GAMES ON AWS 2026

aws FOR GAMES

인게임 AI 어시스턴트에게 게임 플레이 훈수 받기

묘롱의 설계와 선택



박현수

Solutions Architect

AWS

주철민

AI서비스개발팀 리드 개발

KakaoGames

이종우

AI서비스개발팀 개발

KakaoGames

GS Neotek | MEGAZONE CLOUD

오늘 나눌 이야기

00 게임을 이해하는 AI 어시스턴트의 조건
네 가지 조건 · 데이터 설계 · Amazon Bedrock AgentCore 모듈 · 운영과 품질 평가

01 묘롱이 돕는 게임 플레이

02 에이전트 설계의 선택

03 Amazon Bedrock AgentCore 도입과 PoC

04 라이브 게임 도입 실전 노하우

05 마무리

박현수
AWS
Solutions Architect

주철민
카카오게임즈
AI서비스개발팀 리드

이종우
카카오게임즈
AI서비스개발팀

00 게임을 이해하는 AI 어시스턴트의 조건

1987 닌텐도 핫라인 — 내 상황을 아는 상담

게임 플레이 카운슬러의 세 단계



Nintendo Game Play Counselor
1987년, 사람이 전화로 답하던 게임 공략 상담

FLASHBACK · 1987

1 상황 듣기
지금 어디서 막혔나
장비·진행도는?

→

2 공략 찾기
공략집·손글씨 노트에서
해당 구간 탐색

→

3 연결해 안내
이 사람의 상황에
맞는 다음 한 수

40년 뒤의 질문
게임 안에서, AI가 이 세 단계를 대신할 수 있을까?

설명과 추천 — 서로 다른 데이터

게임 지식만으로 충분한 설명, 플레이어의 지금이 필요한 추천

설명

"이 아이템이 뭐야?"

필요한 데이터

- 게임 지식 — 효과 · 사용 조건 · 획득처

모두에게 같은 답 문서에서 근거를 찾으면 끝

→

추천

"내 캐릭터에는 어떤 아이템이 좋아?"

필요한 데이터

- 게임 지식 — 규칙·조건 → RAG
- 내 장비·스탯 — 현재 상태 → API
- 플레이 이력 — 획득·소비 → SQL
- 대화 맥락·목표 — 목적 → Amazon Bedrock AgentCore Memory

사람마다 다른 답 착용 불가 장비를 추천하면 틀린 답

추천은 '게임의 규칙'과 '내 상황'이 만나는 곳에서 나옵니다

게임을 이해하는 AI의 네 가지 조건

오늘의 지도 — 데이터 설계 세 가지, 실행 기반 하나, 측정

1 근거 있는 지식

답의 출처를 찾을 수 있는가?

청크 설계 · 메타데이터
하이브리드 검색

2 유저의 상황

이 유저의 지금을 아는가?

Text-to-SQL · 검증된 API
인증 범위 강제

3 최신성

어제 패치가 반영됐는가?

변경 수집 · 회귀 검증
배포 보류·복귀

4 실행 기반

수많은 유저가 물어도 안정적인가?

Amazon Bedrock AgentCore
에이전트 루프 · 비용
세션 용량 · 콜드 스타트

①②③은 답의 재료를 준비하는 데이터 설계, ④는 그 재료로 답을 만드는 실행 설계

적용 조건을 보존하는 청크 설계

본문 · 조건 · 메타데이터를 한 단위로, 허용 범위 안에서의 검색

같은 문서, 다른 청크 경계

조건이 떨어진 청크

청크 A	청크 B
공격력 +15%	단, 전사 직업 한정

A만 검색되면 → 마법사에게 추천하는 그럴듯한 오답

조건을 보존한 청크

경로	장비 › 무기 › 용의 검
본문	공격력 +15%
조건	전사 · Lv.50 이상
메타	game_id · patch_version · visible_at · access_scope

색인 — 원문 변경을 청크·색인까지 추적



원문	구조 단위 분할	임베딩	검색 색인
Amazon S3	AWS Lambda	Amazon Bedrock	Amazon OpenSearch Service

source_id · chunk_id · updated_at 으로 추가·갱신·삭제 전파

질의 — 허용 범위 안에서만 찾기

허용 범위 필터 버전·공개 시점·권한 ▶ **키워드 + 벡터** 아이템명 · 의미 검색 ▶ **재순위화** 질문 관련성 ▶ **답변 근거** 출처와 함께

공개 시점·권한 필터는 프롬프트가 아니라 검색 단계에서 서버가 적용.

Text-to-SQL — Agentic workflow 적용

의미 정의 → 실행 경계 → 결과 검증의 3단계 Workflow 설계

"지난주 내가 얻은 아이템별 수량은?" 짧은 질문에도 정해야 할 정의가 세 개 숨어 있습니다

1 의미 정의

지난주	어느 시간대·무슨 요일 기준?
내가	로그인으로 인증된 player_id
얻은	드롭? 거래소 구매? 우편 수령?

스키마·조인 키·집계 정의를 함께 제공

2 실행 경계

```
SELECT item_id, SUM(quantity)
FROM player_acquisition_view
WHERE player_id = :authenticated_id
  AND acquired_at >= :start
  AND acquired_at <  :end
GROUP BY item_id  LIMIT :max_rows
```

서버가 바인딩한 신원 · 읽기 전용 역할
허용 뷰 · 실행 시간·스캔량 제한

3 결과 검증

- ✓ 구문·허용 테이블 검사
- ✓ 실행 성공
- ? **정답 집계와 일치?**

정답을 아는 질문 세트로 실제 결과값 비교

모델은 SQL을 제안하고, 신원·권한·한도는 코드와 DB가 강제. 절차적인 Agentic workflow 가 필수

패치 변경과 답변 최신성

문서는 재청킹 · 재색인, 게임 데이터는 CDC — 같은 회귀 검증으로 합류



정합성 문서는 +15%인데 DB는 +10% — 한쪽만 갱신되면 오답

신선도 반영 시각 − 원문 변경 시각, 정확도와 따로 측정

문서와 게임 데이터를 함께 갱신하고, 같은 질문으로 다시 확인합니다

Amazon Bedrock AgentCore — 골라 쓰는 모듈형 구성

어떤 프레임워크 · 어떤 모델이든, 함께 또는 따로 쓰는 모듈형 서비스

실행 · 상태	도구 연결	보안 · 통제	관측 · 개선
Runtime 서버리스 에이전트 실행	**Gateway** API · Lambda → MCP 도구	**Identity** 에이전트 신원 · 인증 연동	**Observability** 트레이스 · 디버깅 · 모니터링
Harness 단일 API로 쓰는 관리형 에이전트 루프	**Code Interpreter** 격리된 코드 실행 샌드박스	**Policy** 도구 호출 전 규칙 강제	**Evaluations** 에이전트 품질 평가
Memory 단기 · 장기 기억	**Browser** 관리형 클라우드 브라우저	**Payments** 에이전트 소액 결제	**Optimization** 개선 추천 · A/B 테스트
	Registry 에이전트 · 도구 카탈로그		

카카오게임즈 묘롱이 사용한 모듈

프레임워크 LangGraph · Strands · CrewAI · LlamaIndex · 직접 구현

모델 Amazon Bedrock 및 외부 모델

전부 쓸 필요는 없음 — 기존 에이전트에 필요한 모듈만

Amazon Bedrock AgentCore Runtime - 프레임워크에 상관없는 서버리스 실행

기존 프레임워크 · 모델 그대로, 세션별 격리 microVM

기존 에이전트 코드

LangGraph · Strands
CrewAI · 직접 구현
+ Runtime SDK 몇 줄

Amazon Bedrock AgentCore Runtime

세션 A	세션 B	세션 C
전용 microVM	전용 microVM	전용 microVM

모델	Amazon Bedrock · 외부 모델
도구	Gateway · MCP 서버
기억	Memory

세션 격리
세션마다 전용 microVM(CPU · 메모리 · 파일시스템)
종료 시 microVM 종료 · 메모리 정리

장시간 실행
microVM 최대 8시간 · Instances 최대 14일
실시간 대화부터 비동기 작업까지

프로토콜 · 스트리밍
HTTP · MCP · A2A
WebSocket 양방향 스트리밍

내장 인증
Identity 연동 — Cognito · Okta · Entra ID
외부 서비스 아웃바운드 인증(OAuth · API 키)

소비 기반 과금
실제 사용한 CPU · 메모리만 과금
LLM 응답 대기 중 CPU 과금 없음

컴퓨트 유형
microVM: 서버리스 · 즉시 확장
Instances: 내 계정 EC2 · GPU · 장기 세션

코드는 그대로 두고 실행 환경 변경 — 묘롱도 기존 LangGraph 코드를 재사용

Amazon Bedrock AgentCore Gateway — 기존 API의 도구화

도구 · 다른 에이전트 · 모델로 향하는 단일 보안 엔드포인트

에이전트
Runtime의 에이전트

MCP 클라이언트
다른 에이전트 · 앱

MCP →

Amazon Bedrock AgentCore Gateway

- 인바운드 인증
- 프로토콜 변환
- 시맨틱 도구 검색
- 자격 증명 주입

→

OpenAPI · Smithy	기존 REST API
AWS Lambda	함수를 도구로
MCP 서버	기존 서버 연결
1-click 통합	Salesforce · Slack · Jira 등

API → MCP 도구
OpenAPI · Smithy · Lambda를
에이전트가 쓰는 MCP 도구로 변환

인바운드 · 아웃바운드 인증
에이전트 신원 확인과 도구 측 OAuth
토큰 갱신 · 자격 증명 보관을 한 곳에서

시맨틱 도구 검색
도구가 수천 개여도 맥락에 맞는 도구만
프롬프트 크기와 지연 감소

하나의 엔드포인트
여러 API · 함수 · 에이전트를 묶어 제공
여러 모델 제공자로의 추론 라우팅

Policy 연동
도구 호출을 실행 전에 가로채
허용 도구 · 동작 · 조건을 규칙으로 강제

서버리스 · 감사
인프라 관리 없이 자동 확장
내장 관측 · 감사 로그

API마다 MCP 서버를 만들지 않고 명세를 등록 — API가 바뀌면 명세 · 타깃 갱신은 필요

Amazon Bedrock AgentCore Memory — 단기 기억과 장기 기억

세션 안의 맥락과 세션을 넘는 인사이트

대화 이벤트
CreateEvent로 저장

→

단기 기억
세션 안의 턴별 원본 이벤트
만료 기간 설정

→

메모리 전략
추출 → 통합 (전략별 성찰)
대화에서 자동 실행

→

장기 기억
세션을 넘는 기억 레코드
네임스페이스로 구분

내장 전략

시맨틱	사용자 선호	요약	에피소드
사실 · 지식	취향 · 플레이 스타일	세션 요약	경험에서 학습

전략 구성 — 내장 · 내장 + 프롬프트 오버라이드 · 자체 관리

한 메모리에 여러 전략을 함께 쓸 수 있고,
여러 에이전트가 같은 메모리를 공유할 수 있습니다

게임에서는 — 설명용 예시

단기	**"그럼 이건 어때?"**	앞에서 말한 아이템을 가리키는지 이해
장기	**"나는 대인전 위주야"**	다음 세션에서도 추천 기준으로 활용
조회	**지금 착용한 장비**	Memory가 아니라 게임 데이터 조회 도구

보관 기간을 먼저 — 지금의 게임 상태는 기억이 아니라 조회 도구에서 가져온다.

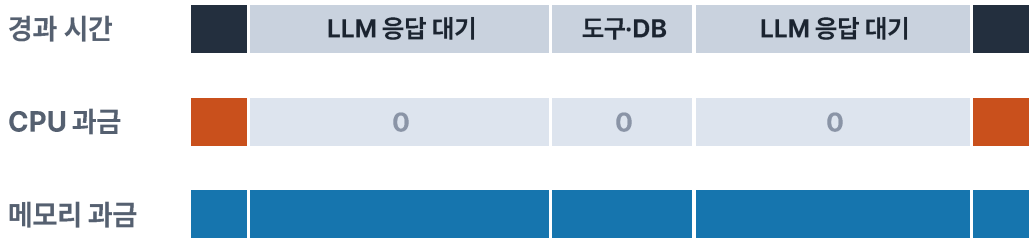
기다리는 워크로드와 서버리스 비용

Amazon Bedrock AgentCore 도입 전후 — 구현 위치와 비용 구조의 변화

무엇이 어디로 옮겨갔나

	도입 전 구상	도입 후
대화 실행	상시 Amazon EC2 API 서버 앱 수준 세션 관리	Amazon Bedrock AgentCore Runtime 세션별 microVM 격리
대화 상태	DB checkpointer PostgreSQL	Amazon Bedrock AgentCore Memory 대화 이벤트 저장·복원
도구 연결	API별 클라이언트 개별 구현	Amazon Bedrock AgentCore Gateway 기존 API를 MCP 도구로
과금	상시 인스턴스 점유 유휴 시간도 비용	**사용한 CPU·메모리 초 단위 과금**

한 번의 질문 — 시간은 어디에 쓰이나



- **I/O 대기 중 CPU는 0으로** LLM 응답 · 도구 호출 · DB 조회를 기다리는 동안
- **초 단위 과금, 쓰지 않으면 0까지 축소** 트래픽이 없는 새벽, 패치 직후 급증에도 같은 구조
- **메모리는 세션 동안 과금** V2는 유휴 메모리 120초 뒤 회수

피크에 맞춰 서버를 상시로 띄우는 대신, 실제로 사용한 CPU·메모리만큼 비용 지불

동시 실행 세션 규모 추정

Little's Law — 평균 개수 = 유입률 × 평균 체류 시간, '무엇을 셀지'부터

$$L = \lambda \times W$$

λ 초당 생성·재생성되는 수

W 생성부터 종료까지 평균 유지 시간

실행 세션

1,200 개 평균 유지

2개/초 × 600초

다음 질문 대기 · 유휴 종료까지 포함

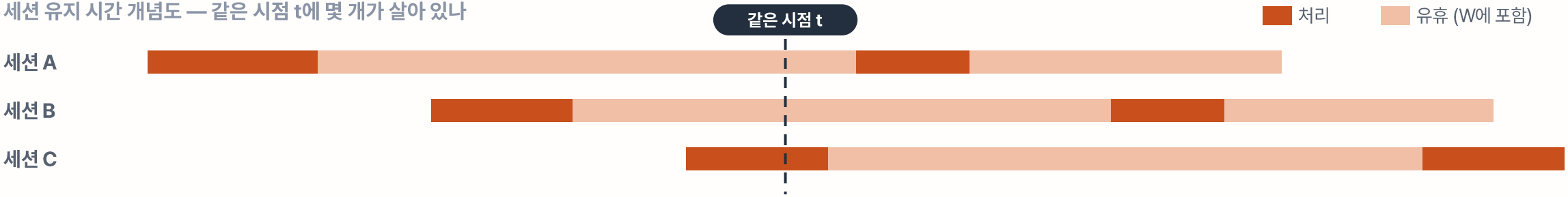
처리 중 요청

80 개 평균 처리

10개/초 × 8초

모델·도구 호출 중인 요청만

세션 유지 시간 개념도 — 같은 시점 t에 몇 개가 살아 있나



게임 동접이 아니라, 세션의 생성·재생성률과 유휴를 포함한 유지 시간을 입력

AgentCore Runtime V2 — 스냅샷 기반 콜드 스타트 개선

이미지 크기와 무관한 새 세션 준비 시간

콜드 스타트 P75 (AWS echo 테스트)



동작 방식

사전 초기화 ▶ 스냅샷 저장 ▶ 새 세션에서 복원 ▶ 요청 처리

초기화 코드 점검표

platformVersion: V2

미리 준비	라이브러리 로딩 · 정적 설정 · 클라이언트 준비
요청 시점	유저 데이터 · 만료되는 자격 증명 고유 ID · 시간 · 난수
복원 후	소켓·DB 연결은 다시 맺기

팁 채팅창을 여는 순간 세션을 시작하면, 입력하는 동안 준비됩니다

무엇을 미리 준비하고, 무엇을 요청 시점에 다룰지 — 초기화 코드를 나눠 점검

이제, 묘롱은 이 조건을 어떻게 풀었을까요?

카카오게임즈 묘롱 — 게임을 이해하는 AI의 조건을 실제 MMORPG에 적용한 설계와 경험

오늘의 네 조건 → 이어지는 발표에서 볼 부분

1 근거 있는 지식	02 질문에 맞는 데이터 소스 선택 · 오타·줄임말부터 게임 ID까지
2 내 상황	02 판단은 AI에게, 필수 규칙은 코드로 · 04 API 도구 연동 (Gateway)
3 최신성	02 공지 · 이벤트 · 공략 소스 — "진행 중인 이벤트는?"
4 실행 기반	03 AgentCore 도입과 PoC · 04 세션 기준 · DB·Redis 연결 · 재시도와 종료

"내 캐릭터에는
어떤 아이템이 좋아?"



묘롱

데이터 연결 · AI와 코드의 책임 분담 · AgentCore 도입과 라이브 운영 노하우 — 카카오게임즈 AI서비스개발팀

01 묘롱이 돕는 게임 플레이

카카오게임즈 AI서비스개발팀

01 PRODUCT

AI 기술의 제품화

기술을 이해하고 실제 문제를 해결하는
제품과 서비스를 만듭니다

02 ENABLEMENT

사내 AI 기술 확산

여러 업무와 서비스에 AI가 쓰이도록
사내 활용의 기회를 넓힙니다




MMORPG의 새로운 차원
도깨비의세계

도깨비의세계와 묘롱

kakaogames

도깨비의세계

게임 안과 밖 어디서든 만날 수 있는 AI 어시스턴트 묘롱



묘롱

반가워, 도사. 이 몸은 묘선생!
오늘은 어떤 것을 알려줄까?

묘롱에게 물어보기 전송

https://myorong.kakaogames.com

MMORPG에서 유저가 막히는 순간

신규 유저의 시스템 이해와 기존 유저의 성장 판단

신규 유저

"도술이 뭐야?"

낯선 용어와 규칙

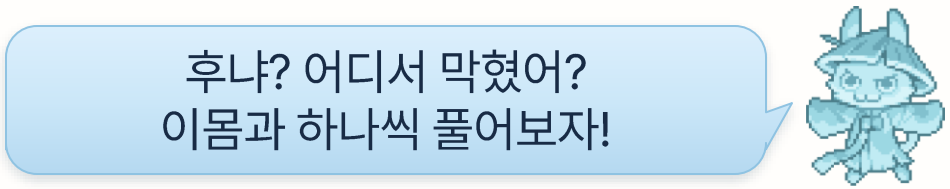
묘롱의 설명 → 시스템 이해 다음 행동 선택

기존 유저

"어제 뭘 얻었지?"

복잡해진 성장과 이력

묘롱의 조회·해석 → 내 상태 파악 성장 방향 판단



신수가 뭐야?

게임 정보 안내



신수는 도사의 영적 동반자야.
내 정보에서 확인할 수 있어, 도사!

공명의 매개체 파편, 어디서 얻어?

게임 데이터 검색



후냐아! 대나무 강의 현랑귀가 떨군대~
이몸이 가호 얹어줄게, 도사!

오늘 사냥으로 골드 얼마나 벌었어?

플레이 이력 조회



냐하하! 오늘 사냥으로
총 52만 엽전 벌었어, 도사!

남들은 몇 강 끼고 있어?

성장 수준 비교



무기 서버 평균은 5강, 도사는 7강이래.
평균 위라니, 이몸이 다 으쓱해진다~

최근 4주 영석, 어디에 제일 많이 썼어?

소비 내역 분석



묘로롱~ 영석 소비의 71.5%가 강화야!
도사, 강화에 가장 많이 썼구나~

기능 문서 기반 답변 예시 · 수치·획득처는 설명용이며 실제 응답과 다를 수 있습니다

내 상태에 맞는 다음 할 일은?

실제 대화로 보는 묘룡의 성장 조언

나 이제 뭐 해야 해?

상태 점검	도사의 상황	묘룡의 판단	묘룡의 답변
만렙 달성 여부 콘텐츠 진행도 현재 전투력	**만렙을 달성 못함**	**퀘스트 진행 필요**	**퀘스트 진행 유도**



도사, 아직 만렙이 안 됐네!
우선 퀘스트를 진행해서 만렙을 목표로 해보는 게 어때?

언론이 소개한 묘롱의 역할

언론이 눈여겨본 두 가지 플레이 지원 방식

게임 안에서 필요한 정보 확인

시사위크 · 조윤찬 기자 · 2026.09.17

‘도깨비의세계’ 공략은 AI ‘묘롱’에 물어라

게임 내부 데이터로
외부 공략 검색 부담 감소

플레이 데이터 기반 성장 조언

블로터 · 최이담 기자 · 2026.08.20

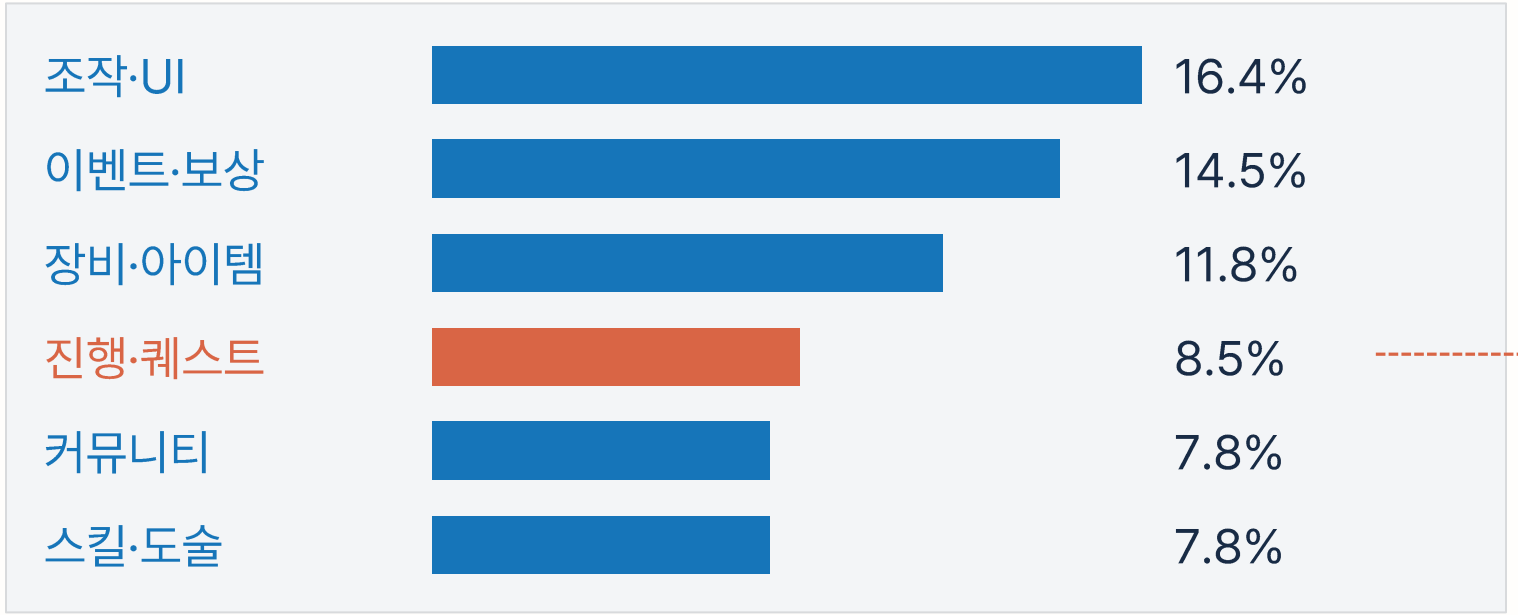
AI '묘롱' 성장 코칭...PK 선택제 도깨비의세계 10월 출격[현장+]

장비·스탯·플레이 기록을 분석해
성장 방향 제안

유저들은 어떤 순간에 묘롱을 찾았을까?

조작 방법부터 다음 할 일과 성장 방향까지

질문 유형별 비중 · 상위 유형



나머지 유형 합계 32.9%

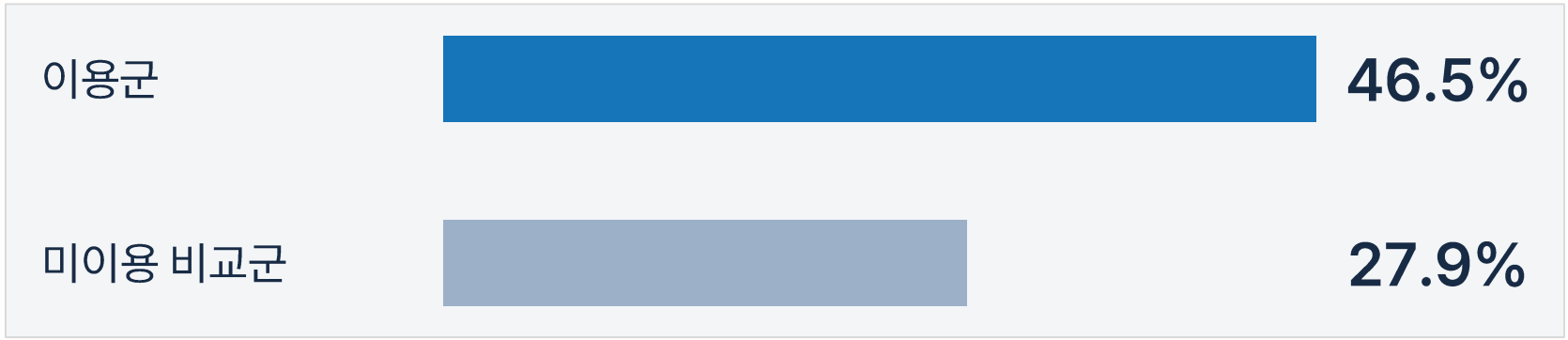
해당 유형의 대표 질문

"지금 뭘 해야 할까?"

성장 가이드 이용자는 안내한 행동을 더 많이 실행

“지금 뭘 해야 할까?” 성장 점검 후 1시간의 행동 변화

안내 항목을 1시간 안에 실행한 비율



(약 8%p)

질문 전 차이를 보정하면

약 +11%p

(질문 전후 실행률의 변화를
이용군과 비교군 사이에서 비교)

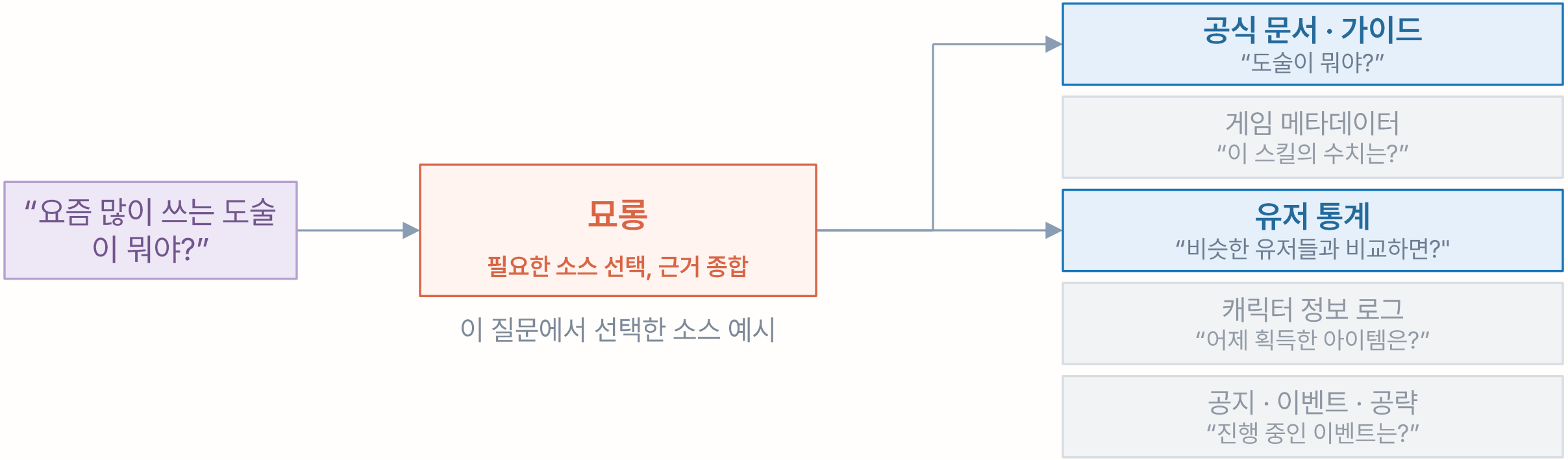
초기 관찰 분석 · 무작위 실험은 아니며, 미이용군은 안내 종류·활동 시간 구성을 보정

출처: 송대섭(2026.10), 「<도깨비의세계> 초보 성장 점검 '지금 뭘 해야할까?' 이용 후 행동 변화」, TF 내부 보고서, 발췌

02 에이전트 설계의 선택

묘롱은 질문에 맞는 데이터 소스를 찾습니다

질문에 필요한 데이터만 선택해 근거 종합



질문에 필요한 소스만 선택해 조회하고, 여러 근거를 종합합니다

오타·줄임말부터 게임 ID까지

유저의 표현을 게임 데이터와 연결하고, 이해할 수 있는 답변으로 변환

오니불 어디서 얻어?

이름 확인

게임 속 이름의 후보 확인

오니 뿔로 교정

정보 조회

필요한 데이터 조회

획득처의 몬스터 ID

정책 적용

게임 ID를 이름으로 변환

몬스터 ID → 동구밖 자색 굴레귀

답변 구성

근거를 종합하여 묘롱의 말투로 답변

유저가 이해할 수 있는 설명



묘로롱~! 도사, 혹시 오니 뿔을 말한거라면, 동구밖 자색 굴레귀가 떨궈.

설명용 예시 · 질문에 따라 필요한 단계만 수행하며, 모든 질문이 같은 순서로 처리되지는 않습니다

그럼 ReAct 에이전트를 만들면 되지 않을까?

자율적인 탐색과 함께 필요한 비용·속도·실행 범위 관리

실제 유저에게 제공하기 위한 서비스 요구

1. 동작의 예측 가능성

도구와 실행 범위·종료 조건 지정
불필요한 탐색 제한

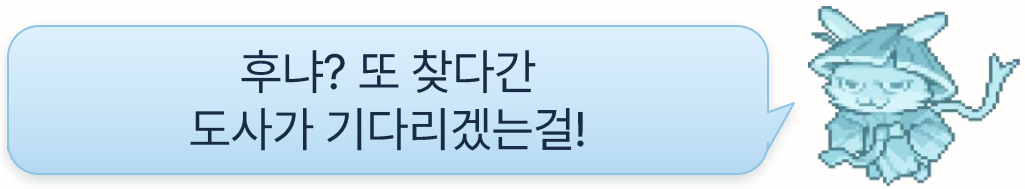
2. 요청당 비용 관리

반복 판단과 도구 호출마다
누적되는 비용 관리

3. 플레이 중 응답속도

추가 탐색에 따른 대기와
응답 지연 관리

당시의 선택: 비용과 응답 시간을 관리할 수 있는 DAG (Directional Acyclic Graph)



판단은 AI에게, 필수 규칙과 순서는 코드로

AI의 유연한 판단과 코드가 보장하는 필수 흐름

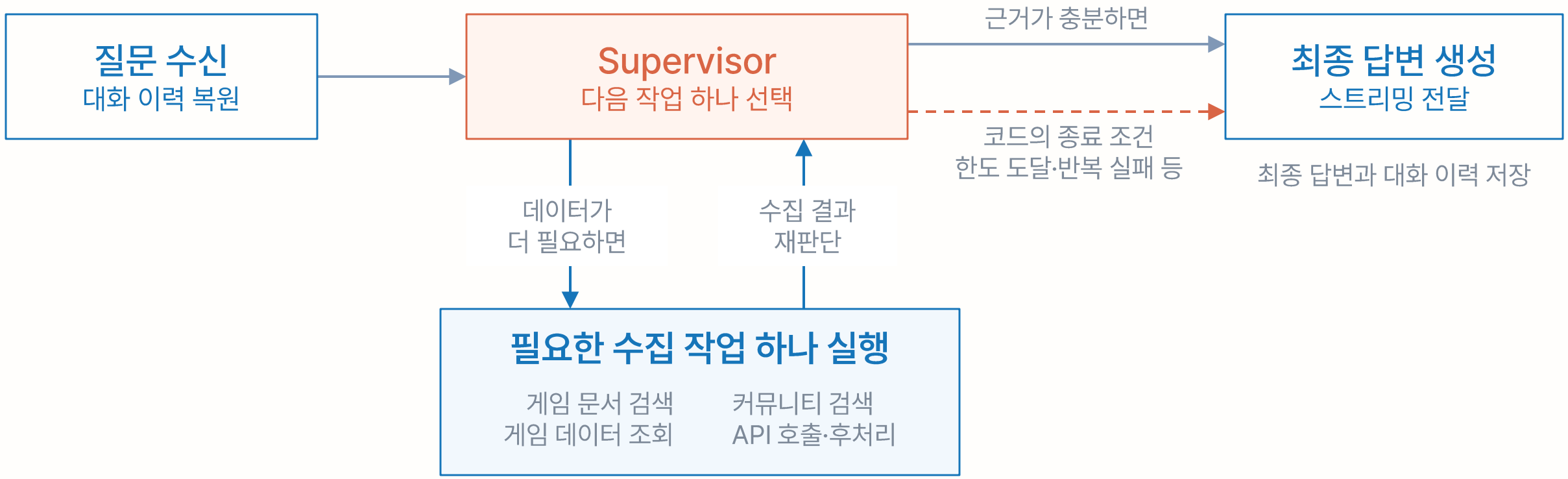
처리 경로를 미리 정의 → AI 판단을 단계별로 분리 → **현재: AI 판단 + 코드 통제**

- 동작 예측, 비용과 응답 시간 고려
- 판단과 데이터 수집을 분리
- AI가 도구와 호출 계획을 판단
- 코드가 필수 규칙·순서를 보장

설계의 균형: AI의 유연한 판단 + 코드가 보장하는 필수 흐름

Supervisor 중심의 에이전트 실행 흐름

AI의 다음 작업 선택과 코드의 반복·종료 제어



03 Amazon Bedrock AgentCore 도입과 PoC

묘롱의 대화 워크로드는 어떤 모습인가?

조회와 추론을 반복하고, 답변을 스트리밍하는 대화 워크로드

실행 길이가 가변적

- 질문마다 다른 조회·추론 횟수
- 가변적인 처리 시간

외부 응답을 기다림

- LLM 생성과 도구 호출 사이
- 외부 응답 대기

스트리밍 연결 유지

- 답변 생성 중 연결 유지
- 생성한 내용을 순차 전달

요청량의 변동

- 출시 초기 수요 예측의 어려움
- 긴 유휴와 순간 피크

워크로드를 따라 바뀐 인프라 선택

요청량 변동과 대기 시간, 과금 방식에 따른 선택

Amazon EC2

상시 서버로 시작

- 긴 유휴와 순간 피크
- 상시 자원의 낭비

AWS Lambda

요청량에 맞춘 서버리스 검토

- LLM·도구 응답 대기
- 실행 시간 기반 과금 부담

Amazon Bedrock AgentCore

대화 요청에 맞는 서버리스 환경 선택

- AWS Summit에서 대안 발견
- 기존 코드로 즉시 PoC

Amazon Bedrock AgentCore 도입 전후 역할별 변화

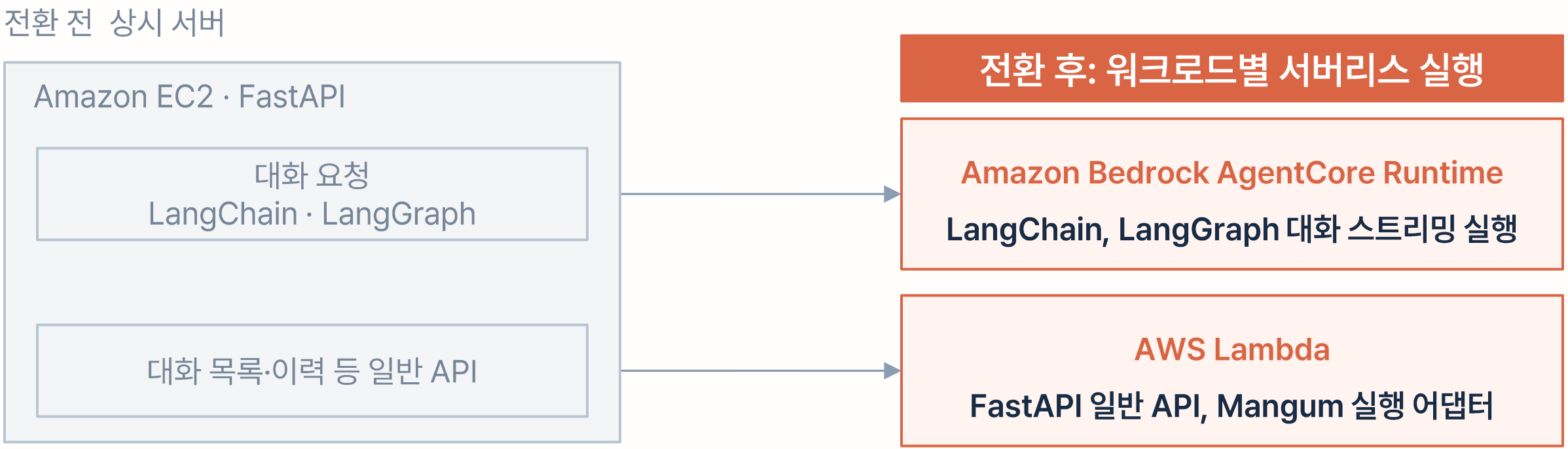
실행·이력·도구 연결은 관리형으로, 기존 로직은 재사용

역할	기존 구성	도입 후: 실행·이력·도구 연결을 관리형으로
대화 실행	Amazon EC2 · FastAPI에서 그래프 실행	**Amazon Bedrock AgentCore Runtime** 서버리스 실행
대화 이력	PostgreSQL checkpointer	**Amazon Bedrock AgentCore Memory** 이력 복원·적재
도구 연결	애플리케이션의 개별 API 연동	**Amazon Bedrock AgentCore Gateway** MCP 도구 연결
일반 API	FastAPI의 목록·이력 조회 API	**AWS Lambda** FastAPI 재사용

코드 재사용: LangChain&LangGraph의 에이전트 로직과 FastAPI 로 만들어진 서비스 API

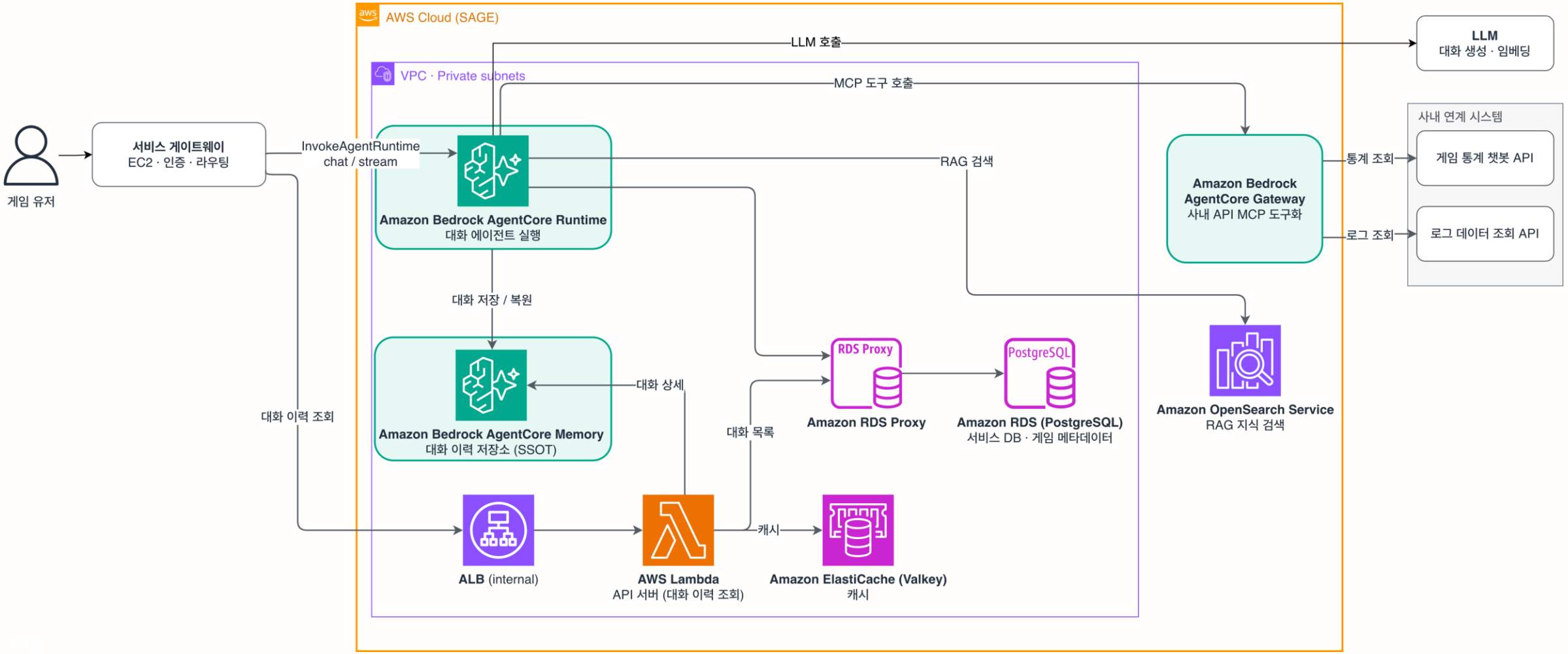
전환과 PoC 대화 요청과 일반 API의 분리

LangChain·LangGraph와 FastAPI를 워크로드에 맞게 분리



공통 저장소: Amazon Bedrock AgentCore Memory · 서비스 DB

묘롱의 AWS 서비스 구성



게임 속 질문에서, 서비스의 실행까지

AI: 판단
질문 해석과 데이터 소스 선택
다음 작업 결정

코드: 보장
필수 규칙과 실행 범위
신원 확인과 종료 조건



Amazon Bedrock AgentCore
대화 실행을 서버리스로

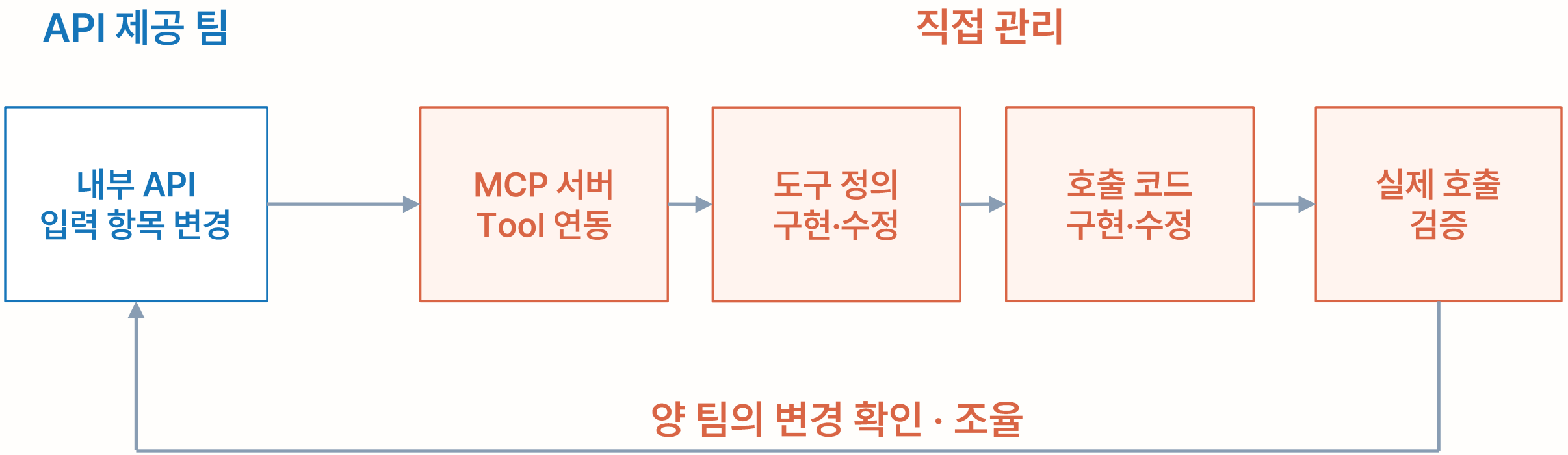
도사, 다음 이야기도
이 몸과 함께 들어보자!



04 라이브 게임 도입 실전 노하우

API 도구 연동 부담 줄이기

기존 API 연동의 구현 부담



API 도구 연동 부담 줄이기

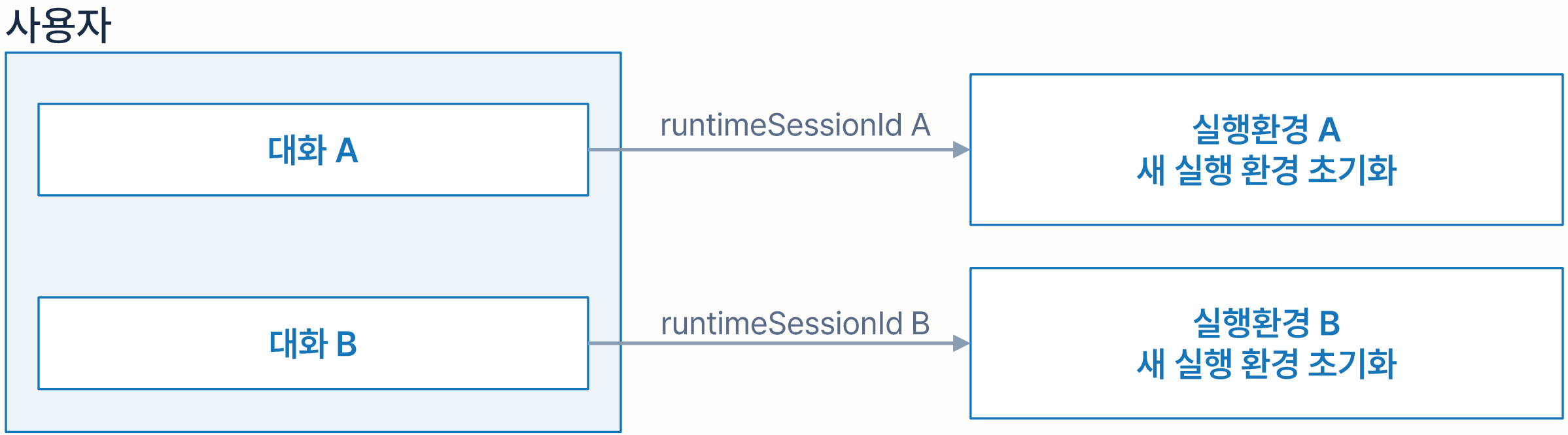
OpenAPI 기반 MCP 도구 연동



OpenAPI 명세 기반 연동으로, 도구 연결 코드의 구현·관리 부담 감소

Amazon Bedrock AgentCore Runtime 세션의 기준

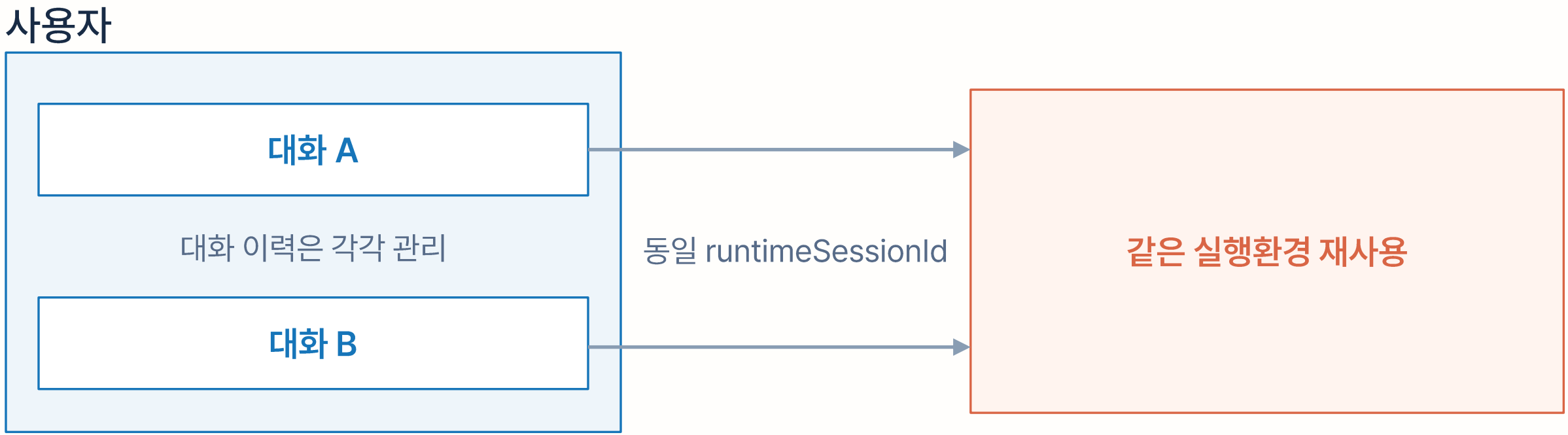
대화 단위로 나눈 실행 세션



같은 사용자도 새 대화마다 실행 환경을 초기화하며 대기 발생

Amazon Bedrock AgentCore Runtime 세션의 기준

유저 단위로 재사용하는 실행 환경



대화 이력은 분리하고 실행 환경은 재사용해, 반복 초기화 부담 감소

Amazon Bedrock AgentCore Runtime 확장과 DB 연결 관리

실행 환경과 함께 늘어나는 DB 연결

요청 집중

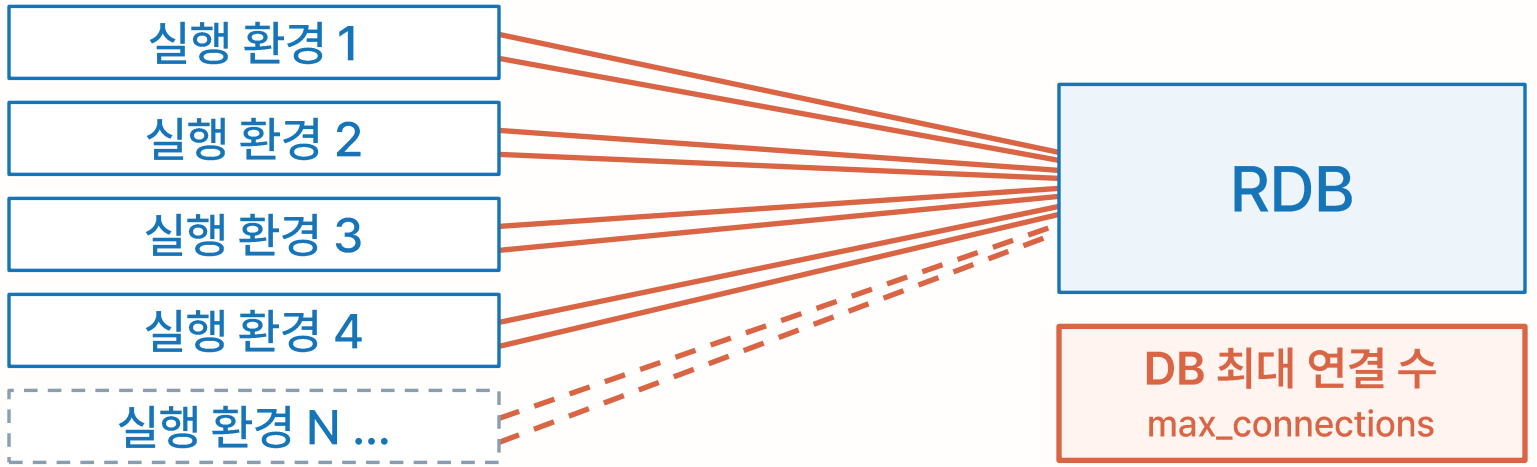
신규 오픈 · 점검 직후



요청 패턴 예시

실행 환경 확장

요청에 따라 탄력적으로 증가



DB 연결 급증

수용 가능한 연결 수에는 한도

실행 환경 수 × **환경당 평균 DB 연결 수** = **전체 DB 연결 수**

Amazon Bedrock AgentCore Runtime 확장과 DB 연결 관리

Amazon RDS Proxy를 통한 DB 연결 재사용

실행 환경의 연결

확장된 실행 환경의 요청

Amazon RDS Proxy

DB 연결 풀 관리

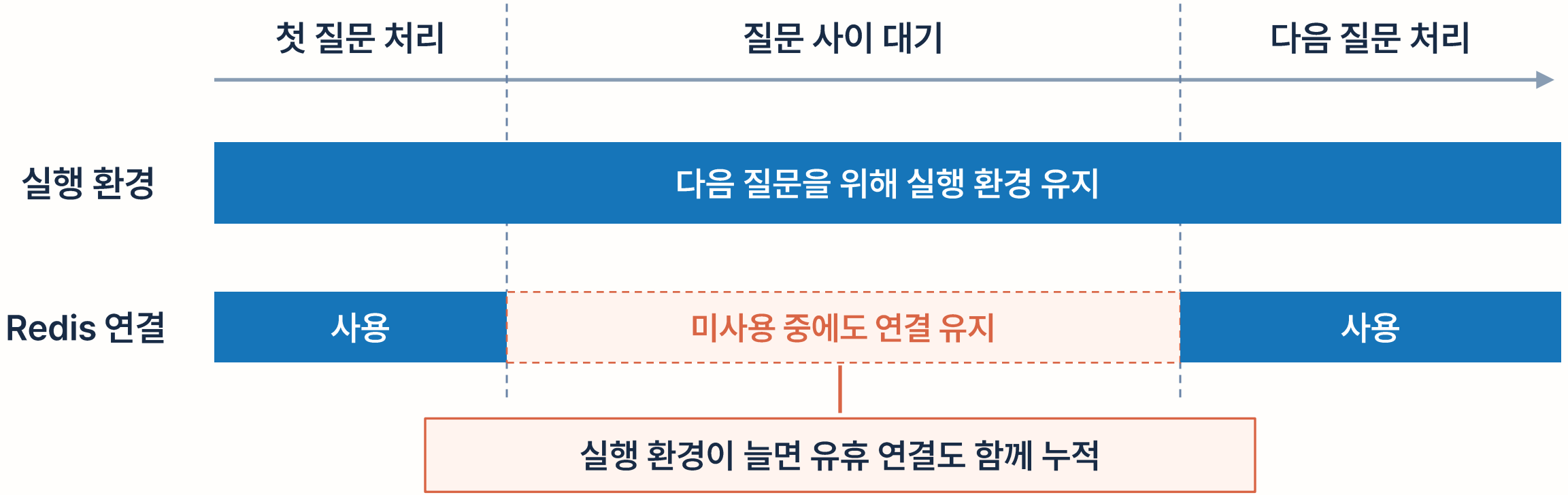
RDB

DB 연결 부담 완화



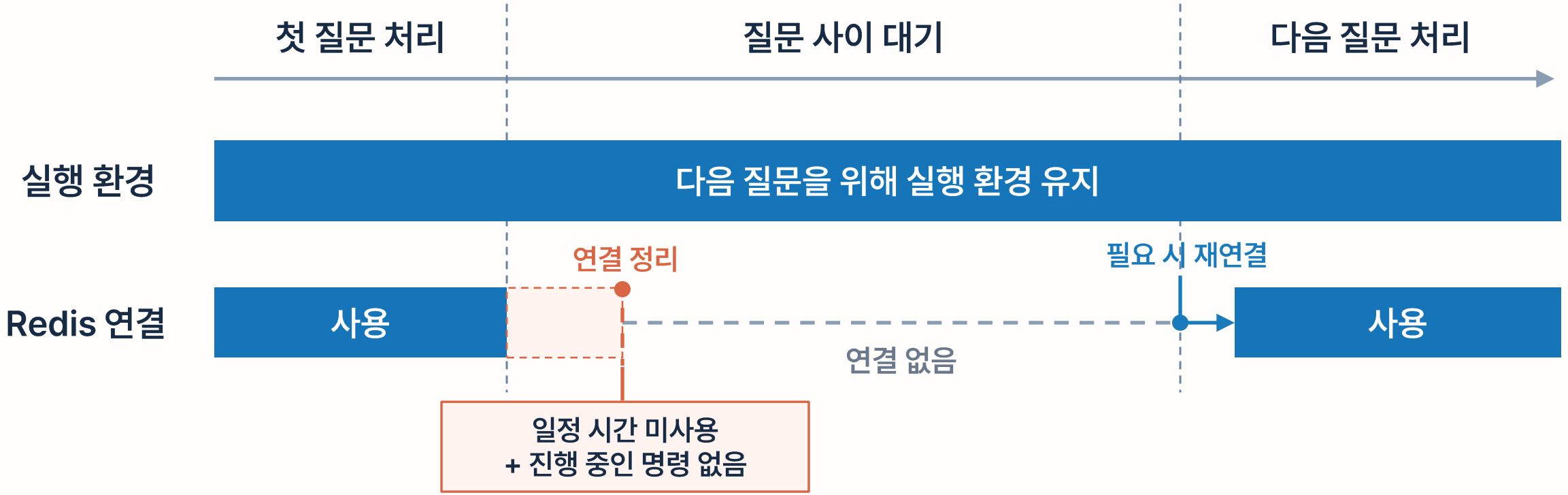
Amazon Bedrock AgentCore Runtime과 Redis 연결의 수명 분리

실행 환경에 남아 있는 유휴 연결



Amazon Bedrock AgentCore Runtime과 Redis 연결의 수명 분리

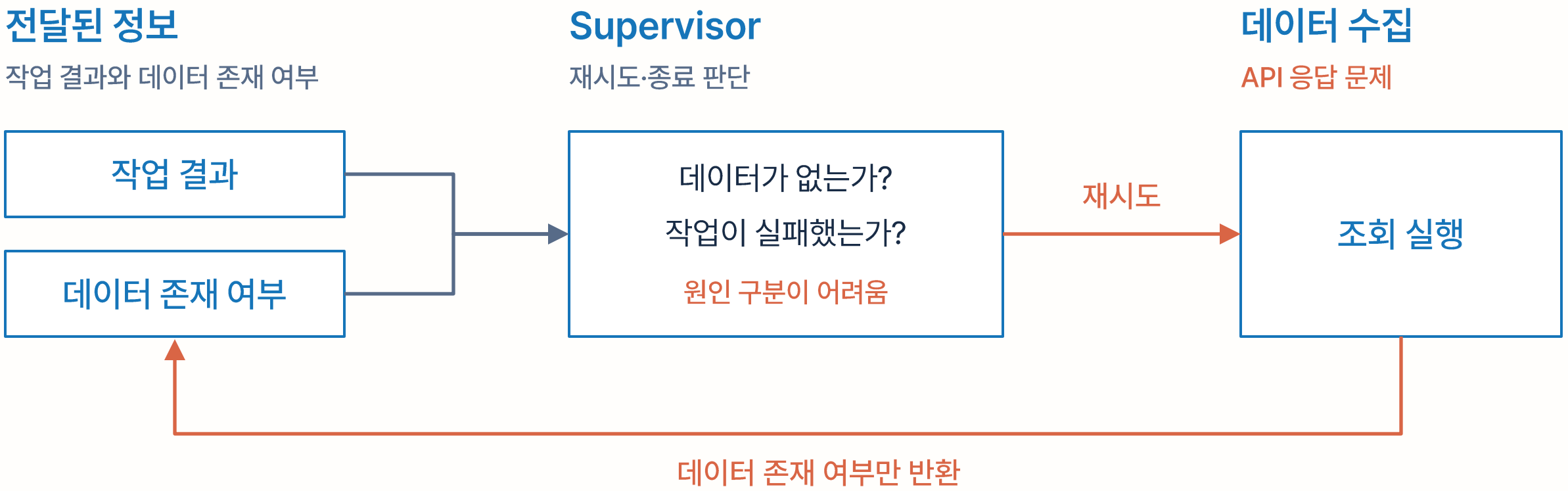
미사용 연결은 정리하고, 필요할 때 재연결



실행 환경과 Redis 연결의 수명을 분리

에이전트의 재시도와 종료 기준

판단 정보 부족으로 반복된 실행



재시도 근거를 구분하지 못해 수 시간 반복

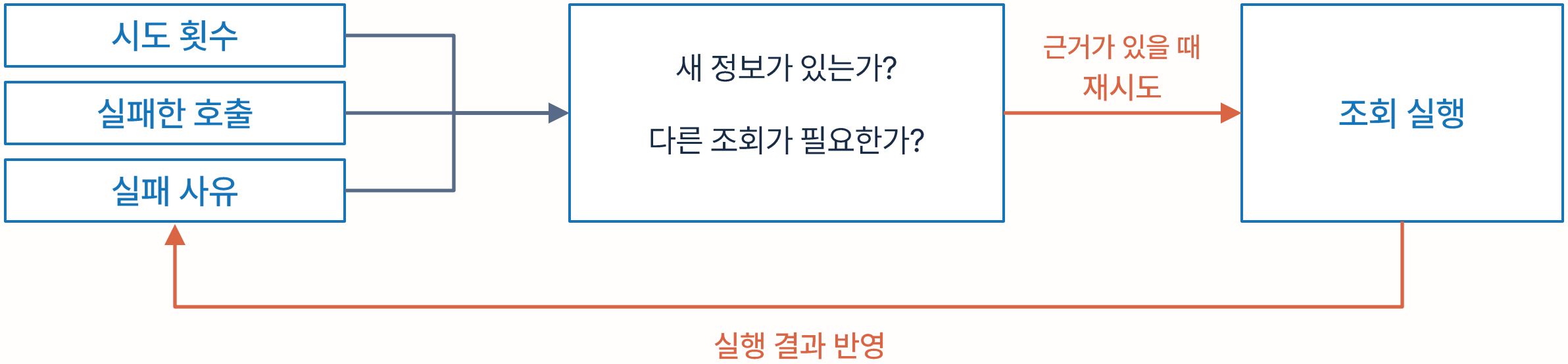
에이전트의 재시도와 종료 기준

재시도 판단 근거 보강

작업 상태
판단에 필요한 정보 전달

Supervisor
추가 시도의 근거 확인

데이터 수집
필요한 조회 실행



추가 시도의 근거가 있을 때 필요한 재시도를 허용

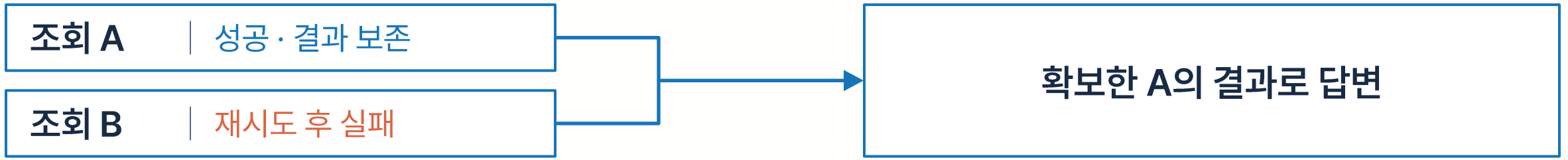
에이전트의 재시도와 종료 기준

실행 상한과 최종 응답 처리

실행 step 상한



부분 성공 시 응답



최종 응답

확보한 A의 결과로 답변

확보한 결과로 적시에 응답해, 답변 품질과 사용자 경험의 균형 확보

AWS Countdown Premium

새로운 기술과 서버리스 구현, 경험의 공백을 함께 보완

서비스 출시와 대규모 이벤트의 안정적 운영을 돕는 AWS 전담 기술 지원 서비스

Amazon Bedrock AgentCore 도입

**출시 후 사례가 쌓여가는 단계,
우리 서비스에 맞는 검증 필요**

도입 당시 참고할 수 있는 활용 사례가 제한적

서비스 전체의 서버리스 구현

**팀에서도 경험이 부족했던
서버리스 설계와 운영**

테스트부터 운영까지 검증할 기준 필요

AWS Countdown Premium

AWS 서비스에 대한 깊은 이해와 긴밀한 소통

기술적 지름길

테스트와 설정 튜닝의 시행착오 감소

AWS 서비스에 대한 깊은 이해로
검증 방향과 설정값을 함께 논의

한 팀 같은 협업

개발 전 과정에서 긴밀한 소통

테스트부터 배포, 운영까지 맥락을 공유하며
함께 문제를 해결

05 마무리

Amazon Bedrock AgentCore 위에서 게임 서비스에 집중하기

에이전트 설계부터 도입과 운영까지, 구현 과정에서 정한 기준

에이전트 설계

판단은 AI에게,
실행 기준은 코드로

도구와 다음 작업은 AI가 판단

필수 규칙·실행 범위·종료 조건은 코드로 보장

Amazon Bedrock AgentCore 도입

기존 구현을 재사용하고,
실행·연동 부담 축소

LangGraph를 재사용해 실행 환경 전환

Amazon Bedrock AgentCore Gateway로 기존 API의 MCP 도구 연동 구현 부담 감소

운영 설계

세션·연결·재시도에
명확한 기준

대화와 실행 세션을 구분하고 실행 환경과 연결의 수명을 분리

실패 정보를 바탕으로 재시도하고 상한 전에 응답으로 전환

Acknowledgements

AI서비스개발팀



주철민



이종우



김혜정

aws FOR GAMES

Thank you



GS Neotek | MEGAZONE CLOUD