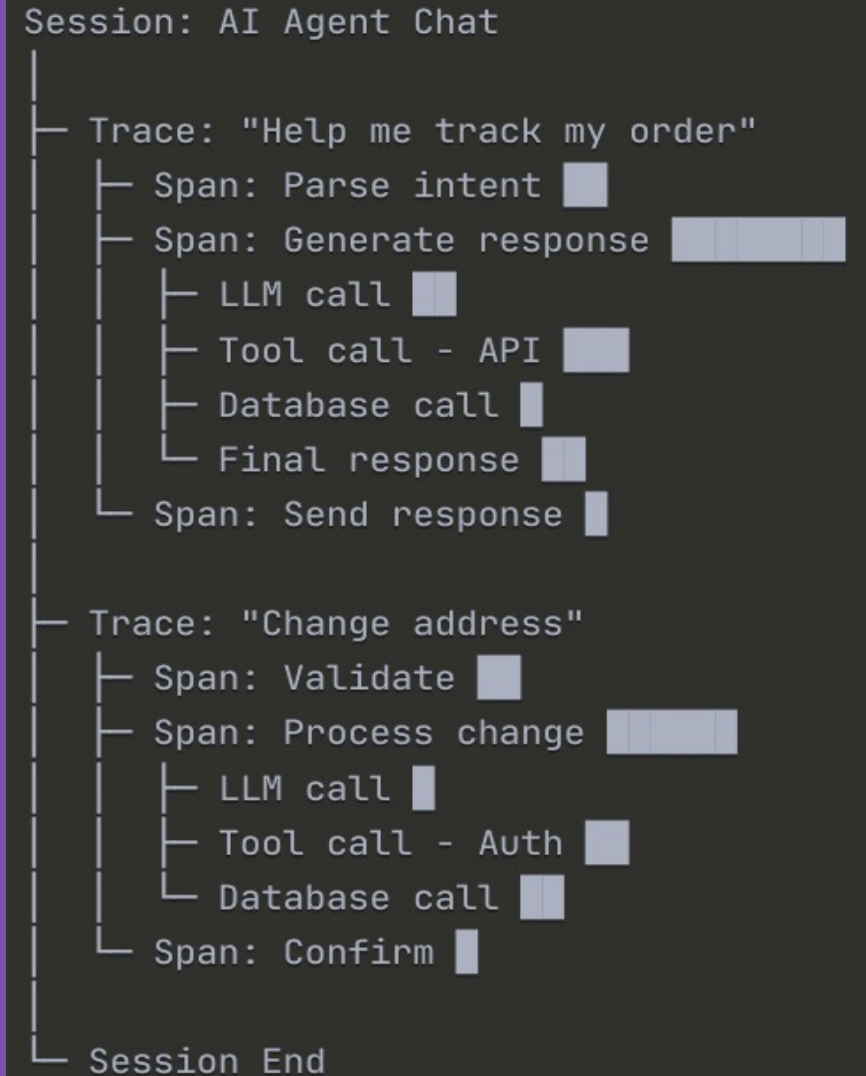


Third-party observability dashboards



Key Concepts

- Session: 사용자와 에이전트 간의 전체 상호작용 기간. 시작부터 종료까지의 모든 대화와 작업을 포함하는 최상위 컨테이너
- Trace: 세션 내에서 하나의 완전한 요청-응답 사이클. 사용자 요청부터 최종 응답까지의 전체 워크플로우를 추적
- Span: 트레이스 내의 개별 작업 단위. 예: LLM 호출, 도구 실행, 메모리 검색 등 각각의 주요 작업
- Sub-Span: 스패ن 내의 더 세부적인 작업. 예: API 호출, 데이터베이스 쿼리, 토큰화 등 스패ンを 구성하는 세부 단계





Amazon Bedrock AgentCore Interpreter

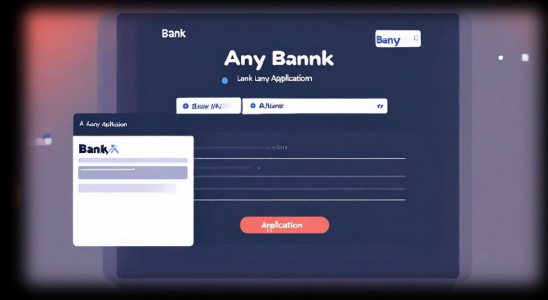
AgentCore Code Interpreter



정확도 향상



고급 분석 기능



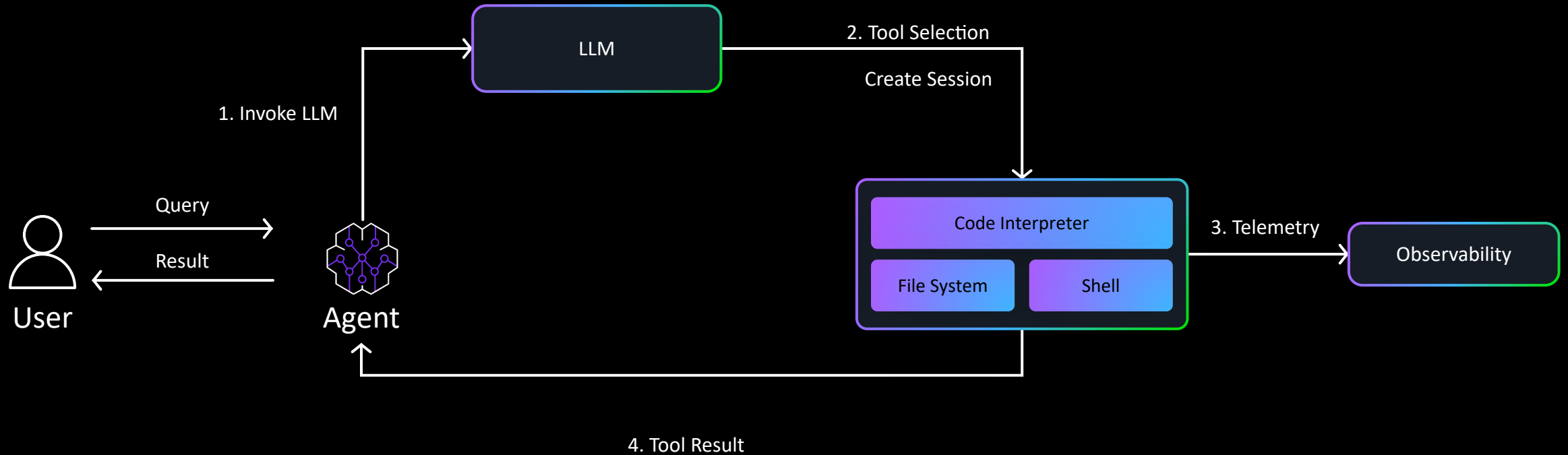
워크플로우 자동화

연산 도구 접근을 통해 대규모 언어 모델(LLM)이 한계를 극복하고 향상된 결과를 얻을 수 있습니다

대규모 데이터셋 분석 및 데이터 시각화, 보고서 생성

복잡한 작업을 포함한 워크플로우를 자동화하여 맞춤형 스크립트 작성 및 유지 관리 필요성 제거

AgentCore Code Interpreter



AgentCore Code Interpreter

- AI 에이전트가 격리된 샌드박스 환경에서 안전하게 코드를 실행할 수 있게 하는 도구

Fully managed serverless runtime

- 인프라 관리가 필요하지 않음
- 세션 실행 시간은 최대 8시간, 기본 설정은 15분
- 빠른 시작 속도 (낮은 레이턴시)

Enterprise grade security

- 안전한 샌드박스 환경에서 코드 실행
- 설정 가능한 네트워크 접근 제어: 외부 접근 없음 또는 공용 네트워크 선택 가능

Data Processing

- API를 통한 파일 업로드는 최대 100MB까지 지원
- S3 통합을 통해 최대 5GB의 대용량 파일 처리 가능
- CSV, Excel, JSON 등 구조화된 데이터 형식 처리 지원

Pre-loaded Runtime & Libraries

- 여러 프로그래밍 언어 지원: Python, Javascript, Typescript

Observability

- CloudTrail 로그 기록
- CloudWatch 지표 모니터링

Runtime, Memory 등 다른 서비스와의 통합



Amazon Bedrock AgentCore Browser

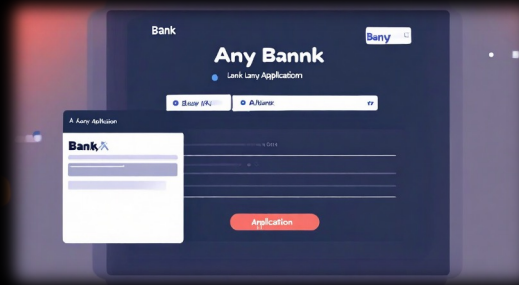


AgentCore Browser



웹 탐색 및 상호작용

- 웹사이트와 사람처럼 상호작용하고 작업을 실행할 수 있게 합니다
- 예시: 자동화된 여행 예약, 온라인 구매 등



워크플로우 자동화

- 에이전트가 양식 작성 및 기타 복잡한 작업과 같은 워크플로우를 자동화하여 맞춤형 자동화 스크립트 작성 및 유지 관리 필요성 제거
- 예시: HR 온보딩 자동화, 보험 청구 제출 자동화 등



웹 데이터 수집 및 웹 크롤링

- 웹사이트에서 구조화되거나 비구조화된 데이터 추출
- 예시: 제품 세부 정보 수집, 가격 변동 추적, 경쟁사 조사 등

AgentCore Browser

- AI 에이전트가 웹사이트와 상호작용할 수 있게 하는 클라우드 브라우저

Fully managed serverless browser

- 인프라 관리 불필요
- 낮은 시작 지연 시간
- 세션은 최대 8시간까지 실행 가능, 기본값은 15분

Enterprise grade security

- 보안 격리
- 컨테이너화된 환경에서 실행
- 브라우저 세션은 임시적이며 사용 후 매번 초기화
- 안전한 자격 증명 처리

Observability

- 실시간 모니터링을 위한 라이브 스트리밍 URL
- 디버깅을 위한 세션 재생
- 모든 브라우저 명령을 CloudTrail에 광범위하게 로깅

Scalable

- 수백 개의 세션으로 확장 가능
- 최대 500개의 동시 세션 지원

Runtime, Memory, Observability 등 다른 서비스와의 통합

AgentCore Browser Tool – Live view on console

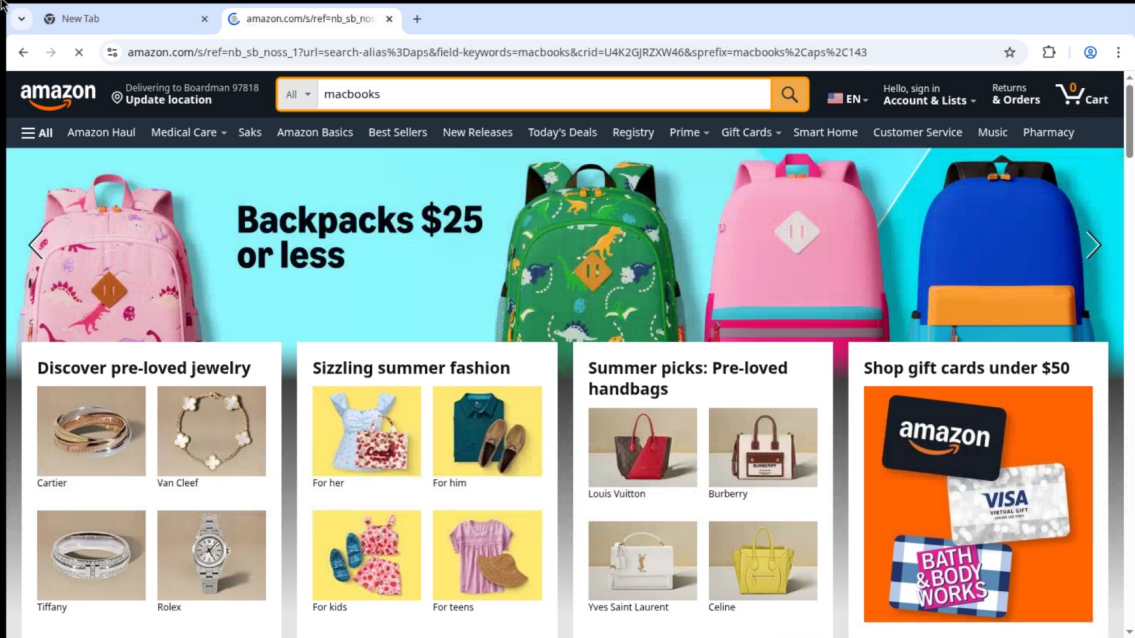
Amazon Bedrock AgentCore > Built-in Tools > AgentCore Browser Tool > 01K1F81Y4JJ0JB0F0J1NA3ZKH4

browser-session-42bab0bd - Session 01K1F81Y4JJ0JB0F0J1NA3ZKH4

Copy live URL Terminate session

Live browser session

Take control of session

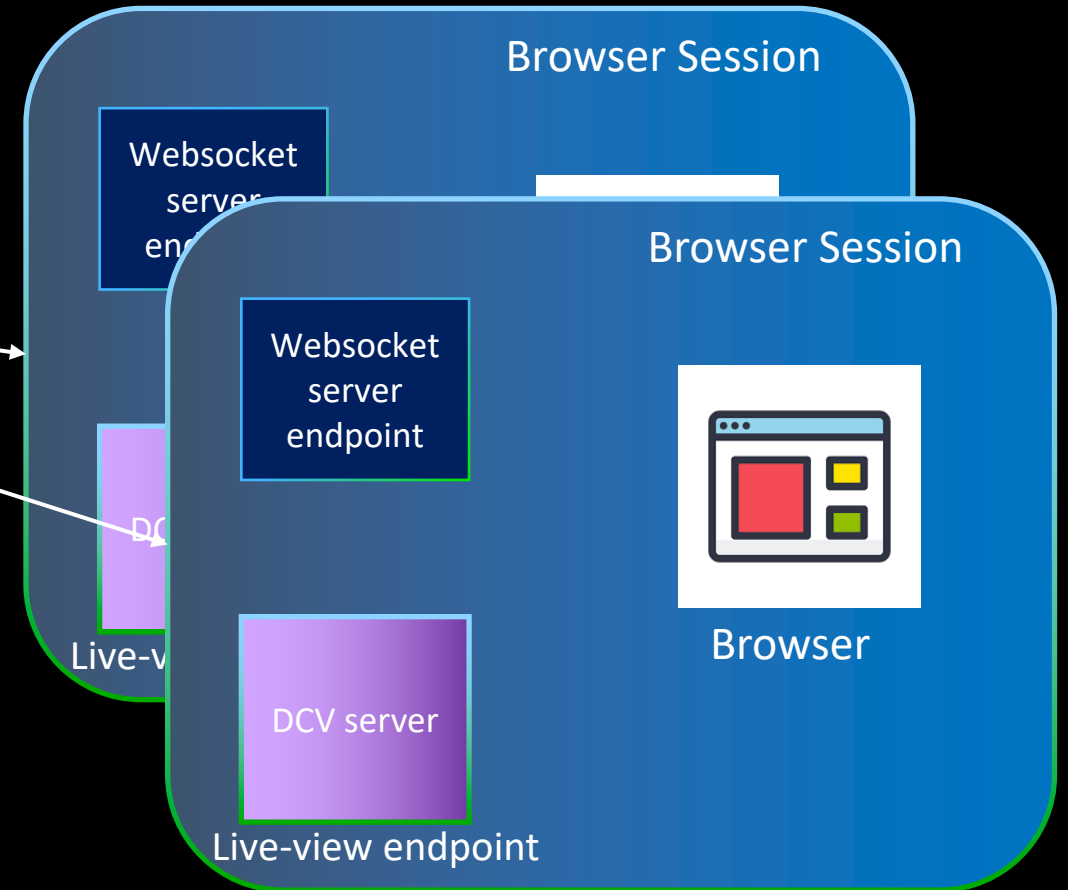


The screenshot displays the Amazon Bedrock AgentCore console interface. On the left, a sidebar menu includes sections for 'Discover' (Overview), 'Build and Deploy' (Agent Runtime, Built-in Tools, Gateways, Memory, Identity), 'Test' (Agent sandbox), and 'Docs and resources' (Amazon Bedrock, User guide, Bedrock Service Terms). The main area shows a 'Live browser session' for session ID 01K1F81Y4JJ0JB0F0J1NA3ZKH4. The browser window displays the Amazon website with a search for 'macbooks'. The page features a large banner for 'Backpacks \$25 or less' and several product recommendation tiles: 'Discover pre-loved jewelry' (featuring Cartier, Van Cleef, Tiffany, and Rolex), 'Sizzling summer fashion' (featuring clothing for her, him, kids, and teens), 'Summer picks: Pre-loved handbags' (featuring Louis Vuitton, Burberry, Yves Saint Laurent, and Celine), and 'Shop gift cards under \$50' (featuring Amazon, Visa, and Bath & Body Works). The console interface includes buttons for 'Copy live URL' and 'Terminate session' at the top right, and a 'Take control of session' button on the right side of the browser view.

AgentCore Browser Tool – Key concepts

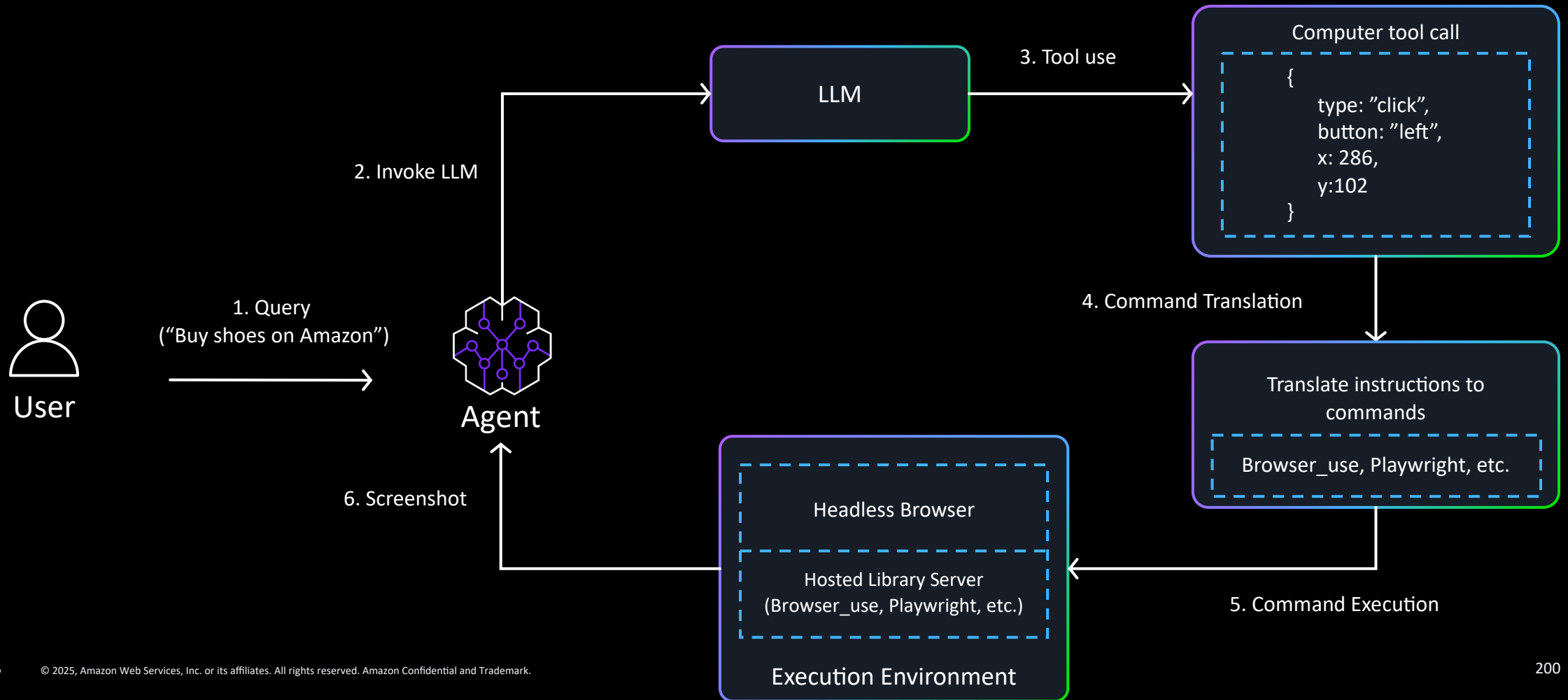


- ARN을 가진 리소스
- 네트워크, 녹화 설정 등과 같은 구성 설정 포함
- 여러 브라우저 세션과 연결
- 기본값: AgentCore Browser Tool



AgentCore Browser

- AI 에이전트가 웹사이트와 상호작용할 수 있게 하는 클라우드 브라우저





Amazon Bedrock Agent Marketplace

신규

AWS Marketplace의 AI 에이전트 및 도구

AWS Marketplace에서 직접 AI Agent와
툴(MCP)을 발견, 구매 및 배포

인기 있는 공급업체의 AI Agent, Agent Tool (MCP),
솔루션(SaaS)에 대한
중앙 집중식 카탈로그

MCP 및 A2A를 지원하는 API 기반 배포

Amazon Bedrock AgentCore에서 선별된 사전
구축된 에이전트 및 에이전트 툴(MCP) 배포
[Preview]

AWS Marketplace의 AI 에이전트와 도구

AI agents:

사전 구축된 특화 어시스턴트

Agentic tools:

AgentCore Runtime을 사용하여 도구와 에이전트 배포

AgentCore Gateway를 사용하여 도구와 에이전트를 에이전트 호환 도구로 변환

Professional services:

AI 에이전트 활용 사례를 지원하는 전문 서비스

The screenshot shows the AWS Marketplace homepage with a focus on AI Agents. The header includes the AWS Marketplace logo, a search bar, and navigation links. The main banner reads "INTRODUCING AI AGENTS" and "What can our agents do for you?". Below this, there's a text box with the example: "I want to automate customer support requests. We handle a large volume across different geos." followed by a right arrow. Three categories are listed: "Sales & marketing" (I need to analyze customer data and personalize email campaigns.), "Customer support" (I need to prioritize and route customer support tickets.), and "Finance & accounting" (I need to reconcile accounts and flag unusual transactions.). Each category has a right arrow. At the bottom, the text "Everything you need to succeed with AI agents" is followed by a list of benefits: "Speed to market", "Native AWS integration", "Enterprise-grade security", "Flexible deployment", and "One Platform for Billing, Procurement, and Governance", each with a plus sign to its right.

aws marketplace

Search

Sign in or Create a new account

About Categories Delivery Methods Solutions AWS IQ Resources Your Saved List Become a Channel Partner Sell in AWS Marketplace Amazon Web Services Home Help

INTRODUCING AI AGENTS

What can our agents do for you?

I want to automate customer support requests. We handle a large volume across different geos.

→

Sales & marketing
I need to analyze customer data and personalize email campaigns. →

Customer support
I need to prioritize and route customer support tickets. →

Finance & accounting
I need to reconcile accounts and flag unusual transactions. →

Everything you need to succeed with AI agents

| | |
|---|---|
| Speed to market | + |
| Native AWS integration | + |
| Enterprise-grade security | + |
| Flexible deployment | + |
| One Platform for Billing, Procurement, and Governance | + |

aws marketplace

Search

EnglishHello, assumed-role/AWS

AboutCategoriesDelivery MethodsSolutionsResourcesYour Saved ListBecome a Channel PartnerSell in AWS MarketplaceAmazon Web Services

Claude for Enterprise

Info

Sold by: Anthropic

View purchase options

OverviewPricingLegalUsageSupportSimilar productsReviews

2. Financial Analysis Solution: Specialized version for financial institutions with pre-built integrations to S&P Capital IQ, Daloopa, Morningstar, and other essential financial data sources for investment analysis and due diligence.

Please visit <https://www.anthropic.com/contact-sales> to request a private offer for a custom quote

Product Highlights:

- Enterprise-Grade Security: Single Sign-On (SSO), domain capture, audit logs, SCIM, custom data retention, and role-based permissions ensure the safety and compliance of your organization's data.
- Tool Integration: Connect Claude to your entire workflow by linking your essential tools and platforms. Through native, pre-built, and custom integrations - Claude understands your complete work context.

chats.

Details

Sold by: Anthropic

Categories: Content CreationSoftware DevelopmentResearch

Delivery method: Software as a Service (SaaS)

Deployed on AWS: Yes

Document Intelligence - AI Agent

Info

Sold by: Upstage

View purchase options

OverviewPricingLegalUsageSupportSimilar productsReviews

- Layout-Aware Parsing: Handles rotated pages, checkbox states, multi-page tables, and embedded visuals.
- Blazing Fast: Parses 100 pages in under 1 minute - up to 10x faster than alternatives.
- High Accuracy: Benchmarked with industry-leading performance on DP-Bench (TEDS score 93+).

Information Extract

Extract key data from documents as **structured JSON** - aligned to your custom or auto-generated schema. Ideal for automating workflows.

- Universal Document Compatibility: Extracts information from any document format, including complex PDFs, images, and Office documents.
- Schema-Agnostic Flexibility: Instantly adapts to different schemas, providing structured outputs without additional customization.
- Hidden and Implied Data Extraction: Surfaces not only visible text but also inferred values and line-item calculations.
- No Fine-Tuning Required: Works out-of-the-box without template

Details

Sold by: Upstage

Categories: Text/OCRDData AnalysisGenerative AI

Delivery method: API-Based Agents & Tools

Integration protocol: MCP

Type: MCP server

Deployed on AWS: No

aws marketplace

Search

EnglishHello, assumed-role/AWSReservedSS...

AboutCategoriesDelivery MethodsSolutionsResourcesYour Saved ListBecome a Channel PartnerSell in AWS MarketplaceAmazon Web Services HomeHelp

Brave Search API Pro

Info

Sold by: Brave

View purchase options

OverviewPricingLegalUsageSupportSimilar productsReviews

Usage information

Delivery details

API-Based Agents & Tools

API-Based Agents and Tools integrate through standard web protocols. Your applications can make API calls to access agent capabilities and receive responses.

Additional details

Usage instructions

MCPAPI

The Brave Search API can be conveniently used via Brave's official Brave Search MCP server.

The MCP server is available [for use locally](#), or as an [Amazon Bedrock AgentCore container image](#).

Articul8 LLM-IQ Agent

Info

Sold by: Articul8 AI

View purchase options

OverviewPricingLegalUsageSupportSimilar productsReviews

model for financial document summarization? or Compare Claude 3 and GPT-4 on reasoning tasks. Behind the scenes, it runs precision-tuned tests using multiple prompt variations and decoding strategies to simulate realistic workflows.

With actionable insights delivered through a professional-grade API, LLM-IQ Agent API enables intelligent decision-making at every stage of the GenAI lifecycle. Development teams can embed the API directly into inference workflows to power real-time model selection and dynamic prompt routing, automatically choosing the best-fit model for each user query. Procurement and vendor management functions gain standardized metrics for evaluating LLM providers, while engineering teams can offload the burden of framework development. For regulated industries, the API offers audit-ready evaluations aligned to compliance standards and domain-specific requirements. With LLM-IQ, enterprises gain a trusted layer of evaluation and transparency to support retrieval-augmented generation (RAG), multi-agent orchestration, and large-scale model deployment strategies.

Details

Sold by: Articul8 AI

Categories: Generative AIQuality AssuranceObservability

Delivery method: API-Based Agents & Tools

Type: Agent

Supported services: Amazon Bedrock AgentCore - Preview

Deployed on AWS: Yes

Amazon Bedrock



Open source

Protocols:

MCP, A2A (coming soon)

Frameworks:

CrewAI, LangGraph,
Strands Agents



Amazon Bedrock



AgentCore



Runtime



Identity



Gateway



Code interpreter



Memory



Browser tool



Observability

Model Capabilities



Model access



Cost and performance
optimization



Customization



Guardrails

Knowledge Bases

Thank you!

하남봉

nambong@amazon.com

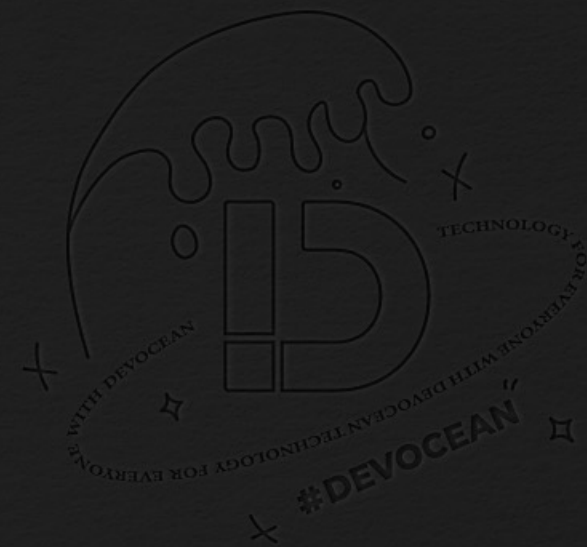




T멤버십 사례로 보는

Aurora MySQL DB의 HLL에 대한 이해

SK텔레콤 조진구



목차

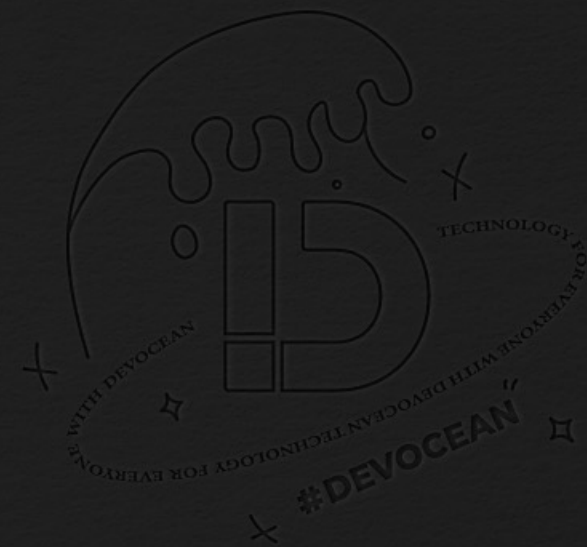
01 DB 장애가 났어요

02 HLL과 MVCC

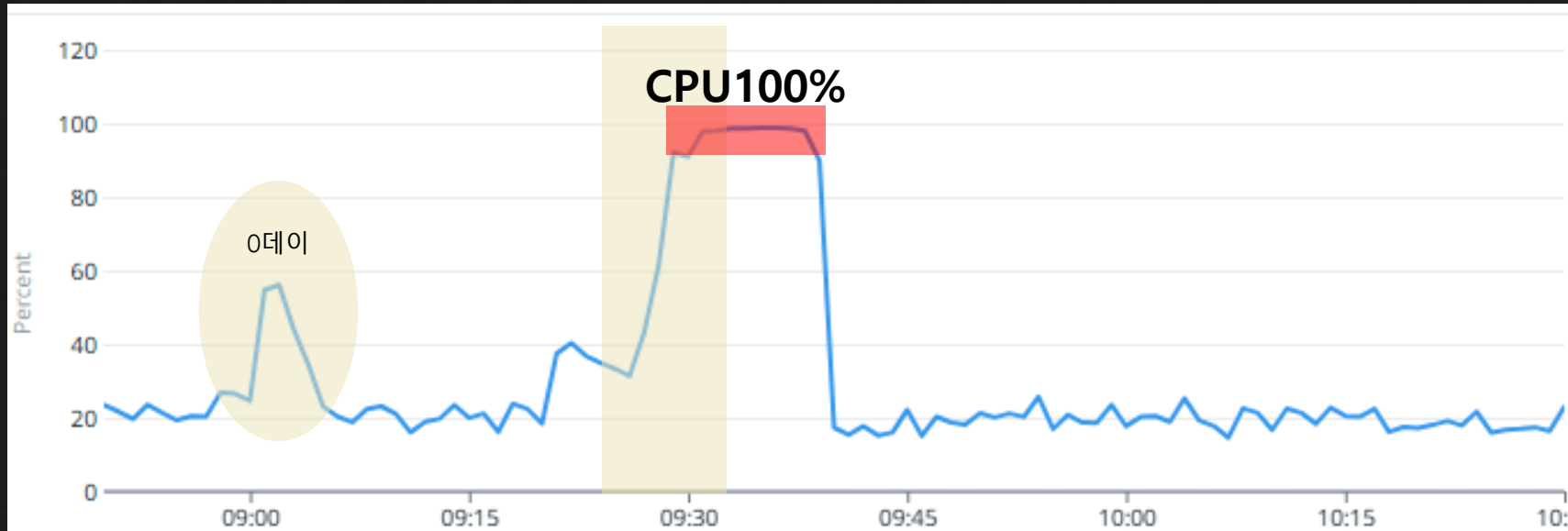
03 Reader DB를 쓰는데 Writer DB에 영향이?

04 HLL, 어떻게 관리할까요

05 결론



01 DB장애가 났어요



데이터베이스 로드

슬라이스된 기준

대기

막대 유형

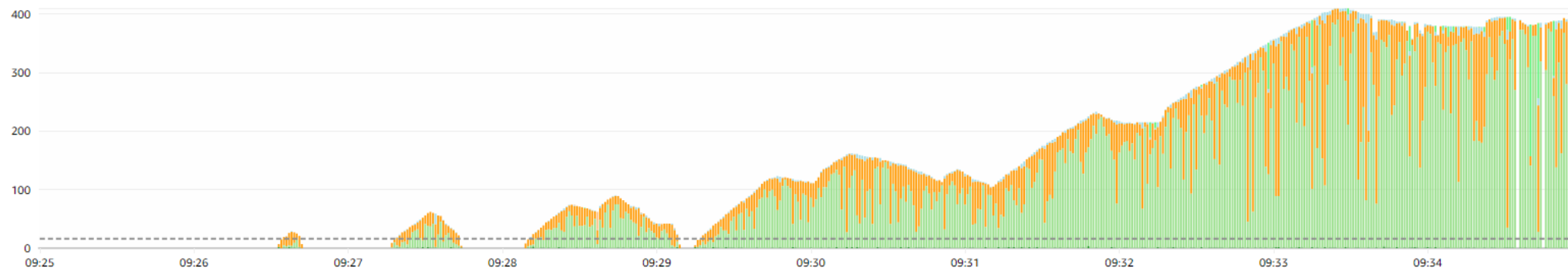
선 유형

표 유형

☒ 최대 vCPU 표시

범위:sql: SELECT COUNT (`onetimebar0\_`.`id`) A... x

평균 활성 세션(AAS)

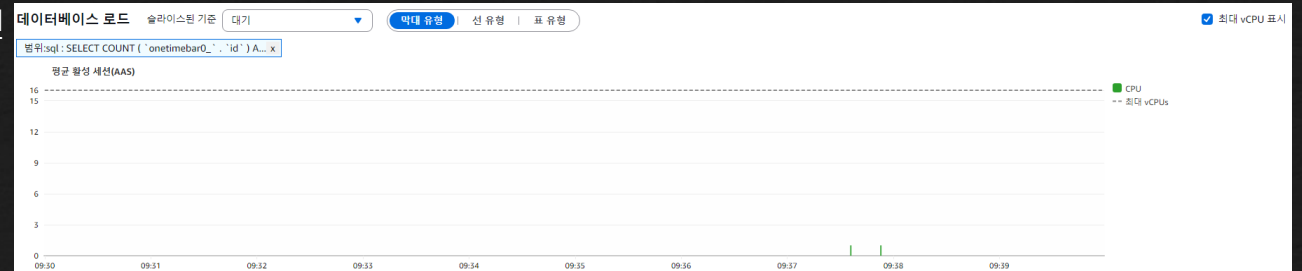


- wait/synch/mutex/innodb/fil_system_
- wait/synch/mutex/innodb/sync_array_
- wait/synch/mutex/innodb/buf_pool_LR
- wait/synch/sxlock/innodb/trx_purge_
- wait/synch/sxlock/innodb/index_tree
- wait/io/table/sql/handler
- wait/synch/sxlock/innodb/hash_table
- CPU
- 최대 vCPUs

01 DB장애가 났어요 - 이력을 보니..

- 평소에는 부하가 거의 없는 쿼리
- 어느 순간부터 쿼리 시 CPU부하가 점차 증가

3일 전



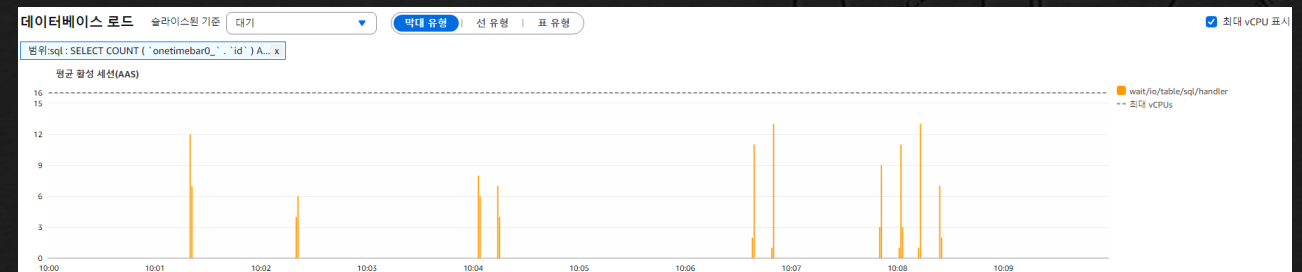
2일 전 7시



2일 전 8시

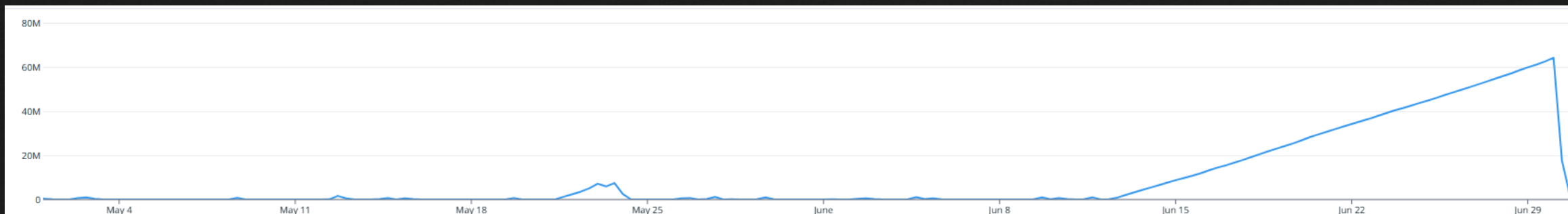


2일 전 10시



01 DB장애가 났어요 - 원인과 조치

- 장애 대응 중 배치 세션에서 오류로 인해 쿼리 하나가 장기로 비정상 실행중인 것을 발견
- 해당 세션을 죽인 후 HLL이 감소되며 부하 해소
- HLL은 65M(6,500만건)까지 증가했음
- 평소 HLL은 1M(100만건) 미만

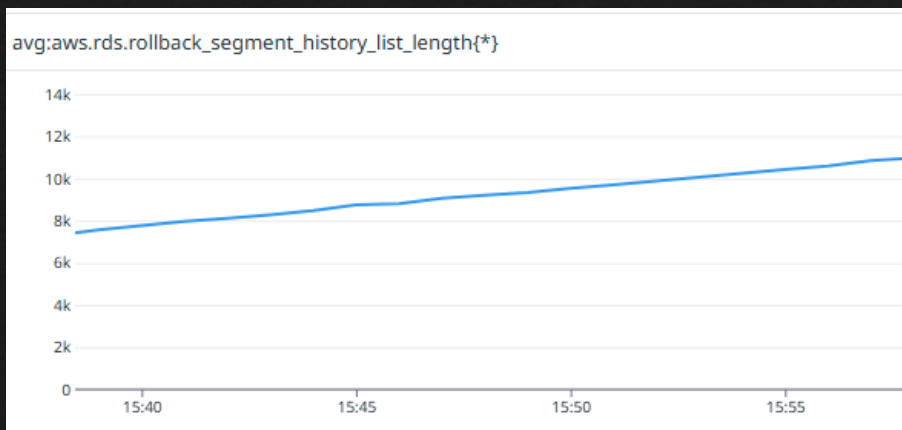
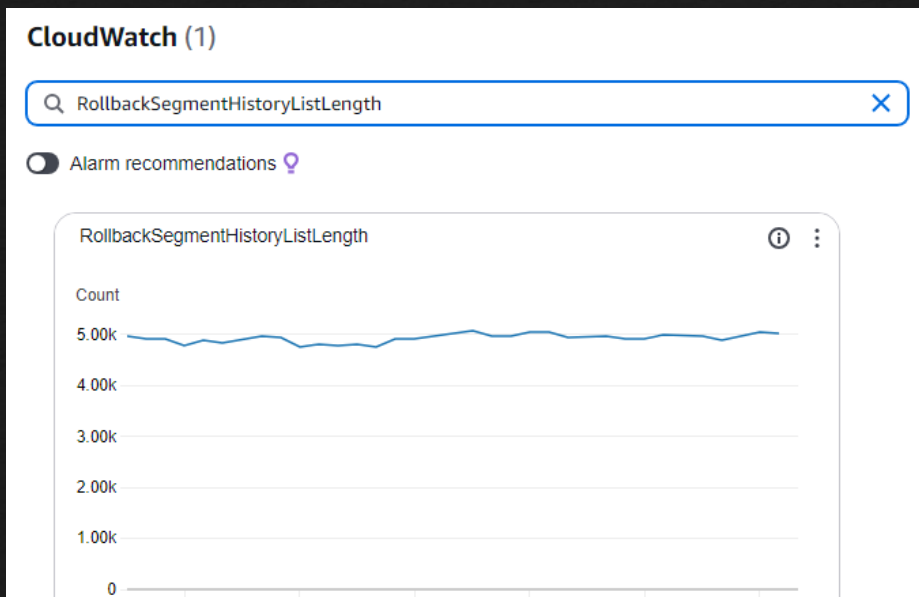


- 💡 모니터링을 통해 HLL 관리가 필요합니다



01 DB장애가 났어요 - 어떻게 관리하나요

① 모니터링



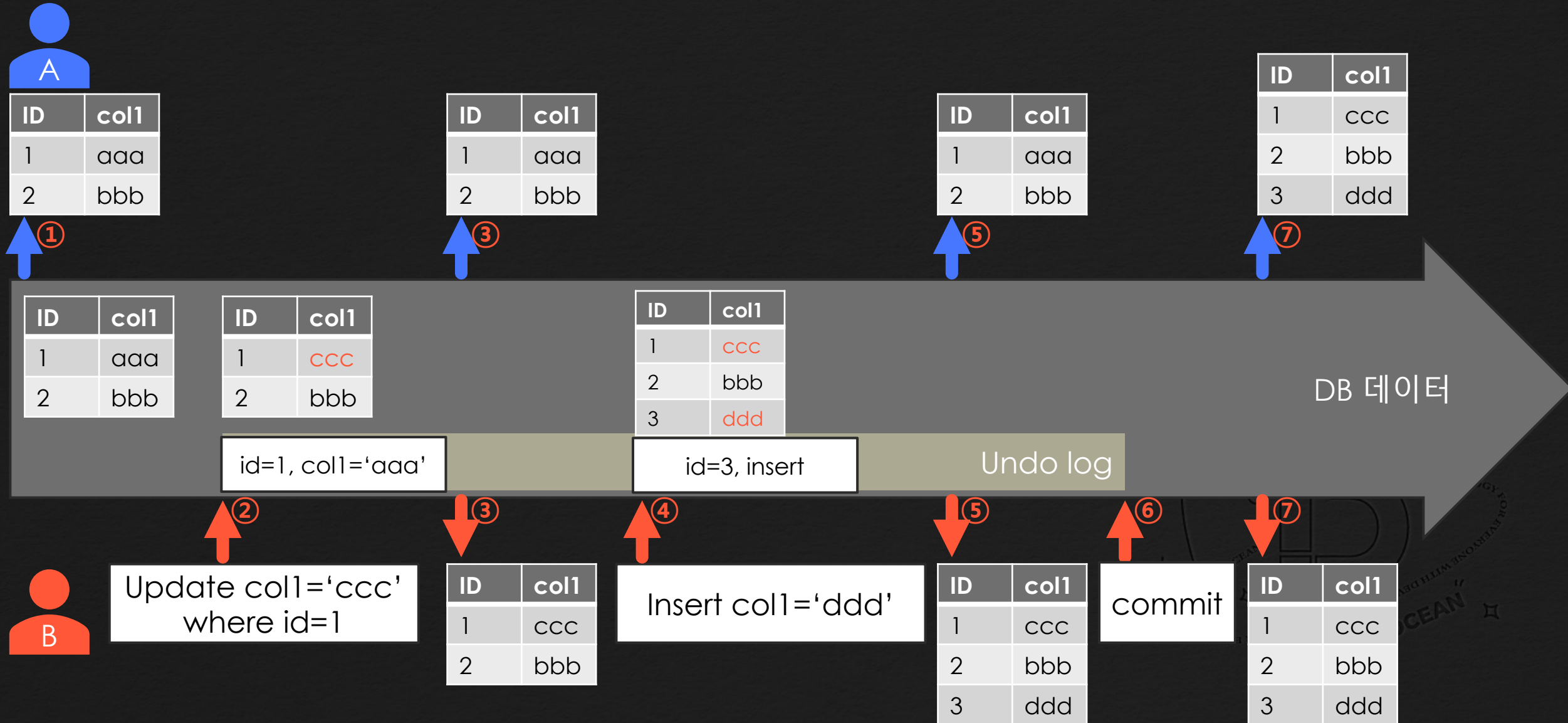
② 문제 세션 확인

```
SELECT * FROM  
information_schema.processlist  
ORDER BY time DESC
```

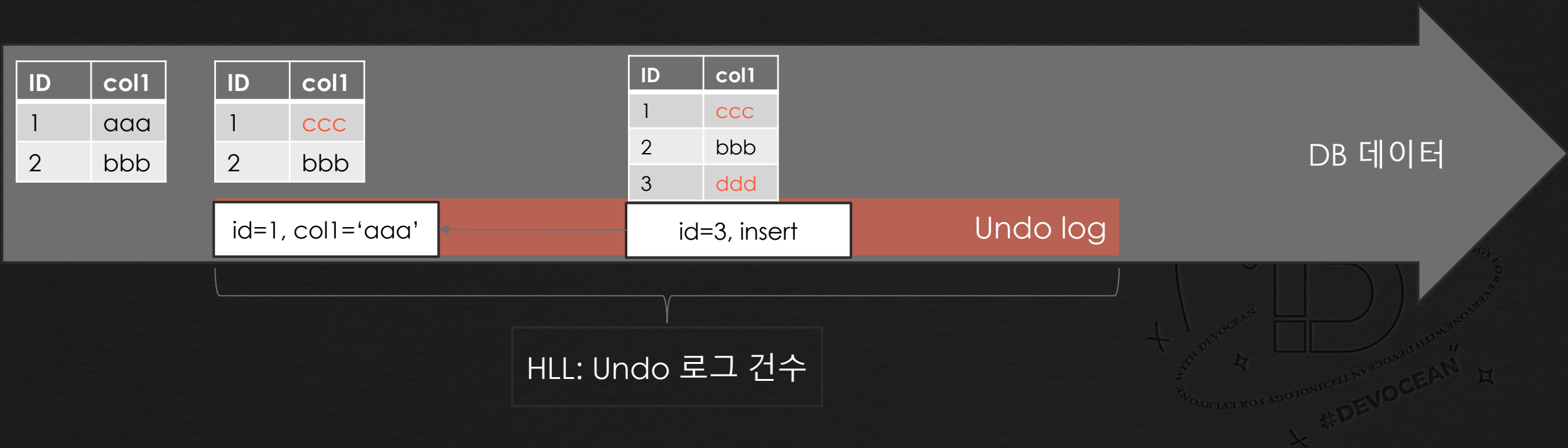
③ 문제 세션 제거

```
CALL mysql.rds_kill(세션ID)
```

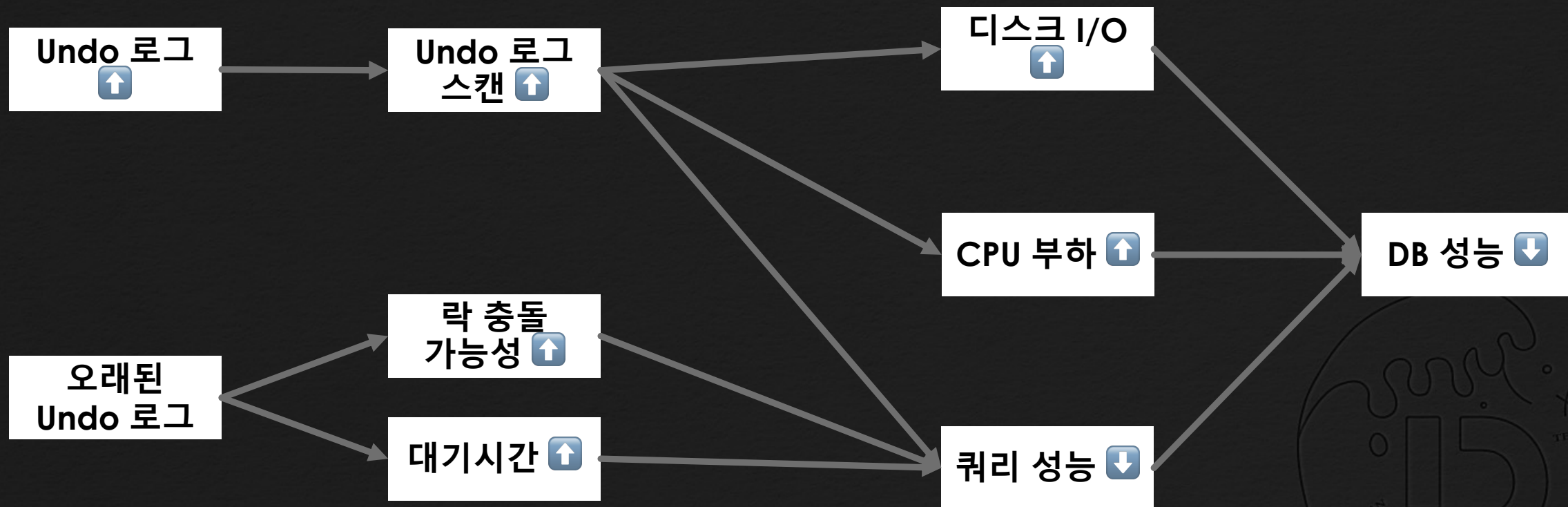
02 MVCC와 HLL - MVCC(Multi Version Concurrency Control)



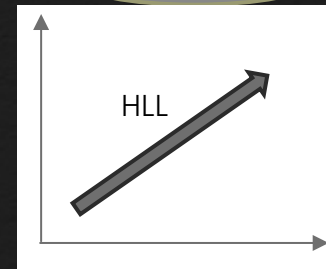
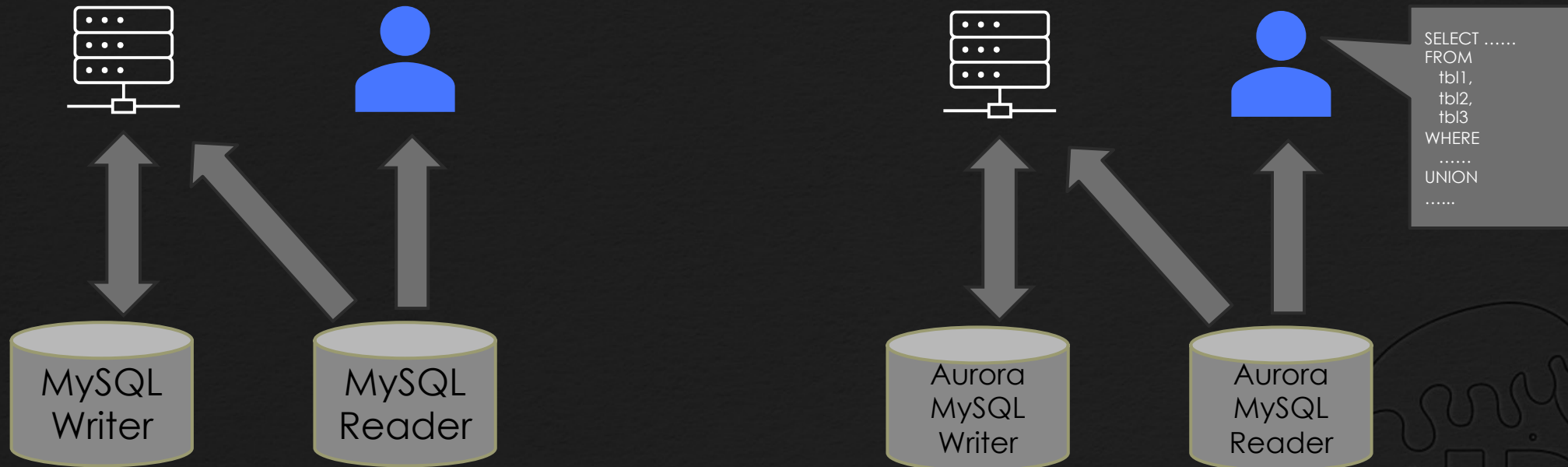
02 MVCC와 HLL - HLL(Rollback segment History List Length)



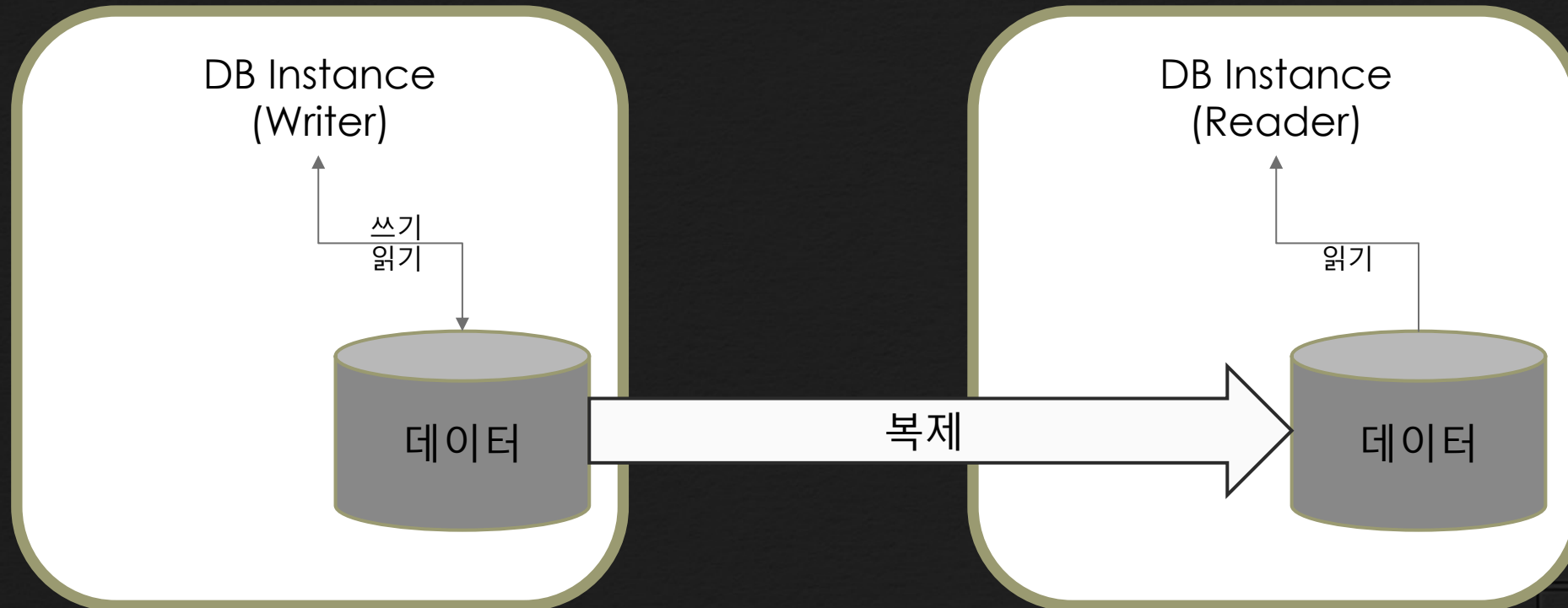
02 MVCC와 HLL - HLL이 증가하면



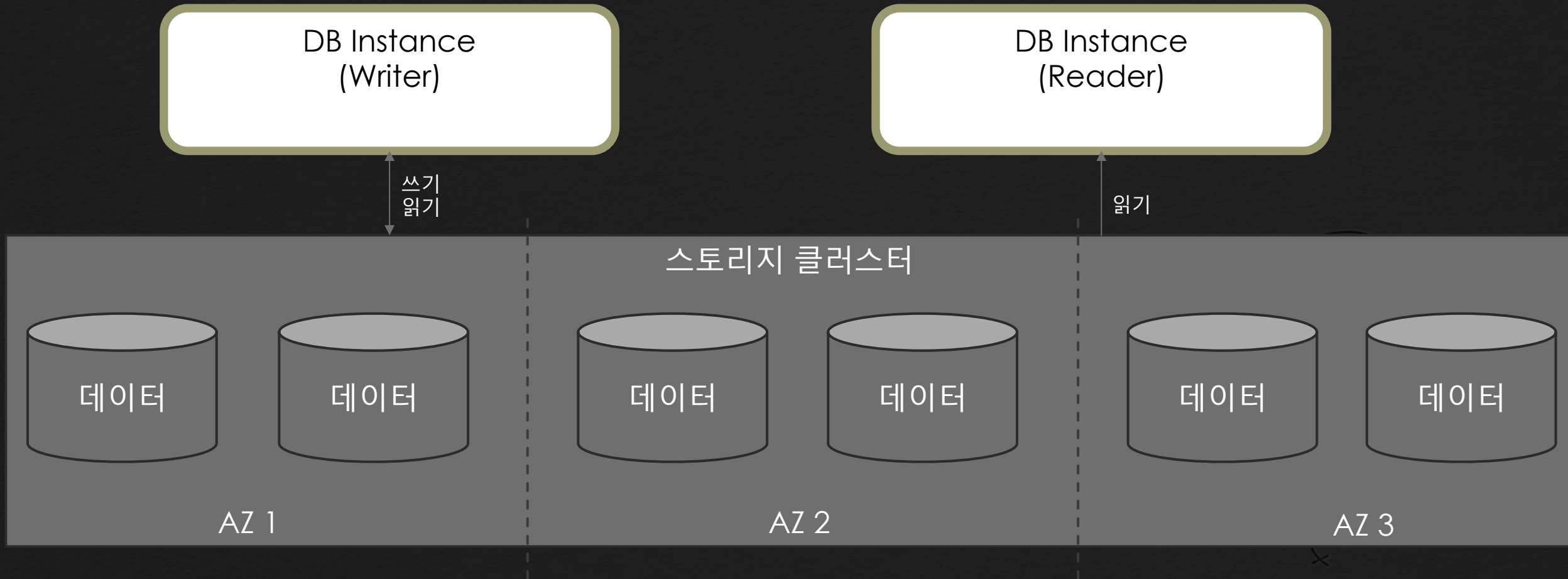
03 Reader DB를 쓰는데 Writer DB에 영향이?



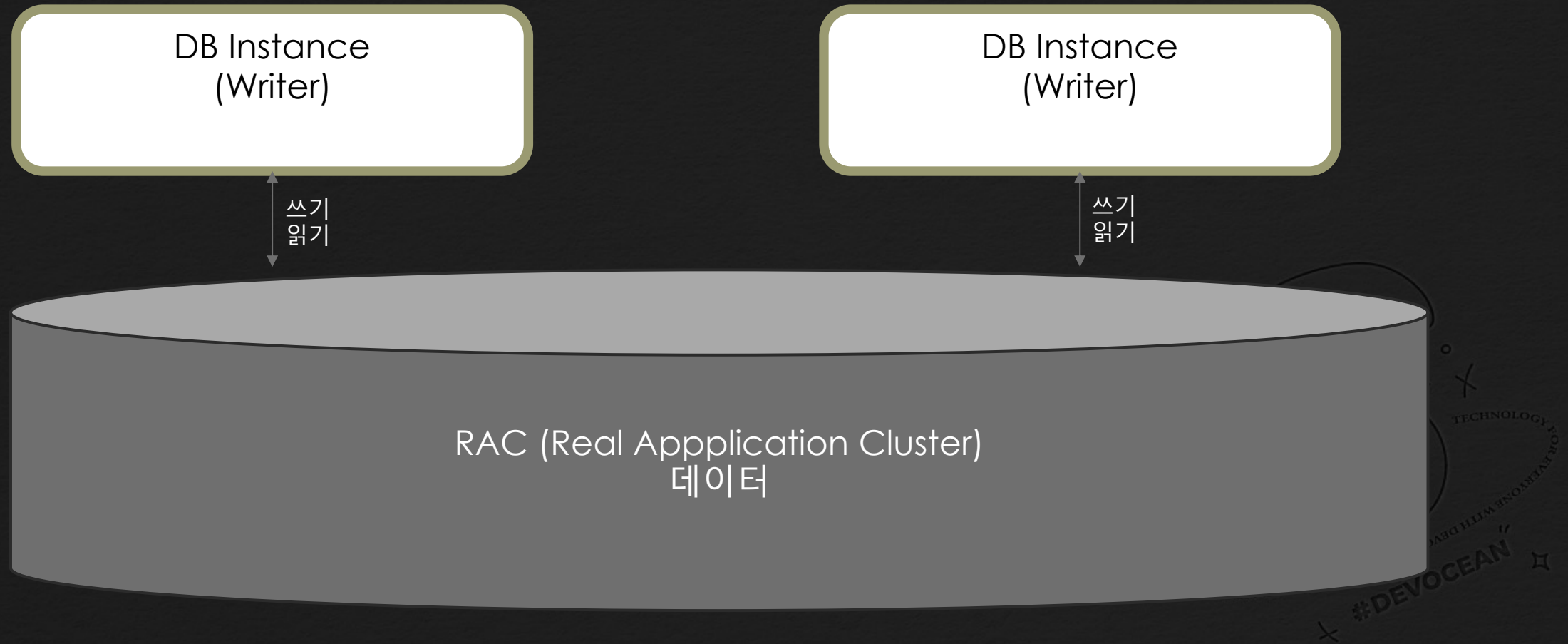
03 Reader DB를 쓰는데 Writer DB에 영향이? - MySQL 저장구조



03 Reader DB를 쓰는데 Writer DB에 영향이? - Aurora MySQL 저장 구조

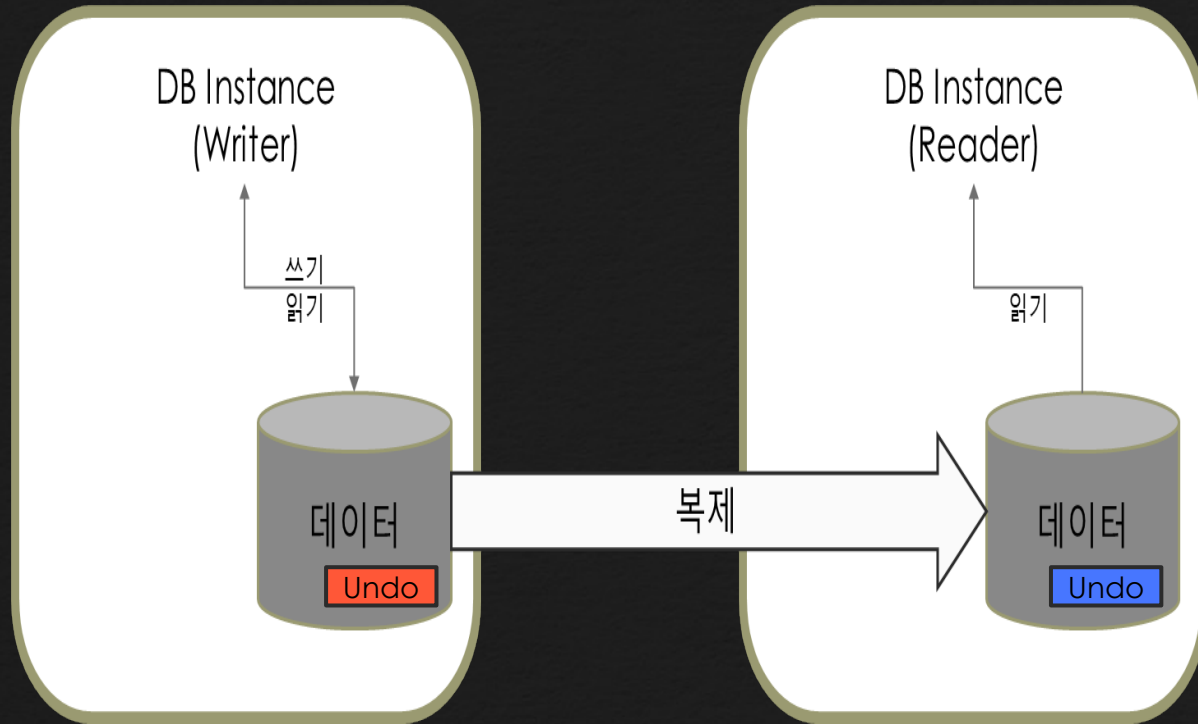


03 Reader DB를 쓰는데 Writer DB에 영향이? - 오라클 저장 구조(참고)

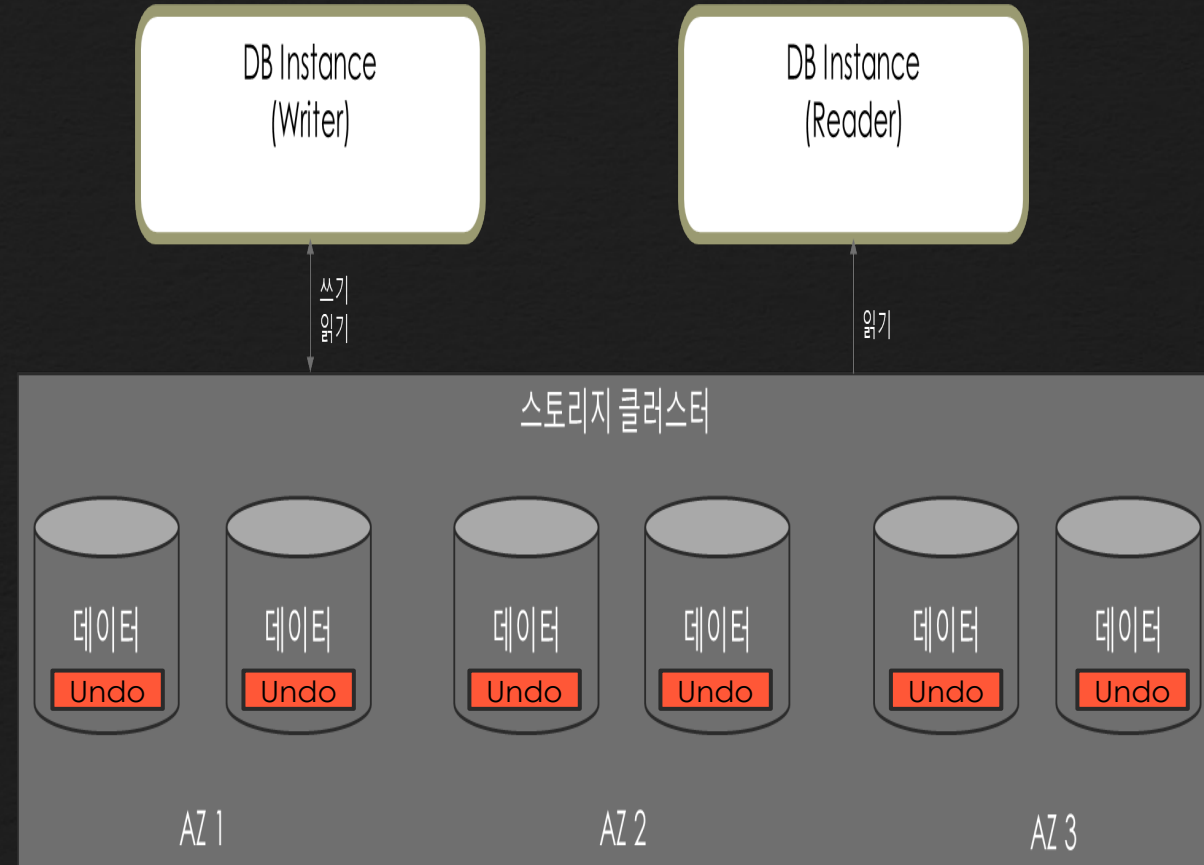


03 Reader DB를 쓰는데 Writer DB에 영향이?

MySQL

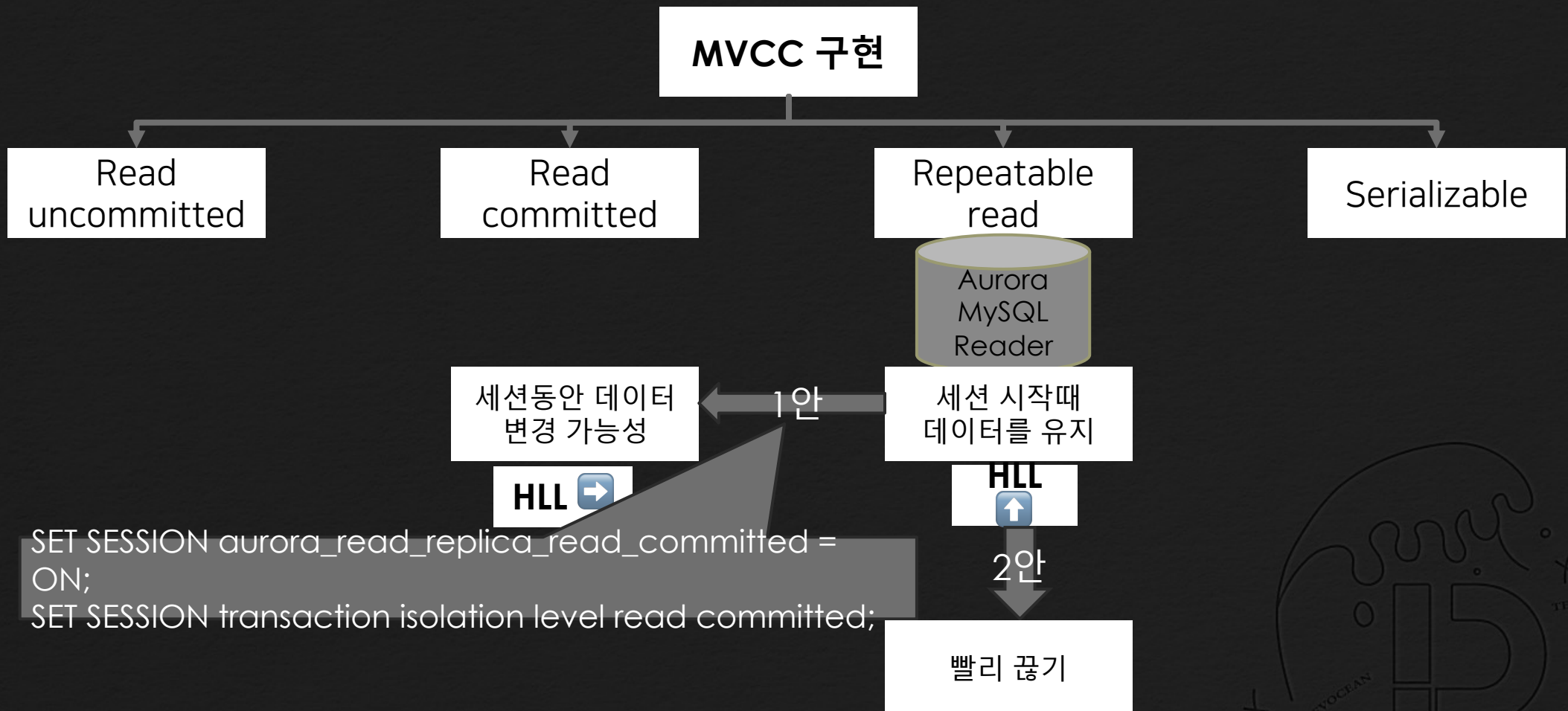


Aurora MySQL



- 💡 Reader DB의 쿼리가 Writer DB의 HLL을 증가시킬 수 있다

04 HLL, 어떻게 관리할까요

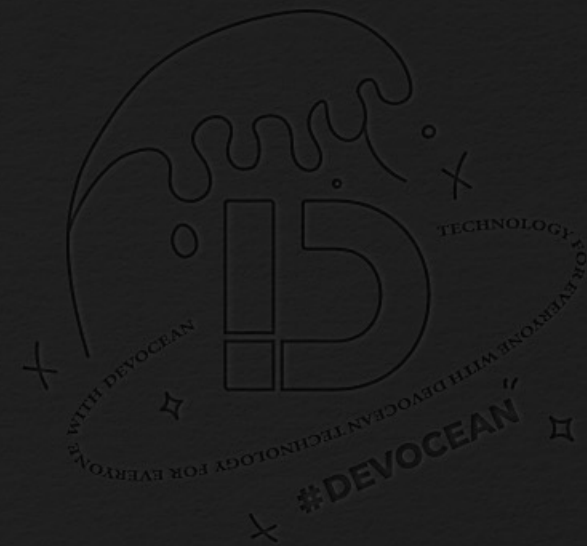


- 💡 HLL 관리를 위해 DB 세션은 짧게 써야한다



05 결론

- 💡 HLL이 많이 커지면 DB 성능이 하락하니 관리가 필요하다
- 💡 Aurora MySQL에서는 Reader DB에서의 쿼리가 Writer DB의 HLL에 영향을 줄 수 있다
- 💡 HLL 관리를 위해 DB 세션은 짧게 써야한다



감사합니다



AI와 함께 일하는 시대, 에이전트와 자동화 기술

새로운 개발 파트너: Q Developer와 Kiro로 여는 개발 자동화의 시대

최우형 (whchoi@amazon.com)

Principal Solutions Architect, AWS





Part 1. **Agentic AI** 기반 개발 도구 소개

"어떻게, 그리고 누가 더 나은 결정을 내릴 것인가?"

Gen AI가 코드를 넘어 결정을 만드는 구조로 진입한 시대

오늘날 개발자의 현실 : 적용은 머리로, 실행은 손으로

좋은 아이디어는 많지만, 실행으로 이어지지 않음



아이디어



실행의 난이도

반복적이고 비효율적인 업무

- ✓ 테스트는 해야 하지만, 손이 안가는 PR 리뷰
- ✓ 리팩토링과 문서화는 늘 뒤로 밀림.

사례: 국내 이커머스 클라우드팀

- ✓ 표준화된 Runbook 작성의 어려움.
- ✓ 서로 다른 IaC 코드 작성 방식

생성형 AI가 코드를 작성하는 시대 - 도전과 변화

1

GenAI 기반의 코드 작성

2

속도보다 방향
협업보다 일관성

3

Agentic AI:
코드 한 줄을 넘어
결정(decision)에 참여



Vibe Coding vs Agentic AI 소개

Vibe Coding

핵심 목적

몰입 기반의 코드생성 지원

AI 주도성

낮음 (반응형)

적용 범위

IDE 중심의 실시간 코드 생산성

사용자 유형

프론트엔드/백엔드 개발자, 학생, 디자이너

Agentic AI

목표 중심의 자율 실행

높음 (질문하고, 실행하고, 중단도 함)

복합적인 개발/운영/문서 작업 전반

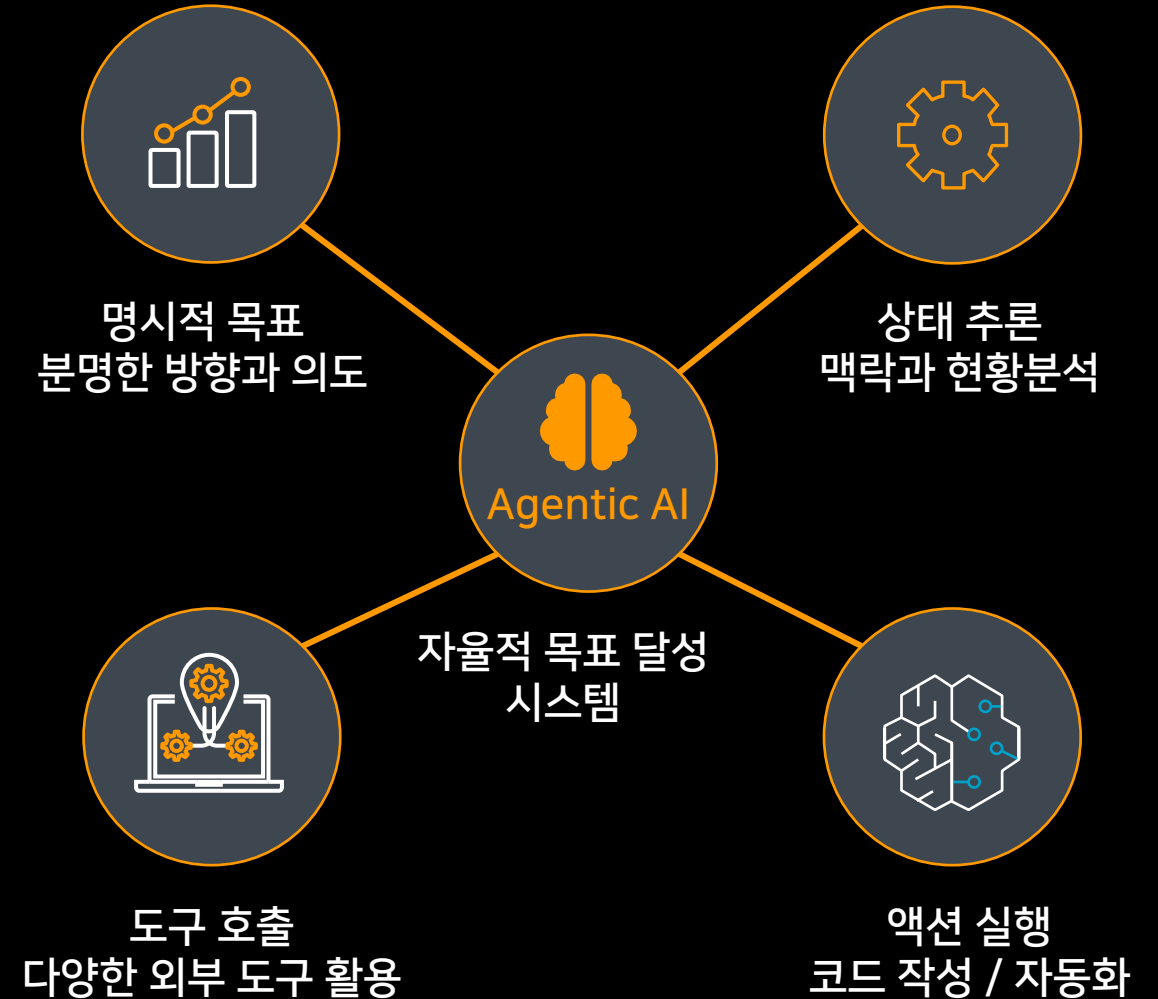
DevOps, SRE, 풀스택 개발자

"어떻게, 그리고 누가 더 나은 결정을 내릴 것인가?"

Gen AI가 코드를 넘어 결정을 만드는 구조로 진입한 시대

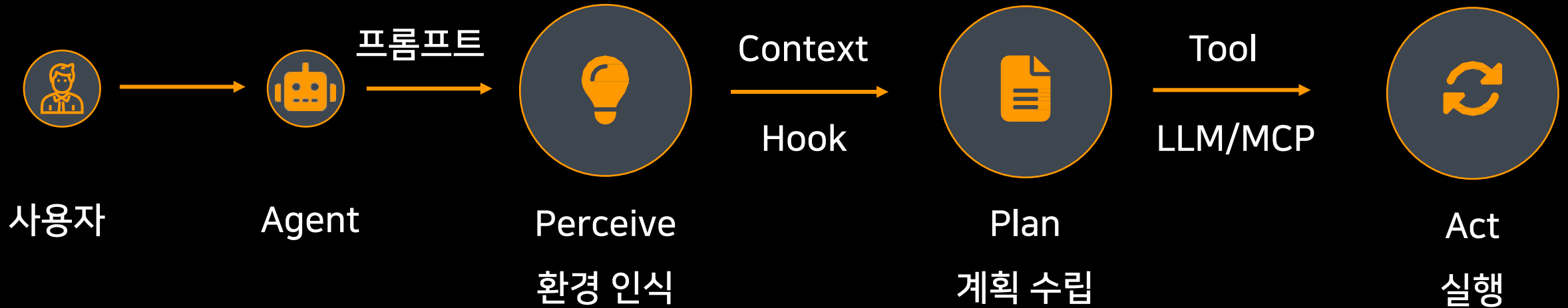
Agentic AI 소개

- ✓ **일관된 목표에** 따라 상태를 파악하고,
주도적으로 Tool과 Action을 실행하는 AI
- ✓ 기존 GenAI의 **수동적 지원**에서 벗어나,
자율적 결정과 실행이 가능한 단계 진화
- ✓ **Context(맥락) 이해 → 의도 파악 → 액션 실행**
→ 결과 평가 의 완전한 사이클을 스스로 실행



Agentic AI 개발도구

단순 LLM (Large Language Model)을 넘어, 실행하는 AI



단순 보조 개발자 + 인턴

- ✓ 단순 질문 답변
- ✓ 지시 후 실행 확인 필요



완벽한 시니어 동료

- ✓ 실행 가능한 계획 수립
- ✓ 스스로 판단하고 직접 실행



개발자 및 DevOps 에서의 Agentic AI 흐름

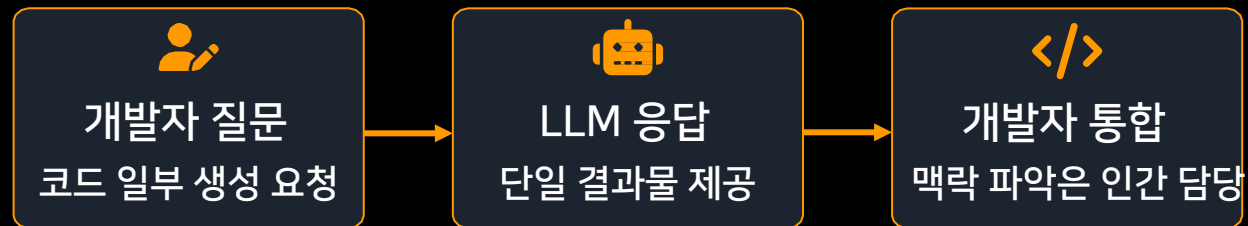
🕒 **과거** : LLM 기반의 보조 도구로 활용
단편적 코드 생성, 질문과 답변 중심

➔ **현재/미래** : 목표 기반 자율 실행 구조
명시적 목표 → 상태 추론 → 도구 호출 → 자동반복 실행

✅ **핵심적인 변화** : 단순 코드 생성에서 탈피,
결정 흐름 자체를 자동화

💡 **대표 도구**: Amazon Q Developer, Kiro

🕒 **과거**



➔ **현재/미래**





Part2. Amazon Q Developer/Kiro

IDE 개발도구를 넘어서 CLI, MCP, GitOps를 포함하는
조직 단위 Agentic 개발 인프라의 모범

Agentic AI 기반 개발도구 , Amazon Q Developer & Kiro

Amazon Q Developer



사용 용도

Non-opinionated AI 도우미
(플러그인 형태)

개발 방식

기존 개발 흐름 유지하면서
AI로 코드/운영 자동화 지원

기능 특화

Q CLI, IDE 플러그인, MCP 통합
(자연어 기반 코드, 리팩터링, 배포, 비용 분석 등)

도입 방식

IDE 플러그인, AWS 콘솔, CLI 등

적합 대상

기존 도구를 유지하며 점진적으로
AI를 도입하고자 하는 조직

Kiro



Opinionated Agentic IDE
(방향성 있는 IDE)

Spec-driven development 중심
(구조화된 요구사항 작성 → 실행 가능한 태스크 자동화)

Agent Hooks, Spec Tasks, Vibe 기반 협업
(AI와의 구조화된 협업)

IDE 플러그인

새로운 워크플로우를 수용할 준비가 된 팀
AI 기반 구조화된 개발 프로세스 실험 중인 조직



Agentic AI 기반 개발도구 , Amazon Q Developer & Kiro

Amazon Q Developer



Kiro



사용 용도

Non-opinionated AI 도우미
(플러그인 형태)

Opinionated Agentic IDE
(방향성 있는 IDE)

개발 방식

기존 개발 흐름 유지하면서
AI로 코드/운영 자동화 지원

Spec-driven development 중심
(구조화된 요구사항 작성 → 실행 가능한 태스크 자동화)

기능 특화

Q CLI, IDE 플러그인, MCP 통합
(자연어 기반 코드, 리팩터링, 배포, 비용 분석 등)

Agent Hooks, Spec Tasks, Vibe 기반 협업
(AI와의 구조화된 협업)

도입 방식

IDE 플러그인, AWS 콘솔, CLI 등

IDE 플러그인

적합 대상

기존 도구를 유지하며 점진적으로
AI를 도입하고자 하는 조직

새로운 워크플로우를 수용할 준비가 된 팀
AI 기반 구조화된 개발 프로세스 실험 중인 조직





Amazon Q Developer 핵심구성 요소

 **IDE 통합** - VSCode, JetBrains 연동으로 실시간 코드 해설, 문서 자동화, 테스트 생성 지원

 **Q CLI** : 자연어 기반의 채팅과 코드 생성, 신속한 기능 통합

 **MCP (Model Context Protocol)**: 맥락(Context) 기반 서버 연결로 Amazon MCP, Git, Shell, Playwright 등 조직 리소스 연동

Amazon Q Developer는 단일 도구가 아니 조직 전체 개발 생태계를 연결하는 통합 AI 생산성 플랫폼 입니다.



Amazon Q
Developer

IDE 통합

- ✓ VSCode, JetBrains, Eclipse 통합
- ✓ 실시간 코드 생성 및 해설
- ✓ 자동 문서화 및 테스트 생성

Q CLI

- ✓ 자연어 기반 채팅 및 코드 생성
- ✓ CLI 자동 완성
- ✓ 간편한 설치와 통합

MCP

- ✓ MCP 지원 및 백그라운드 로딩
- ✓ 도구, 프롬프트/리소스 활용
- ✓ 확장성/보안성/유연성





Amazon Q Developer 사례 - 국내 A사

Java 레거시 코드 마이그레이션:

☑️ JDK 8 → JDK 17 전환에 약 6MM 예상

☑️ Q Developer 기반 4주만에 완료

> 코드 분석 부터 리팩토링, 테스트 생성:

☑️ 63만줄 코드와 140개 라이브러리 자동 설치

☑️ Deprecated API 자동 수정, 단위 테스트 기반 리팩토링

Q CLI + IDE 연동으로 개발자 생산성 향상:

☑️ 기존 도구(VSCode) 그대로 사용하며 AI 기능 통합

☑️ AI 와 개발자의 협업으로 품질과 속도 확보

“ AI와 함께 하는 코드 현대화 : Q Developer 실전 적용 사례 ”



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

630,000+

코드 라인 수

4w

완료 기간

100%

단위 테스트 생성

140+

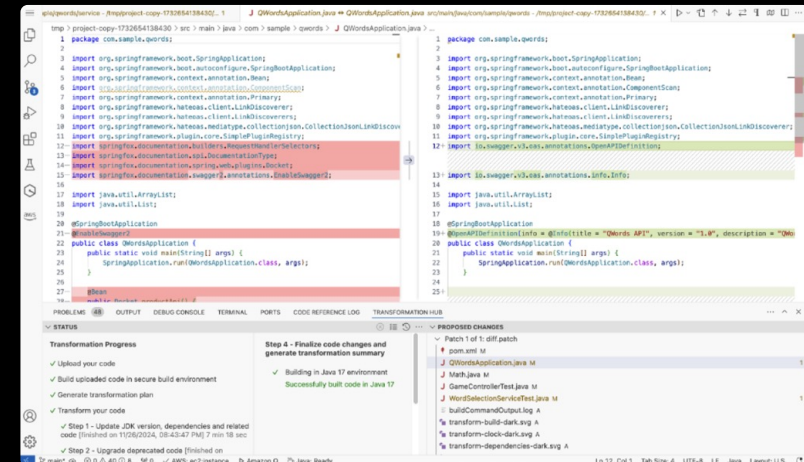
라이브러리 수

70%

자동화 비율

4x

생산성 향상





Amazon Q Developer 사례 - 국내 B사

🔗 운영 자산 전수 조사 자동화:

- ✔ K8s 리소스 최적화 자동 분석
- ✔ Idle 자원 자동 탐색 및 AZ 간 전송 시각화

> FinOps 기반 비용 분석 자동화:

- ✔ 이벤트 기간별 자원 비용 분석 리포트
- ✔ 이벤트 기간별 CloudFront 비용 상세 분석 리포트

📊 관측 기반 대시보드 자동 생성:

- ✔ Amazon EKS 노드 자원 사용량 시각화
- ✔ RDS 전체 사용량 대시보드 자동 구성

15+

자동 분석 항목

30sec

리포트 생성 시간

30K

월간 절감비용

5x

생산성 향상

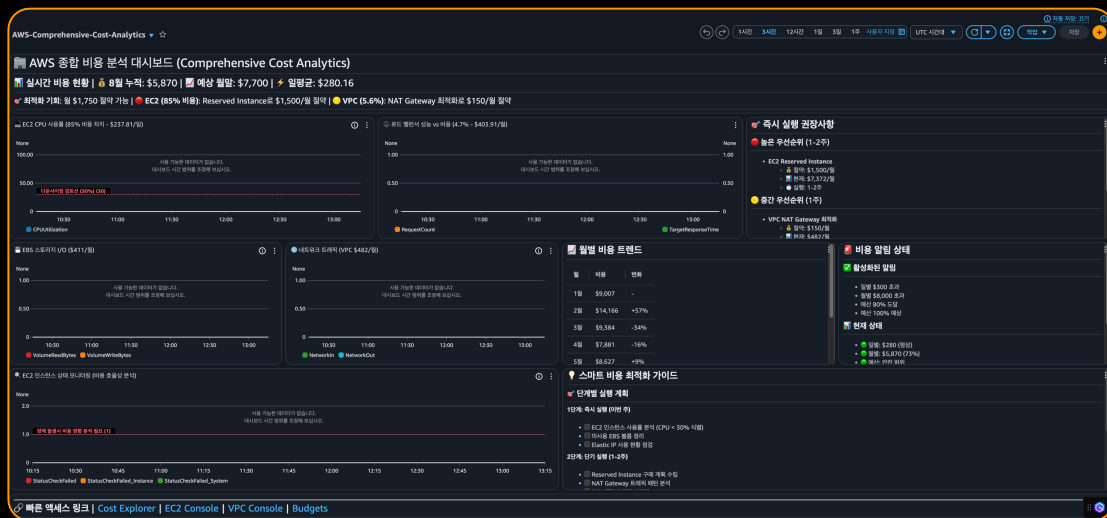
20x

MCP 도구

Amazon Q CLI를 통한 DevOps / SRE / FinOps 자동화 사례



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.





Amazon Q Developer 사례 – 국내 C사



코드 개발 및 리뷰 자동화:

- ✓ Go 기반 K8s 컨트롤러 개발 자동화
- ✓ Django API 구조 리팩토링 및 최적화
- ✓ Ansible 플레이북 리뷰 및 재구성

40%

코드 생산성 향상

27%

API 응답 개선

60%

tf Drift 감소

12+

운영효율화

>_ AWS 플랫폼 운영 고도화:

- ✓ Terraform/CDK 코드 자동화 및 최적화
- ✓ EKS 클러스터 리소스 최적화 및 분석
- ✓ CI/CD 파이프라인 재설계 및 개선

50%

로그 분석 시간 단축

30%

CI/CD 롤백 이슈 감소



문제 해결 및 FinOps 자동화:

- ✓ 로그 분석 및 트러블 슈팅
- ✓ 리소스 튜닝 및 비용 병목 분석, 보안 정책 리뷰/최소 권한 설계

Amazon Q Dev기반의 개발 효율화 및 운영 혁신 사례





Amazon Q Developer 사례 - 국내 D사



보안 컴플라이언스 자동화:

- Steampipe 기반 서비스 자산 수집 자동화
- ISMS-P 진단 자동화
- Security Hub 연동으로 컴플라이언스 상태 가시화
- 컨테이너 이미지(trivy) 스캔 및 Slack 알림 자동화



운영 및 분석 자동화:

- Boto3 + Q CLI로 라우팅 테이블 시각화
- Q CLI 기반 SecOps 분석 스크립트 템플릿화
- Presigned URL 기반 리포트 배포 자동화

Amazon Q CLI 기반 보안 및 네트워크 운영/개발 사례



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

85%

보안 점검 자동화율

60%

인증 심사 증적
준비 시간 단축

70%

보안 이슈 대응
시간 단축

80%

TGW 분석 시간

네트워킹 분석

AWS 계정 통합 분석 보고서 | 생성일: 2023년 06월 17일

← 이전 보고서로 돌아가기

AWS 계정 통합 분석 보고서 - 네트워킹 분석

네트워킹 아키텍처 종합 분석

분석 범위: VPC, 서브넷, 라우팅, 방화벽, 트랜зит 게이트웨이
분석 일자: 2023년 6월 17일
이전: ap-northeast-2

네트워킹 현황 개요

VPC 구성 현황

| VPC 이름 | VPC ID | CIDR 블록 | 상태 | 종류 | 보안 점수 |
|-------------|-----------------------|---------------|-----------|------------|-------|
| mgmtvpc | vpc-026c88967880323ac | 10.254.0.0/16 | Available | 관리자용 전용 | 10/10 |
| DMZVPC | vpc-0a033176a6b0759af | 10.11.0.0/16 | Available | DMZ/퍼블릭 영역 | 10/10 |
| VPC01 | vpc-0f642a4d468f32133 | 10.1.0.0/16 | Available | 워크로드 영역 1 | 10/10 |
| VPC02 | vpc-0a782375524f9004 | 10.2.0.0/16 | Available | 워크로드 영역 2 | 10/10 |
| Default VPC | vpc-0b248ec7e4e4b5bde | 172.31.0.0/16 | Available | 기본 VPC | 10/10 |

네트워킹 아키텍처 평가

| 평가 항목 | 점수 | 상태 | 세부 내용 |
|--------|-------|----|----------------------------|
| VPC 설계 | 10/10 | 우수 | 독립된 VPC 분리, 적절한 CIDR 범위 |
| 서브넷 구성 | 10/10 | 양호 | Multi-AZ 배치, 리플리케이션 팩트 |
| 라우팅 설계 | 10/10 | 우수 | Transit Gateway 활용 등용선 라우팅 |
| 보안 설계 | 10/10 | 양호 | 세분화된 보안 그룹, 최소 권한 원칙 |
| 태깅 | 10/10 | 우수 | 태깅 가능한 아키텍처 설계 |

VPC 아키텍처 상세 분석

보안 분석

AWS 계정 통합 분석 보고서 | 생성일: 2023년 06월 17일

← 이전 보고서로 돌아가기

AWS 계정 통합 분석 보고서 - 보안 분석

보안 아키텍처 종합 분석

분석 범위: IAM, IAM 역할, IAM 정책, IAM 그룹, IAM 사용자
분석 일자: 2023년 6월 17일
이전: ap-northeast-2

보안 현황 개요

보안 영역별 평가

| 보안 영역 | 점수 | 상태 | 주요 특징 |
|-----------|-------|----|-----------------------|
| 네트워킹 보안 | 10/10 | 양호 | VPC 분리, 방화벽, 보안 그룹 적용 |
| 데이터 보안 | 10/10 | 적당 | 적절한 암호화, 보안 그룹 적용 |
| IAM 및 액세스 | 10/10 | 적당 | 최소 권한 원칙, 보안 그룹 적용 |
| 로그/태깅 | 10/10 | 적당 | CloudTrail 활용, 로그 분리 |
| 전체 보안 | 10/10 | 적당 | 기본 보안 적용, 모니터링 |

현재 보안 수준: 7.0/10 (양호)

네트워킹 보안 분석

VPC 보안 아키텍처

보안 영역 평가

- 네트워킹: VPC 분리, 방화벽, 보안 그룹 적용
- 데이터: 암호화, 보안 그룹 적용
- IAM: 최소 권한 원칙, 보안 그룹 적용
- 로그/태깅: CloudTrail 활용, 로그 분리
- 전체 보안: 기본 보안 적용, 모니터링

보안 영역 상세

- 네트워킹: VPC 분리, 방화벽, 보안 그룹 적용
- 데이터: 암호화, 보안 그룹 적용
- IAM: 최소 권한 원칙, 보안 그룹 적용
- 로그/태깅: CloudTrail 활용, 로그 분리
- 전체 보안: 기본 보안 적용, 모니터링



"Amazon Q Developer는 조직 전체의 생산성을 끌어 올리는 AI 운영체제로 기업내에서 진화"



Amazon Q Developer



IDE를 넘어선 조직 전체 개발 프로세스를 지원하는 AI 기반 도구



CLI, MCP, GitOps 환경 전반에서 일관된 개발 흐름 구현



DevOps, QA, SRE, 기획/PM 전체 부서와 협업을 구현하는 도구



보안 통합, 감사/추적, 역할 분리를 통한 엔터프라이즈급 안정성 확보



Agentic AI 기반 개발도구 , Amazon Q Developer & Kiro

Amazon Q Developer



Kiro



| | | |
|-------|---|---|
| 사용 용도 | Non-opinionated AI 도우미
(플러그인 형태) | Opinionated Agentic IDE
(방향성 있는 IDE) |
| 개발 방식 | 기존 개발 흐름 유지하면서
AI로 코드/운영 자동화 지원 | Spec-driven development 중심
(구조화된 요구사항 작성 → 실행 가능한 태스크 자동화) |
| 기능 특화 | Q CLI, IDE 플러그인, MCP 통합
(자연어 기반 코드, 리팩터링, 배포, 비용 분석 등) | Agent Hooks, Spec Tasks, Vibe 기반 협업
(AI와의 구조화된 협업) |
| 도입 방식 | IDE 플러그인, AWS 콘솔, CLI 등 | IDE 플러그인 |
| 적합 대상 | 기존 도구를 유지하며 점진적으로
AI를 도입하고자 하는 조직 | 새로운 워크플로우를 수용할 준비가 된 팀
AI 기반 구조화된 개발 프로세스 실험 중인 조직 |



Spec 기반 Agentic 개발 플랫폼 – Kiro 소개

Spec 드리븐 기반 개발



- ✓ Requirement / Design / Task 기반
- ✓ 점진적 개발
- ✓ 구조화된 협업
- ✓ 파일 참조
- ✓ 단계별 방향성 수정
- ✓ 문서화 및 재사용성
- ✓ 품질 보장

반복되는 작업 자동화



- ✓ 이벤트 기반 Hook
- ✓ 생산성 향상 – 자동화 및 품질 유지
- ✓ 워크플로우 최적화

컨텍스트와 지침



- ✓ 팀의 지식과 표준을 체계화
- ✓ 작업 수행 방법
- ✓ 워크플로우 최적화
- ✓ 프로젝트 컨텍스트를 이해





Spec 기반 Agentic 개발 플랫폼 – Kiro 소개

SPECS

AGENT Hooks

AGENT STEERING

MCP SERVERS

SPECS 3단계

1 Requirements 2 Design 3 Task list

상태 메뉴 창

모델 선택

Spec: battle-tank-game

1 # 요구사항 문서 (Requirements Document)

2 ## 소개 (Introduction)

3 클래식 1990년대 스타일의 전투 탱크 게임을 HTML5 Canvas와 JavaScript로 개발합니다. 이 게임은 웹 브라우저에서 실행되며, 플레이어가 녹색 탱크를 조작하여 적 탱크를 파괴하는 액션 게임입니다. 순수 JavaScript와 HTML5 Canvas를 사용하여 가볍고 접근성이 높은 게임을 제작합니다.

4 ## 요구사항 (Requirements)

5 ## 요구사항 1: 플레이어 탱크 조작 시스템

6 **사용자 스토리:** 게이머로서, 방향키를 사용하여 탱크를 자유롭게 조작하고 싶습니다. 그래픽을 피하고 전략적으로 움직일 수 있습니다.

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

요구사항 문서 (Requirements Document)

소개 (Introduction)

클래식 1990년대 스타일의 전투 탱크 게임을 HTML5 Canvas와 JavaScript로 개발합니다. 이 게임은 웹 브라우저에서 실행되며, 플레이어가 녹색 탱크를 조작하여 적 탱크를 파괴하는 액션 게임입니다. 순수 JavaScript와 HTML5 Canvas를 사용하여 가볍고 접근성이 높은 게임을 제작합니다.

요구사항 (Requirements)

요구사항 1: 플레이어 탱크 조작 시스템

사용자 스토리: 게이머로서, 방향키를 사용하여 탱크를 자유롭게 조작하고 싶습니다. 그래픽을 피하고 전략적으로 움직일 수 있습니다.

승인 기준 (Acceptance Criteria)

1. WHEN 사용자가 ↑ 방향키를 누르면 THEN 탱크는 현재 바라보는 방향으로 전진해야 합니다

2. WHEN 사용자가 ↓ 방향키를 누르면 THEN 탱크는 현재 바라보는 방향의 반대로 후진해야 합니다

3. WHEN 사용자가 ← 방향키를 누르면 THEN 탱크는 반시계방향으로 회전해야 합니다

4. WHEN 사용자가 → 방향키를 누르면 THEN 탱크는 시계방향으로 회전해야 합니다

5. WHEN 사용자가 여러 방향키를 동시에 누르면 THEN 탱크는 해당 조합에 따라 대각선 이동을 해야 합니다

6. WHEN 탱크가 화면 가장자리를 벗어나면 THEN 반대편 가장자리에서 나타나야 합니다. (화면 래핑)

요구사항 2: 무기 시스템

사용자 스토리: 게이머로서, 스페이스바를 눌러 총알을 발사하고 싶습니다. 그래픽 적 탱크를 파괴할 수 있습니다.

승인 기준 (Acceptance Criteria)

1. WHEN 사용자가 스페이스바를 누르면 THEN 탱크가 바라보는 방향으로 총알이 발사되어야 합니다

2. WHEN 총알이 발생되면 THEN 총알은 직선으로 이동해야 합니다

3. WHEN 총알이 화면 가장자리에 도달하면 THEN 총알은 사라져야 합니다

4. WHEN 총알이 적 탱크와 충돌하면 THEN 적 탱크는 파괴되고 총알도 사라져야 합니다

요구사항 3: 적 탱크 시스템

사용자 스토리: 게이머로서, 도전적인 적 탱크와 싸우고 싶습니다. 그래픽 게임이 흥미롭고 재미있을 수 있습니다.

승인 기준 (Acceptance Criteria)

1. WHEN 게임이 시작되면 THEN 적 탱크가 화면 외곽에서 주기적으로 생성되어야 합니다

2. WHEN 적 탱크가 생성되면 THEN 발간색으로 표시되어야 합니다

3. WHEN 적 탱크가 움직이면 THEN 플레이어가 탱크 방향으로 이동해야 합니다

4. WHEN 적 탱크가 플레이어 탱크와 충돌하면 THEN 게임이 종료되어야 합니다

5. WHEN 적 탱크가 총알에 맞으면 THEN 적 탱크는 파괴되어야 합니다

요구사항 4: 점수 시스템

사용자 스토리: 게이머로서, 적을 파괴할 때마다 점수를 얻고 싶습니다. 그래픽 성취감을 느끼고 더 높은 점수를 목표로 할 수 있습니다.

승인 기준 (Acceptance Criteria)

Create Spec: legacy-...

Execute task: 1. 기본 ...

Execute task: 2. 입력 ...

Execute task: 3. 7 +

요구사항 문서 작성했습니다! 전투 탱크 게임의 모든 핵심 기능을 7개의 주요 요구사항으로 정리했습니다:

1. 플레이어 탱크 조작 시스템 - 방향키를 통한 360도 이동 및 회전, 화면 래핑

2. 무기 시스템 - 스페이스바를 통한 발사 및 충돌 처리

3. 적 탱크 시스템 - AI 적의 생성, 이동, 충돌 처리

4. 점수 시스템 - 적 파괴 시 점수 획득 및 표시

5. 게임 상태 관리 - 게임 오버 및 재시작 기능

6. 시각적 표현 - 색상 구분 및 UI 요소

7. 기술적 구현 - 단일 HTML 파일 배포

각 요구사항은 사용자 스토리와 EARS 형식의 승인 기준으로 구체적으로 정의했습니다.

요구사항이 괜찮아 보이나요? 그렇다면 설계 단계로 넘어갈 수 있습니다.

좋습니다! 이제 설계 문서를 작성하겠습니다. 요구사항을 바탕으로 게임의 아키텍처와 구현 방법을 상세히 설계해보겠습니다.

설계 문서를 완성했습니다! 전투 탱크 게임의 전체 아키텍처를 다음과 같이 설계했습니다:

주요 설계 특징:

• 객체 지향 구조: Tank, PlayerTank, EnemyTank, Bullet, Game 클래스로 모듈화

• 게임 루프 아키텍처: 입력 처리 → 상태 업데이트 → 충돌 감지 → 렌더링 순환

• 성능 최적화: 델타 타임 기반 업데이트, 객체 풀링, 효율적인 충돌 감지

• 에러 처리: Canvas 지원 확인, 안전한 이벤트 처리, 폴백 메커니즘

핵심 컴포넌트:

1. Game 클래스 - 전체 게임 상태 및 루프 관리

2. Tank 계층 - 기본 Tank 클래스를 상속받는 PlayerTank, EnemyTank

3. Bullet 클래스 - 총알 물리 및 생명주기 관리

4. InputHandler - 키보드 입력 상태 추적

5. 충돌 감지 시스템 - 원형 충돌 감지 알고리즘

설계에는 벡터 수학, 화면 래핑, 적 AI, 성능 최적화 등 게임 개발의 핵심 요소들이 모두 포함되어 있습니다.

설계가 괜찮아 보이나요? 그렇다면 구현 계획으로 넘어갈 수 있습니다.

훌륭합니다! 이제 구현 계획을 작성하겠습니다. 설계를 바탕으로 단계별 코딩 작업을 정의하여 체계적으로 게임을 구현할 수 있도록 하였습니다.

구현 계획을 완성했습니다! 전투 탱크 게임을 13단계로 나누어 체계적으로 개발할 수 있도록 계획했습니다.

Ask a question or describe a task...

© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

243



Kiro 사례 - 국내 E 사

IaC 전문서 및 표준화/자동화:

- ✓ Terraform 모듈 구조 최적화/리소스 네이밍 컨벤션 준수
- ✓ Agent Steering 기반 표준 자동 적용
- ✓ 변수/출력값 일관성 유지/ 문서 자동 생성
- ✓ HCL 문법, 리소스 의존성, 구문 오류 사전 검증
- ✓ 퍼블릭 접근 차단, 암호화, 태깅 등 보안 규칙 자동 반영

90%

IaC 코드 품질 향상

100%

리소스 태깅
일관성

70%

인프라 작성 시간
향상

60%

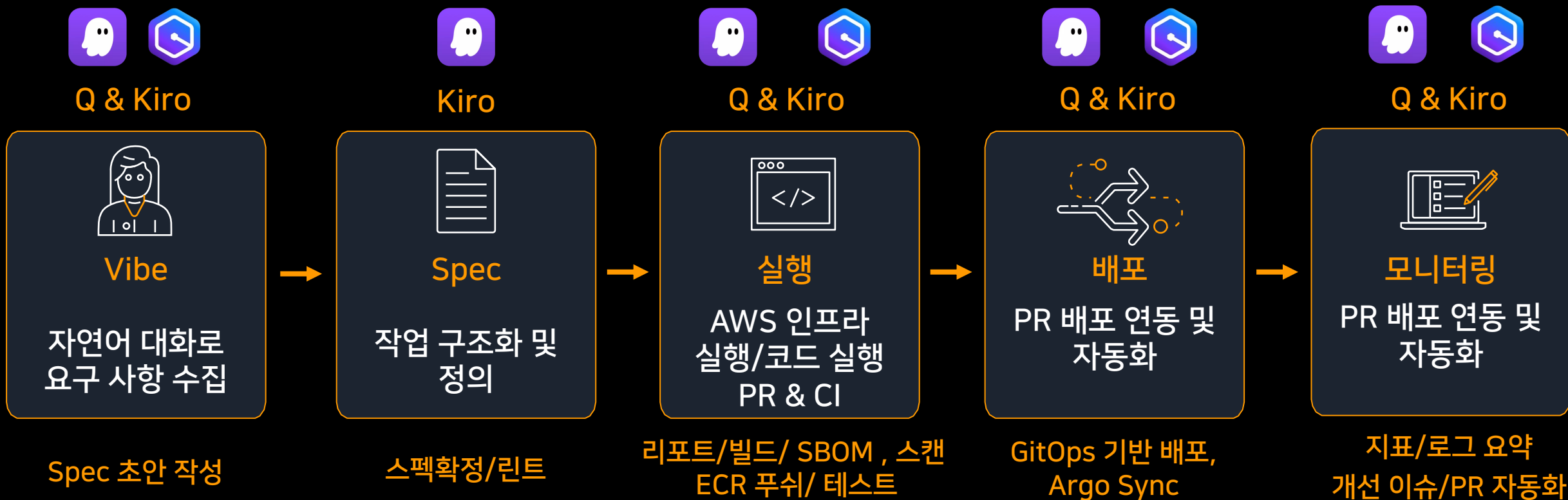
협업 및 문서화
시간 단축

> 개발 및 협업 워크플로우 자동화:

- ✓ 코드 저장시 Agent Hook 자동 실행 (fmt, validate, tfsec, infracost)
- ✓ 환경별 변수 관리 및 모듈 재사용 패턴 적용
- ✓ 멀티 환경 구조에서 재사용 가능한 코드 템플릿 제공



Amazon Q + Kiro 통합 워크플로우



AWS MCP, Github MCP, Github Action MCP

결과 : 자동화된 개발자/운영자 협업 구현

“개발부터 배포, 운영까지 Seamless한 자동화 환경 구축”



Part3. AI 개발도구와 함께 하는 개발자 미래

단순 반복은 AI에게, 창의성은 개발자에게...

AI 동료와의 협업

 새로운 협업 모델 : 개발자 1인 + GenAI 1 = OneTeam

 Kiro + Q Developer : 조합으로 생산성/품질 향상

 개발자는 전략적 의사결정, AI는 코드와 품질 집중

 통합된 맥락(Context)인식, Spec 드리븐으로 일관성 있는
코드품질/ 프로젝트 방향성 유지



지금 바로 시작하세요 !!!



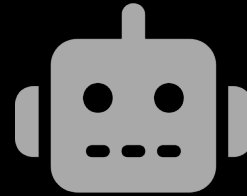
Amazon Q Developer

<https://aws.amazon.com/ko/q/developer>



Kiro

<https://kiro.dev>



반복은 AI 에게



창의는 개발자에게

“Agentic AI 는 더 이상 미래가 아닙니다.”

Thank you!

