aws FOR GAMES

GAMES ON AWS 2026

GS Neotek | MEGAZONE CLOUD

GAMES ON AWS 2026

aws FOR GAMES

퍼블리싱 가속화 에이전트 Part 2

코드 리뷰부터 문서 자동 반영까지

송정우

솔루션즈 아키텍트

AWS

이아영

퍼블리싱 기술실

LINE Games

GS Neotek | MEGAZONE CLOUD

목차

01 AI Transformation(AX) on AWS

02 LINE Games의 퍼블리싱 가속화 여정

03 퍼블리싱 가속화 프레임워크

04 LINE Games의 AX on AWS

05 다음 단계와 남은 교훈

01

AI Transformation(AX) on AWS

AI Transformation(AX)는 왜 어려운가?



천차만별인 AI
사용 경험

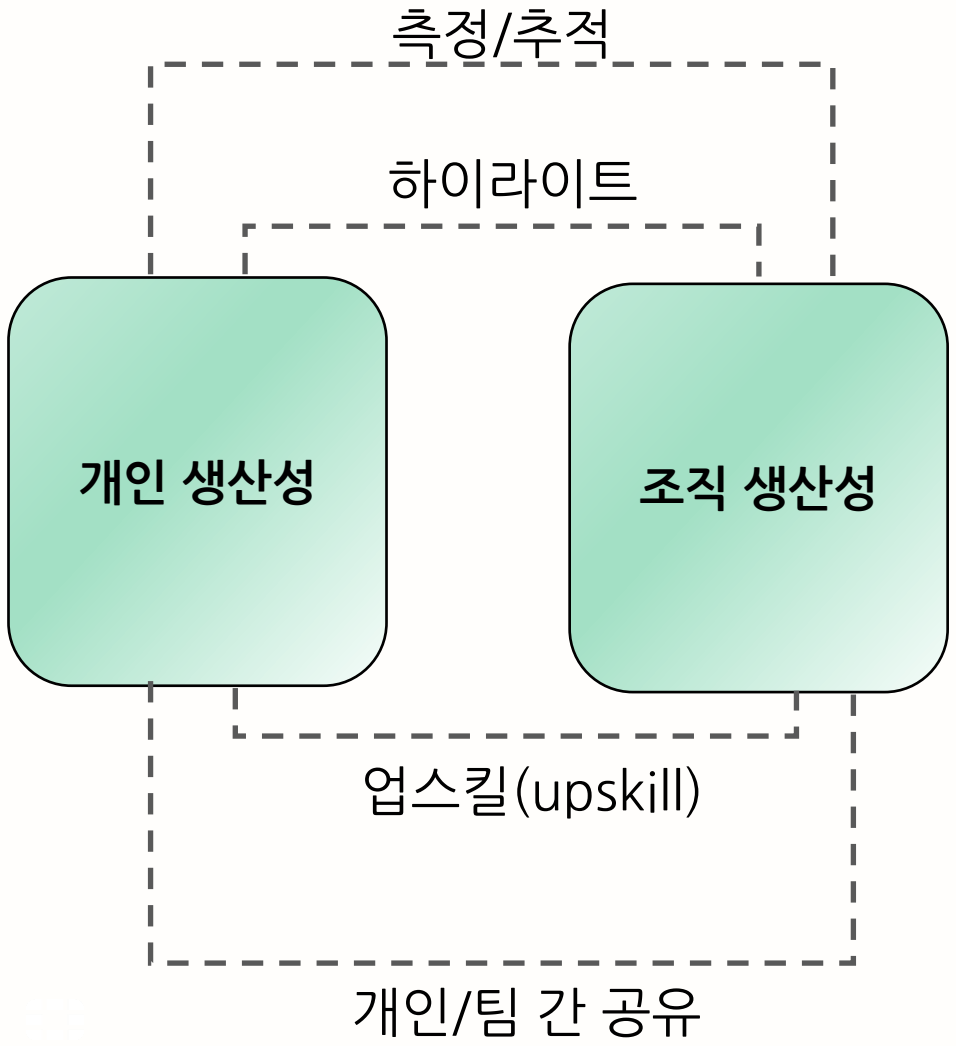


개인 – 조직의
AI 이해도 간극



비용 - 생산성

AI Transformation(AX) Journey



개인의 생산성이 조직의 생산성이 되고,
조직의 생산성이 개인의 생산성이 되는 과정

AI Transformation(AX) on AWS

AGENT

Amazon Bedrock AgentCore



에이전트 개발

OPS

AWS DevOps & AWS Continuum





운영과 보안

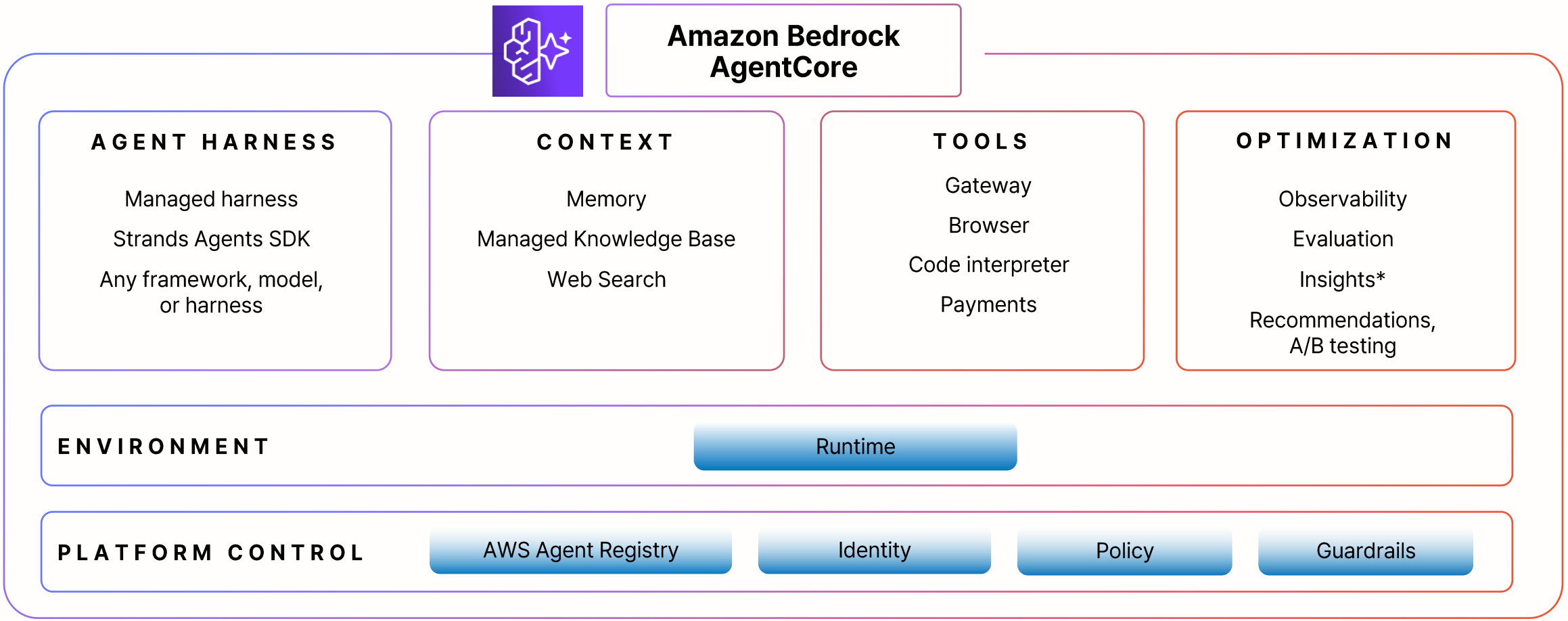
BUISNESS

Amazon Quick



비개발자를 위한 AI 도구

Amazon Bedrock AgentCore



디자인에 따라 조합하고, 사용한 것에만 지불하세요

AWS DevOps Agent & AWS Continuum

For operation



AWS DevOps Agent

자동화된 장애 조사

자동으로 운영 이슈를 조사합니다.

사전 장애 방지

이전의 패턴 분석을 통해 사전에 장애를 방지합니다.

프로덕션 레벨 지능

여러분의 환경, 코드, 운영 인사이트로부터 배웁니다.



AWS Continuum

Security Agent

- 디자인 리뷰
- 코드 리뷰
- 위협 모델링
- 침투 테스트

Amazon Quick

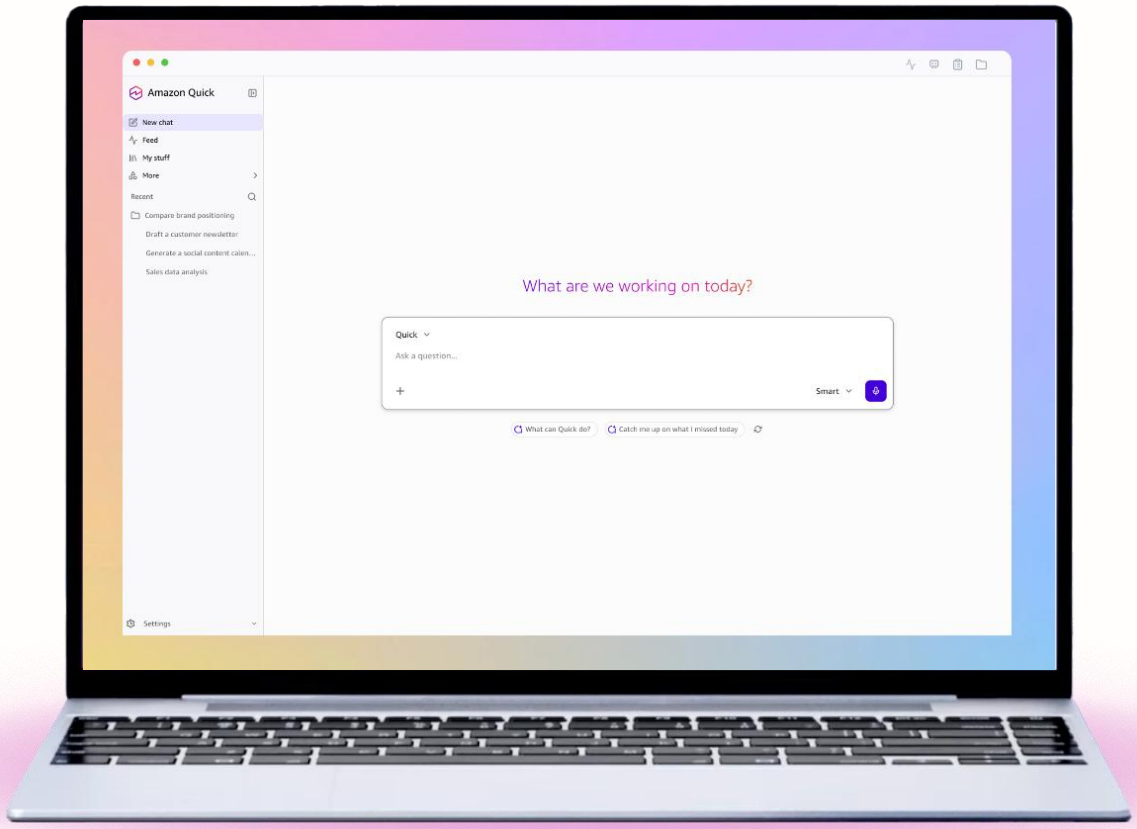
For business

질문에 답하고, 답변을 액션으로 바꾸고, 액션을 결과로 바꾸는 AI 어시스턴트

초개인화
이미 쓰고 있던 툴, 데이터, MCP를 손쉽게 연결하세요. 웹이 아닌 데스크탑에서도 사용할 수 있습니다.

일하는 방식을 바꿉니다
답변만 하는 것이 아니라, 대시보드를 만들고 주기적인 액션 만들어 보고서를 받아볼 수 있습니다.

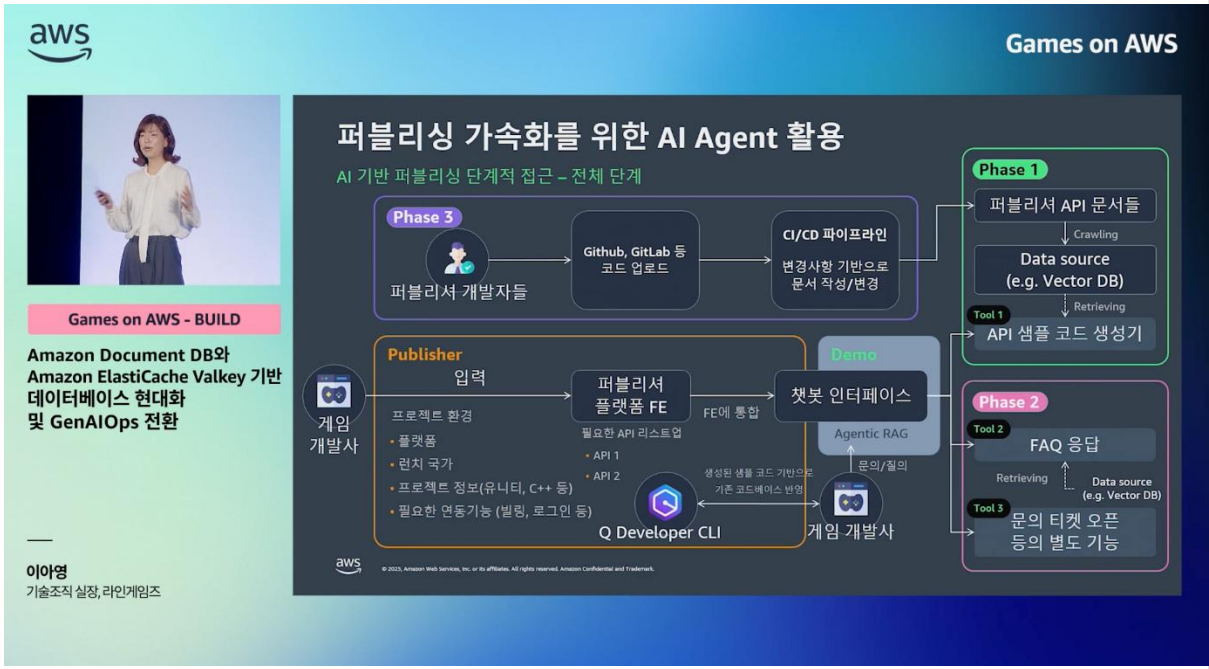
조직 레벨의 AI
조직 레벨의 Connector, AI Agent를 운영할 수 있습니다.



퍼블리싱 가속화 에이전트 Part 1

2025 Games on AWS

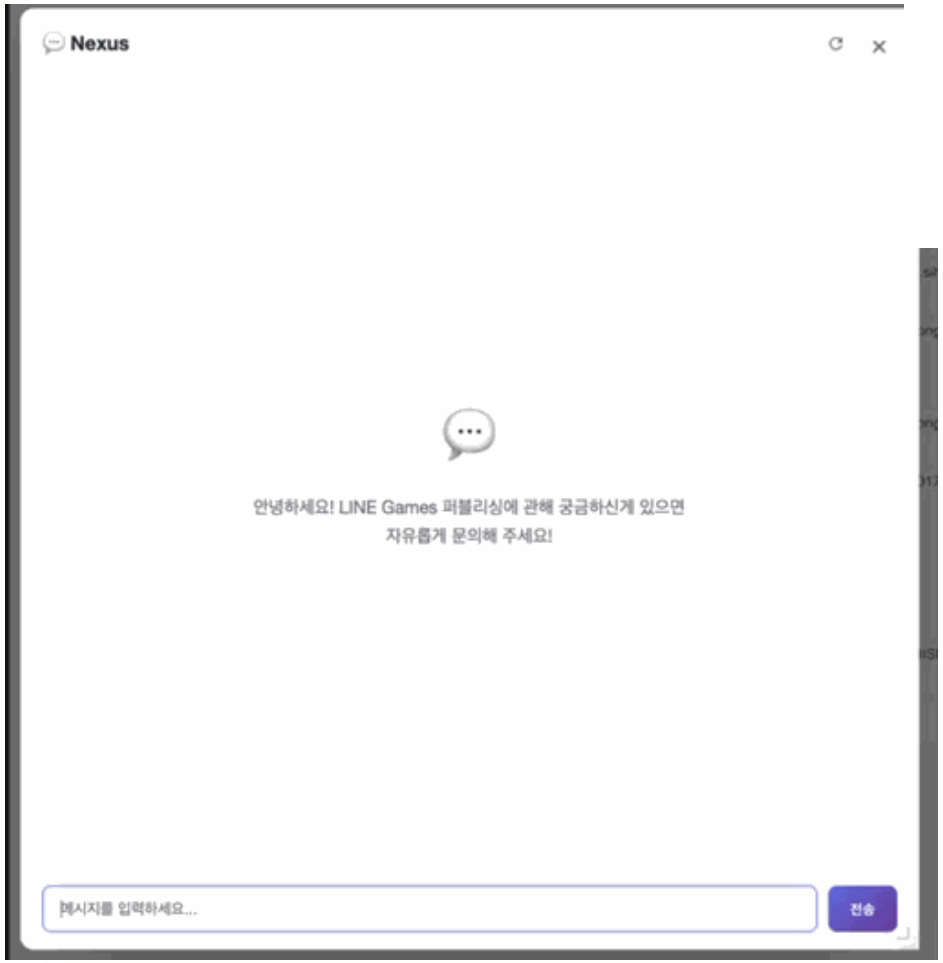




https://www.youtube.com/watch?v=IdFWE1jFC2w

퍼블리싱 가속화 에이전트 Part 1

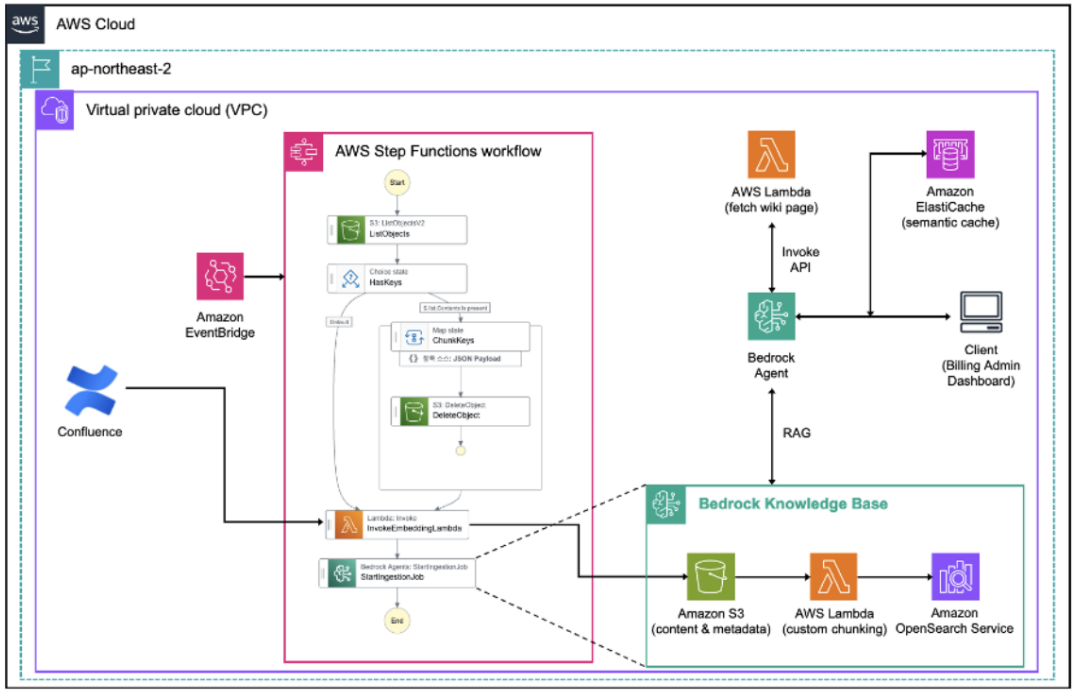
AWS Tech blog



AWS 기술 블로그

LINE Games의 AI Agent를 통한 게임 퍼블리싱 가속화 여정

by Jungwoo Song | on 10 12월 2025 | in Amazon Bedrock, Amazon Bedrock Agents, Amazon ElastiCache, Amazon OpenSearch Service, Amazon Simple Storage Service (S3), AWS Lambda, AWS Step Functions, Customer Solutions | Permalink | Share



[그림 1] Nexus AI 아키텍처

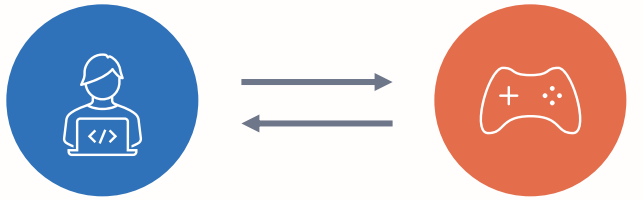
02

LINE Games의 퍼블리싱 가속화 여정

회사 소개



커뮤니케이션 코스트



개발자 게임 퍼블리싱

반복되는 질문

01 연동 시작점

이 SDK 연동 경험이 없는데,
어디서부터 시작해야 할까요?

02 문서 최신성

문서대로 했는데 에러가 나요.
혹시 최신 버전이 따로 있나요?

03 담당자 변경

담당자가 변경돼서
그동안의 히스토리를 잘 모르겠어요.

오늘 발표에서
여기에 답하겠습니다

작년 로드맵

Phase 1

Nexus AI



CHATBOT

챗봇
API 문서 기반 질의 응답

Phase 2

Nexus AI



AI AGENT

에이전트
도구를 골라 답변 · 샘플 코드

Phase 3

1



PUBLISHING DEV

개발자가 코드를 변경하면

2



CI/CD

CI/CD가 변경사항을 받아서

3



DOC AUTOMATION

문서를 작성하거나 수정

그 뒤, 달라진 설계

계획대로 되지 않은 세 가지 이유

01 코드 변경만으로는 문서를 못 고친다

변경의 의미가 diff에 없음

02 검증 없는 자동 반영은 위험하다

틀린 문서가 그대로 확산

03 개발자가 체감하지 못한다

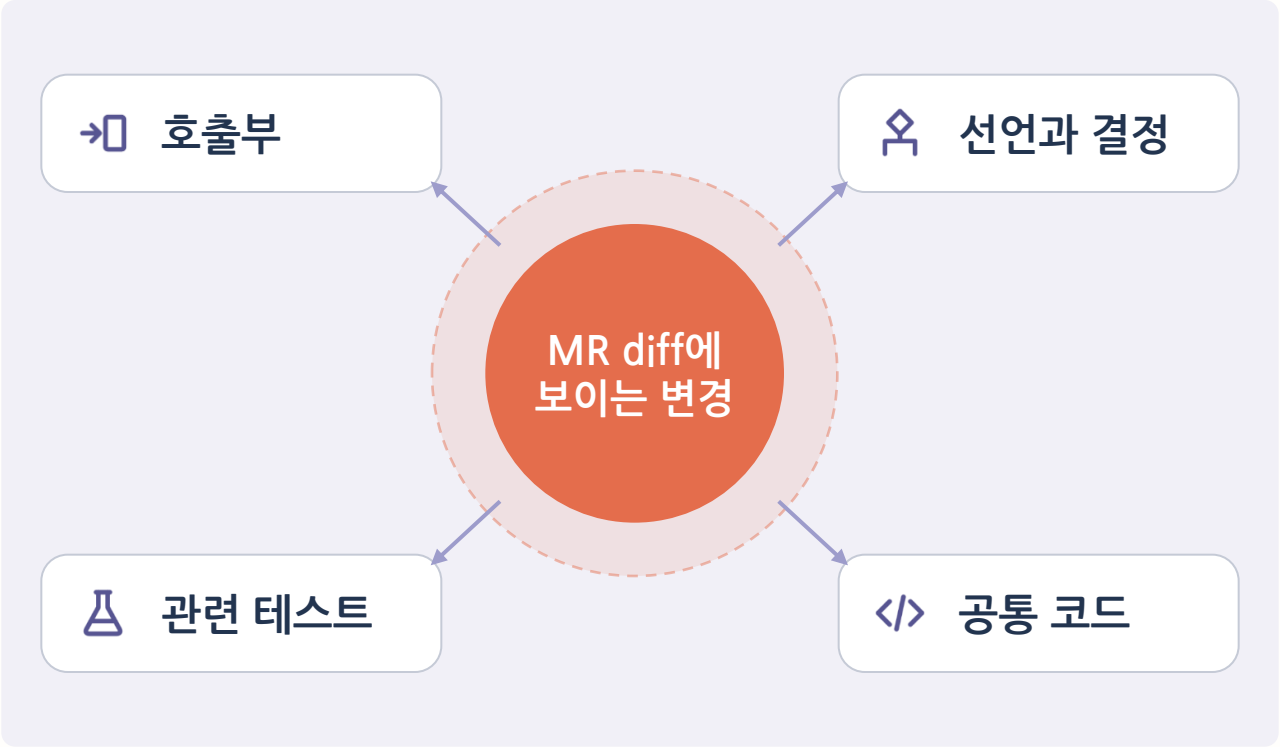
매일 하는 MR 리뷰에 들어가야 체감

달라진 설계

	계획	실행
도구	Q Developer CLI	Amazon Bedrock AgentCore 기반 리뷰 에이전트
경로	코드 → 문서	코드 → 리뷰 → 검증 → 문서
트리거	CI/CD 자동 반영	MR 리뷰: 사람이 보는 지점

코드 리뷰부터 시작한 이유

diff 밖의 영향



팀마다 다른 수준과 기준

팀 간 격차

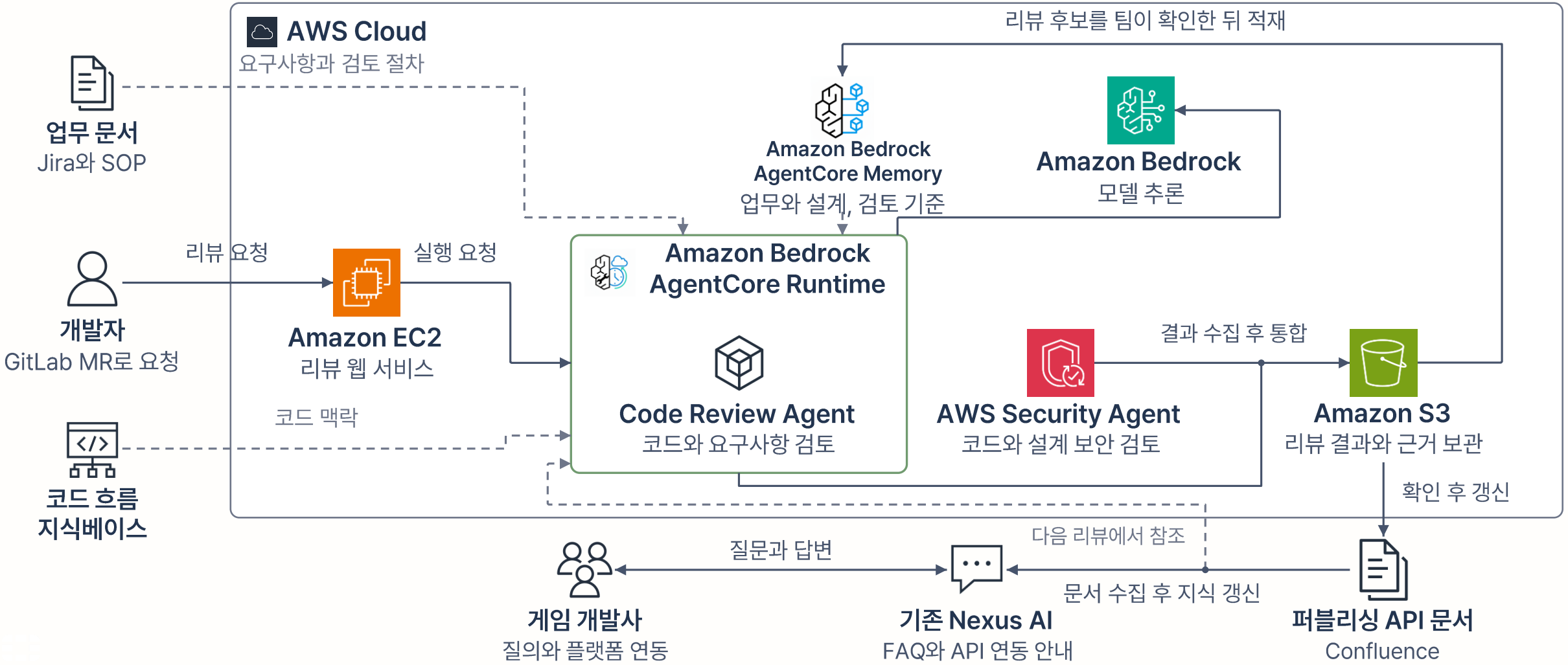
- 이미 쓰는 팀
- 시작 못한 팀

팀별 기준 차이

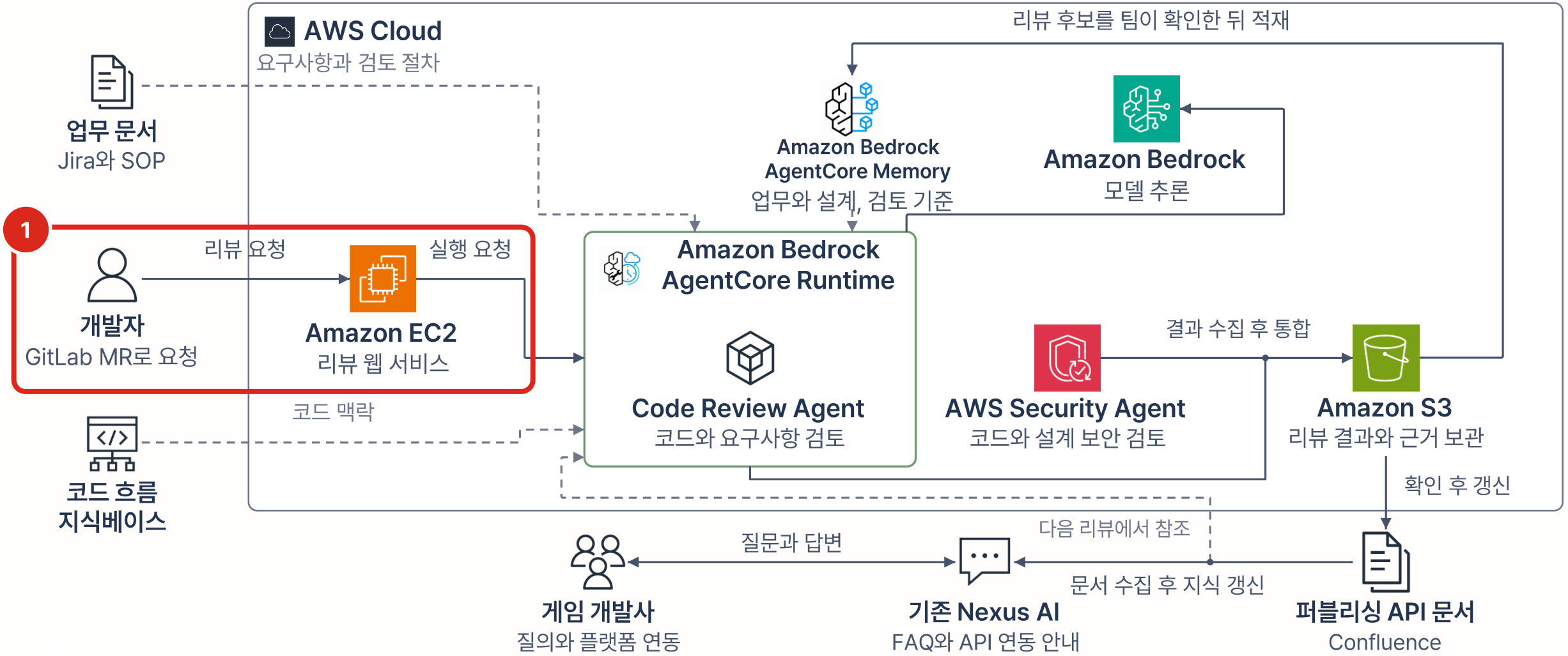
- Unity C#
- Unreal C++
- Terraform
- Java

도입이 아니라 승격, 팀의 관행을 조직 인프라로

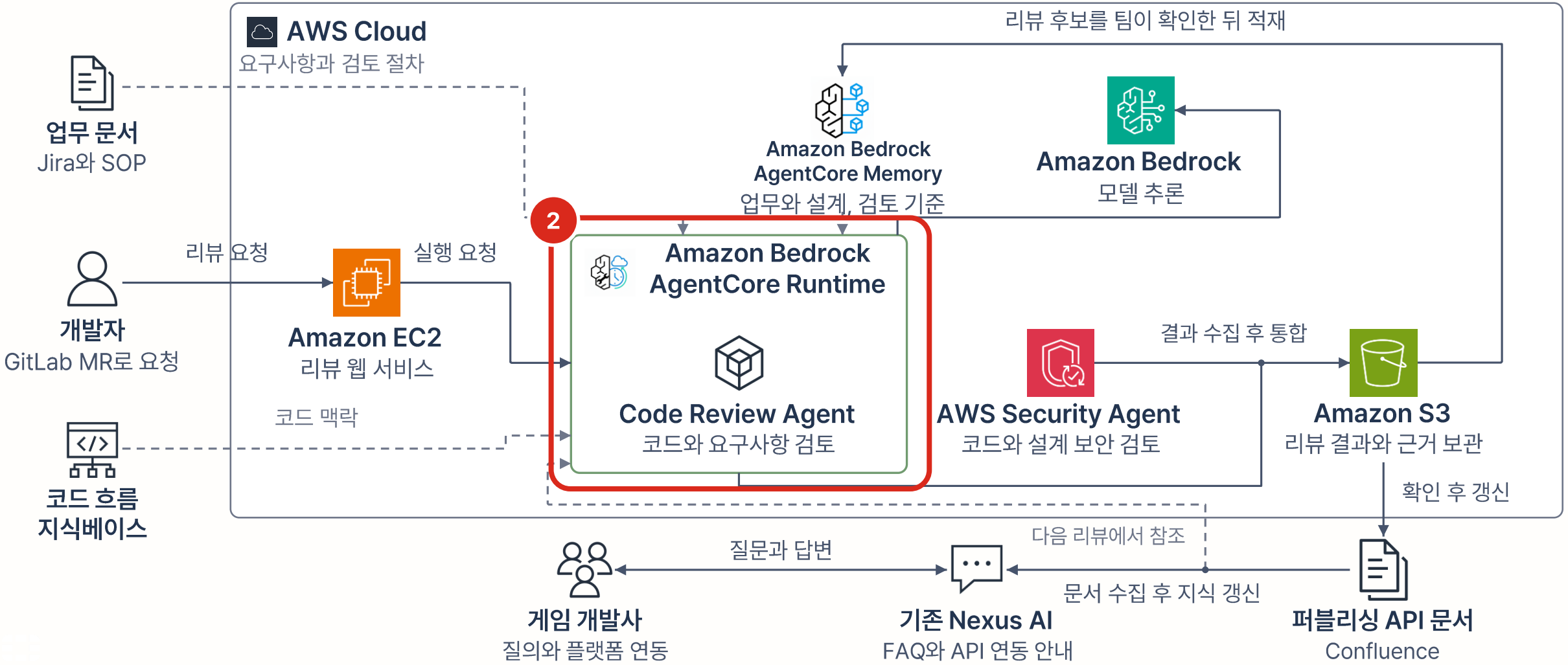
리뷰와 문서의 순환 구조


AWS Cloud
요구사항과 검토 절차
리뷰 후보를 팀이 확인한 뒤 적재
업무 문서
Jira와 SOP
Amazon Bedrock
AgentCore Memory
업무와 설계, 검토 기준
Amazon Bedrock
모델 추론
리뷰 요청
실행 요청
개발자
GitLab MR로 요청
Amazon EC2
리뷰 웹 서비스
Amazon Bedrock
AgentCore Runtime
Code Review Agent
코드와 요구사항 검토
결과 수집 후 통합
AWS Security Agent
코드와 설계 보안 검토
Amazon S3
리뷰 결과와 근거 보관
코드 맥락
코드 흐름
지식베이스
확인 후 갱신
다음 리뷰에서 참조
질문과 답변
문서 수집 후 지식 갱신
게임 개발사
질의와 플랫폼 연동
기존 Nexus AI
FAQ와 API 연동 안내
퍼블리싱 API 문서
Confluence

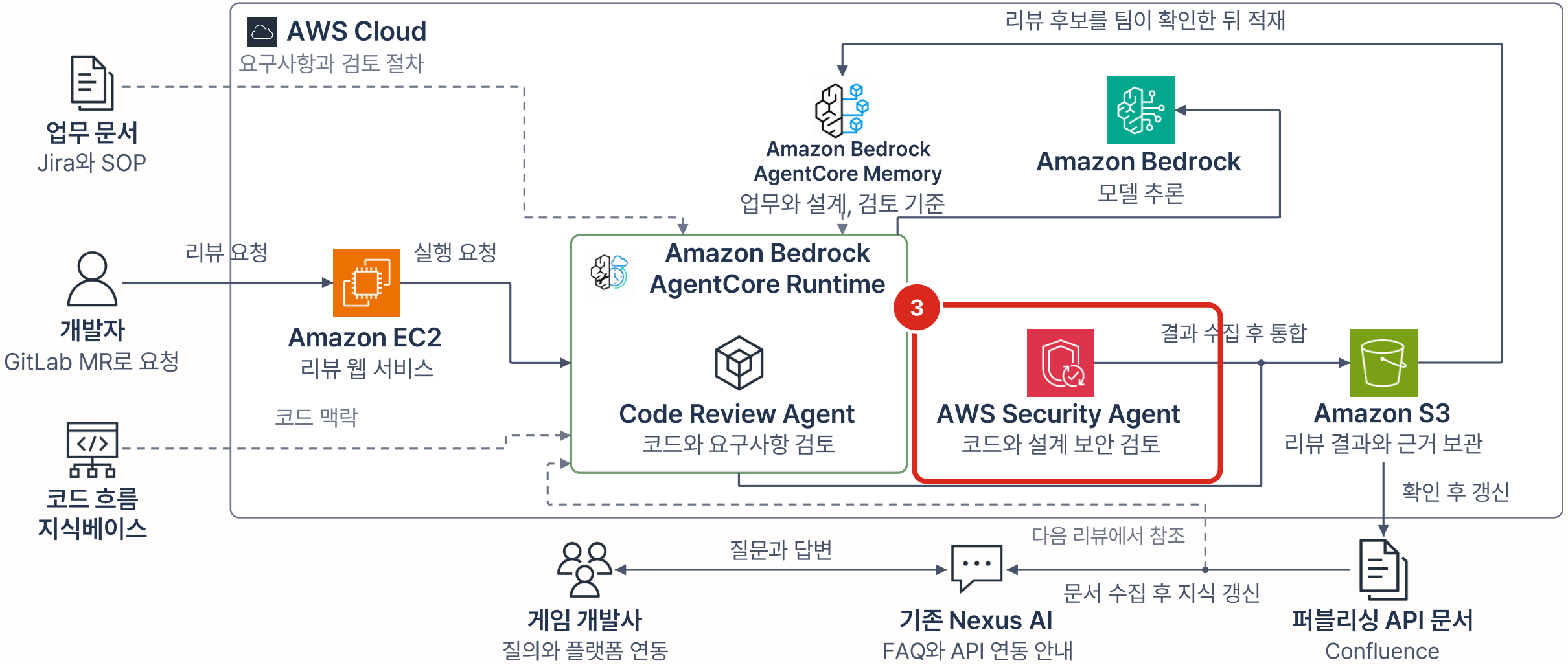
리뷰와 문서의 순환 구조


AWS Cloud
요구사항과 검토 절차
리뷰 후보를 팀이 확인한 뒤 적재
업무 문서
Jira와 SOP
Amazon Bedrock
AgentCore Memory
업무와 설계, 검토 기준
Amazon Bedrock
모델 추론
1
리뷰 요청
실행 요청
개발자
GitLab MR로 요청
Amazon EC2
리뷰 웹 서비스
Amazon Bedrock
AgentCore Runtime
Code Review Agent
코드와 요구사항 검토
결과 수집 후 통합
AWS Security Agent
코드와 설계 보안 검토
Amazon S3
리뷰 결과와 근거 보관
코드 맥락
코드 흐름
지식베이스
확인 후 갱신
다음 리뷰에서 참조
질문과 답변
문서 수집 후 지식 갱신
게임 개발사
질의와 플랫폼 연동
기존 Nexus AI
FAQ와 API 연동 안내
퍼블리싱 API 문서
Confluence

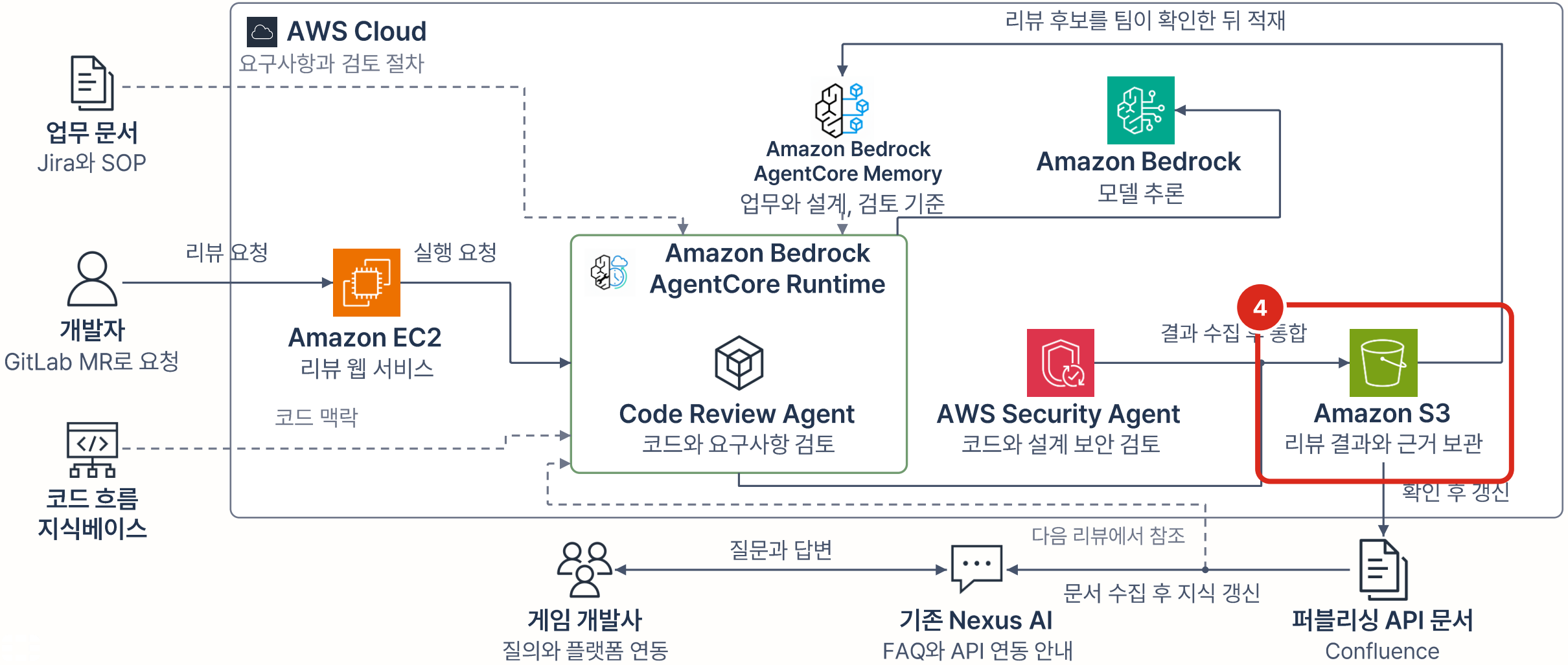
리뷰와 문서의 순환 구조


AWS Cloud
요구사항과 검토 절차
업무 문서
Jira와 SOP
개발자
GitLab MR로 요청
리뷰 요청
Amazon EC2
리뷰 웹 서비스
실행 요청
코드 맥락
코드 흐름
지식베이스
2
Amazon Bedrock
AgentCore Runtime
Code Review Agent
코드와 요구사항 검토
Amazon Bedrock
AgentCore Memory
업무와 설계, 검토 기준
Amazon Bedrock
모델 추론
리뷰 후보를 팀이 확인한 뒤 적재
AWS Security Agent
코드와 설계 보안 검토
결과 수집 후 통합
Amazon S3
리뷰 결과와 근거 보관
확인 후 갱신
다음 리뷰에서 참조
질문과 답변
문서 수집 후 지식 갱신
게임 개발사
질의와 플랫폼 연동
기존 Nexus AI
FAQ와 API 연동 안내
퍼블리싱 API 문서
Confluence

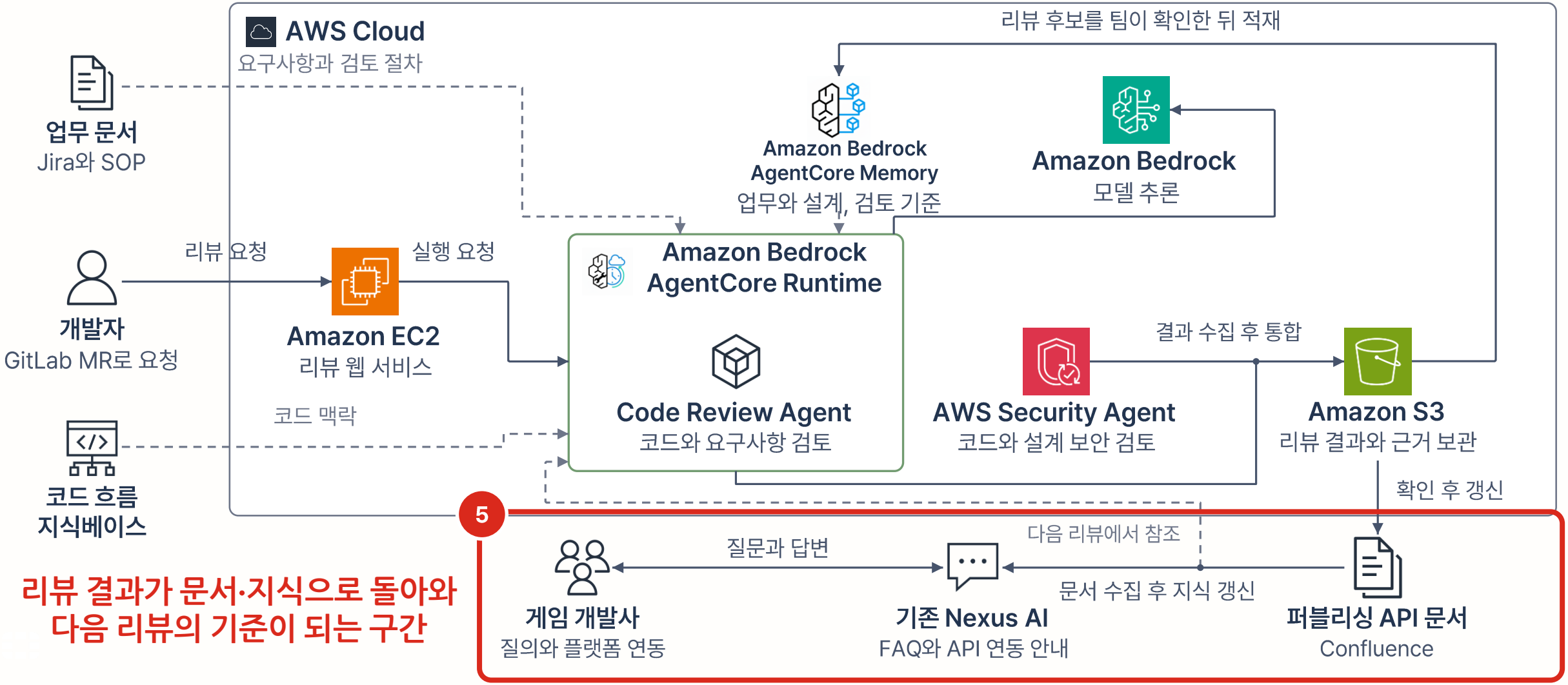
리뷰와 문서의 순환 구조


AWS Cloud
요구사항과 검토 절차
리뷰 후보를 팀이 확인한 뒤 적재
업무 문서
Jira와 SOP
Amazon Bedrock
AgentCore Memory
업무와 설계, 검토 기준
Amazon Bedrock
모델 추론
리뷰 요청
실행 요청
개발자
GitLab MR로 요청
Amazon EC2
리뷰 웹 서비스
Amazon Bedrock
AgentCore Runtime
3
결과 수집 후 통합
Code Review Agent
코드와 요구사항 검토
AWS Security Agent
코드와 설계 보안 검토
Amazon S3
리뷰 결과와 근거 보관
코드 맥락
코드 흐름
지식베이스
확인 후 갱신
다음 리뷰에서 참조
질문과 답변
문서 수집 후 지식 갱신
게임 개발사
질의와 플랫폼 연동
기존 Nexus AI
FAQ와 API 연동 안내
퍼블리싱 API 문서
Confluence

리뷰와 문서의 순환 구조


AWS Cloud
요구사항과 검토 절차
리뷰 후보를 팀이 확인한 뒤 적재
업무 문서
Jira와 SOP
Amazon Bedrock
AgentCore Memory
업무와 설계, 검토 기준
Amazon Bedrock
모델 추론
리뷰 요청
실행 요청
개발자
GitLab MR로 요청
Amazon EC2
리뷰 웹 서비스
Amazon Bedrock
AgentCore Runtime
Code Review Agent
코드와 요구사항 검토
4
결과 수집 후 종합
AWS Security Agent
코드와 설계 보안 검토
Amazon S3
리뷰 결과와 근거 보관
코드 맥락
코드 흐름
지식베이스
확인 후 갱신
다음 리뷰에서 참조
질문과 답변
문서 수집 후 지식 갱신
게임 개발사
질의와 플랫폼 연동
기존 Nexus AI
FAQ와 API 연동 안내
퍼블리싱 API 문서
Confluence

리뷰와 문서의 순환 구조


AWS Cloud
요구사항과 검토 절차
업무 문서
Jira와 SOP
개발자
GitLab MR로 요청
리뷰 요청
Amazon EC2
리뷰 웹 서비스
실행 요청
코드 맥락
코드 흐름
지식베이스
Amazon Bedrock
AgentCore Memory
업무와 설계, 검토 기준
Amazon Bedrock
모델 추론
리뷰 후보를 팀이 확인한 뒤 적재
Amazon Bedrock
AgentCore Runtime
Code Review Agent
코드와 요구사항 검토
AWS Security Agent
코드와 설계 보안 검토
결과 수집 후 통합
Amazon S3
리뷰 결과와 근거 보관
확인 후 갱신
5
게임 개발사
질의와 플랫폼 연동
질문과 답변
기존 Nexus AI
FAQ와 API 연동 안내
다음 리뷰에서 참조
문서 수집 후 지식 갱신
퍼블리싱 API 문서
Confluence
리뷰 결과가 문서·지식으로 돌아와
다음 리뷰의 기준이 되는 구간

1. 개인을 위한 리뷰 에이전트

리뷰 에이전트가 답하는 세 가지 질문

01

무엇을 내보내고 어떻게 검토하나?

데이터 경계 · NTSDK 검토 지침

절차가 도는 실행 화면

02

보낼 근거를 어떻게 고르는가?

① 리뷰 시점에 실시간 추적

② 미리 구축한 지식베이스 검색

03

리뷰가 무엇을 잡아냈나?

결제 정보 로그 노출 사례

diff 밖 호출 관계에서 찾은 문제

1. 개인을 위한 리뷰 에이전트

무엇을 내보내고 어떻게 검토하나?

데이터 경계

- 저장소 · 이력 · 자격증명은 로컬
- 비밀정보를 제거한 diff와 최소 문맥만 클라우드로
- AI 의견은 참고 정보 자동 승인 · 병합 없음

리뷰가 하지 않는 것

- 코드 수정 · commit · push 없음
- 저장소 상태를 바꾸지 않음

SDK 팀 검토 지침

- 코드 변경으로 모듈 설명이 달라지면 문서도 함께 갱신하고, 그대로면 이유를 남긴다
- 결제·인증·구매 브리지는 작은 수정이어도 보안 검토 대상으로 본다

에이전트가 따르는 절차

01 **변경 목적 확인**
02 **관련 문서 조회**
03 **영향 · 보안 검토**
04 **근거 기록**

절차가 도는 실행 화면



- 단계별 실행과 소요 시간 기록
- 절차를 문서로 관리 버전을 붙이고 되돌릴 수 있음
- 절차 내 완료 조건 포함

1. 개인을 위한 리뷰 에이전트

보낼 근거를 어떻게 고르는가? ① 리뷰 시점에 호출 관계를 실시간 추적

01 **변경 코드 파악**

바뀐 심볼·설정값과 주변 코드, MR이 언급한 경로

02 **분석 도구로 관계 추적**

코드 구조와 참조를 인덱싱해 호출 관계 확인 (AST·LSP 등)

03 **리뷰 Context 구성**

관련 코드·테스트를 골라 파일·줄·선택 이유와 함께 전달

근거가 부족하면 리뷰 중 재탐색

1 **누락 확인**

규칙 기반 점검으로 빠진 근거를 먼저 조회

2 **추가 탐색**

LLM이 필요한 도구를 요청해 다시 리뷰

3 **근거 검증**

실제 파일·위치·관계를 확인

정해진 횟수와 토큰 예산 안에서만 반복

1. 개인을 위한 리뷰 에이전트

보낼 근거를 어떻게 고르는가? ② 미리 구축한 코드 흐름 지식베이스(CodeKB) 검색

01 **구조와 호출 관계 추출**

정적 분석 도구로 함수·메서드를 뽑고 이름·타입으로 호출을 연결

02 **호출 관계 기반 설명 생성**

호출한 쪽 관점에서 역할과 영향을 3~5문장으로

03 **설명 벡터화와 저장**

Amazon S3 Vectors에 임베딩해 함수 ID·위치와 함께 보관

리뷰할 때 이렇게 씁니다

1 **자연어로 관련 로직 조회**

질문을 임베딩해 의미가 가까운 설명을 찾음

2 **원본 코드까지 연결**

함수 ID와 파일·라인으로 실제 코드 확인

3 **필요한 코드만 근거로 사용**

저장소 전체가 아니라 연결된 부분만 가져옴

구조 분석으로 못 찾는 것을 의미 검색이 보완

1. 개인을 위한 리뷰 에이전트

리뷰가 무엇을 잡아냈나?

고쳐야 할 것

병합 전에 고쳐야 할 것이 1건 있어요.

1 결제 인증 토큰과 영수증이 감사·재시도·응답 로그에 평문으로 기록됨

주요 신뢰도 85

1
- 위치: `payment/request.py:35-40`
- 영향:
  1. 로그 접근 권한만으로 사용자 인증 토큰과 영수증, 구매 토큰을 탈취할 수 있습니다.
  2. 정상 결제 경로에서도 항상 기록됩니다.

2
- 제안:
  1. to_dict/__repr__에서 auth_token과 receipt를 제외하거나 마스킹하고, 요청·응답 로그는 허용된 필드만 기록하십시오.
  2. 응답 body의 purchase_token도 마스킹하십시오.
- 검증: scripts/check_log_leak.py를 실행해 LEAK가 0건인지 확인하고, 이를 단위 테스트나 CI에 연결하십시오.
- 근거 해석: LLM 제안 신뢰도 93 → 검증 후 85

3
- 코드 근거:
  - to_dict와 __repr__가 auth_token, receipt를 포함한다. (`payment/request.py:35-40`)
  - to_dict 전체를 감사 로그에 기록한다. (`payment/audit.py:14`)
  - %r와 exc_info로 요청을 기록한다. (`payment/transport.py:28-33`)

연결된 Evidence


```
[
  {
    "path": "payment/request.py",
    "lines": "35-40",
    "note": "to_dict와 __repr__가 auth_token, receipt를 포함한다.",
    "evidence_id": "diff:payment/request.py",
    "codekb_sources": []
  },
```


1 **위치와 영향**

payment/request.py:35-40
인증 토큰과 영수증이 로그에 평문으로

2 **제안과 검증 방법**

마스킹할 필드와 확인 스크립트까지 함께

3 **코드 근거**

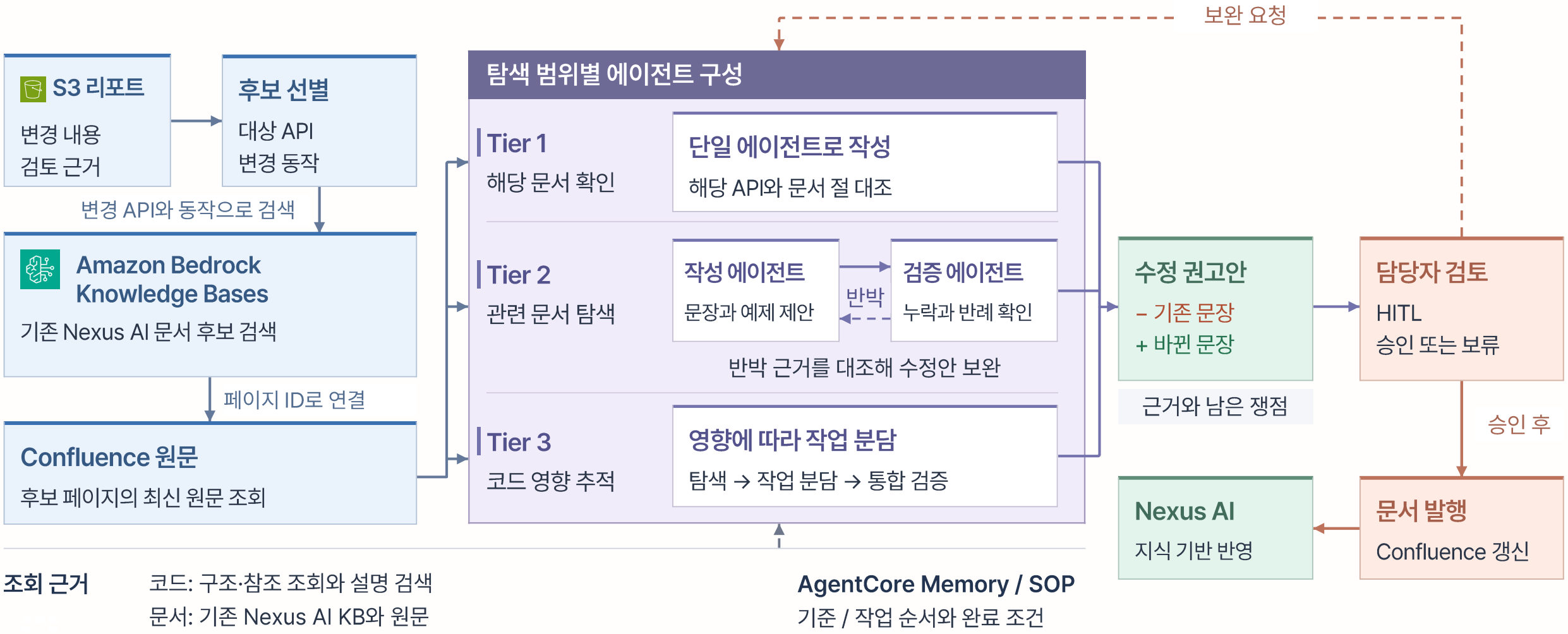
한 지적에 세 파일의 줄 번호가 붙음
리뷰와 문서 수정에 모두 사용

93 → 85 LLM 제안 신뢰도 → 검증 후 **확인된 지적만 남깁니다**

2. 팀 기준의 에이전트

Amazon Bedrock AgentCore Memory	= 기준	확정된 업무 규칙 · 예: 보상 지급은 서버 검증 결과 기준
Agent SOP = 순서		검토 범위 · 필수 검사 · 완료 조건
AWS Security Agent = 보안		같은 코드를 같은 기준으로 · 리뷰와 병렬 · 머지는 막지 않음

3. 문서 자동 갱신 에이전트



3. 문서 자동 갱신 에이전트

후보에서 문서 발행까지: 코드 근거를 붙여 사람이 승인

01 **갱신 후보 선별**
개발사가 알아야 할 변경만

02 **문서 탐색**
Nexus AI 검색 후 원문 대조

03 **작업 범위 결정**
한 구간 · 관련 문서 · 분담

04 **담당자 승인**
수정안과 근거 검토

▼ 선정 이유

V2 결제 요청에서 헤더 값과 요청·응답 본문이 로그에 노출되지 않도록 변경되었습니다. 기존 로그 안내는 요청과 응답 본문을 로그에서 확인할 수 있다고 설명하고 있어 변경된 동작과 맞지 않습니다. 결제 연동 가이드의 로그 안내 문단(P07)을 수정 대상으로 선정하고, 계속 기록되는 URL과 상태 코드도 함께 안내합니다.

수정 권고안 담당자 검토 대기

결제 로그에서 확인할 수 있는 정보가 달라져, 개발사가 장애를 확인할 때 참고하는 로그 안내를 갱신합니다.

본문을 로그에서 확인할 수 있다는 안내를 수정하고, URL과 상태 코드는 계속 남는다는 점을 명시합니다.

결제 연동 가이드 / 로그 안내 / 문단 P07

– 변경 전

로그 안내

결제 요청과 응답 본문은 로그에서 확인할 수 있습니다.

+ 수정안

V2 결제 요청의 헤더 값과 요청·응답 본문은 로그에서 숨깁니다.
URL과 상태 코드는 로그에 남습니다.

▶ 수정 근거 (3)

문서 검색 및 수정안 검토

검토 의견 (선택)

수용 보완 요청 제외

03

퍼블리싱 가속화 프레임워크

단일 에이전트의 한계

처음 시도: 단일 에이전트

에이전트 하나
좋은 모델 + 도구 연결

모든 판단을 위임
탐색 · 검증 · 결론까지

불가능하다고 판단
구조·참조는 구조 분석 도구가 더 정확

채택: 역할 분리

구조 분석 도구 — AST · LSP로 구조와 참조를 확정

LLM — 의미 검색으로 후보를 넓게 수집

확정된 기준 — Memory와 원본 코드로 최종 판단

저장소 하나로는 답할 수 없는 질문

관련 저장소

SDK 저장소
계정 전환과 결제 요청 처리

빌링 저장소
결제 API의 인증 정보 사용

플랫폼 저장소
계정 인증과 세션 처리

자연어 질문

계정이 바뀐 뒤 결제 요청은
어떤 인증 정보를 사용할까?

활용처

코드 리뷰
다른 서비스에 미치는 영향

문서 갱신
관련 API 문서의 수정 범위

질의 에이전트 구축 중
여러 저장소에 걸친 질문

3계층 프레임워크

개인에서 조직으로, 다시 조직에서 개인으로

조직 리뷰 → 문서 → 지식의 순환 · 조직 지식베이스 — 조직 지식

팀 Amazon Bedrock AgentCore Memory · AWS Security Agent · 팀별 추가 규칙 — 팀 기준

개인 SOP · CLI · 코드 어시스턴트 플러그인 — 개인 효율

핵심 **AX가 일회성 이벤트가 아니라 일상이 되는 구조**

퍼블리싱 안의 다른 직무로

PoC 진행 중

사업 · 기획 AX

Amazon Quick

비개발 조직이 자료를 직접 검색·분석

PoC 진행 중

Localization

내부 번역 Hub + AWS와의 협업

번역 품질과 반영 속도 동시 확인

변화 **실험 착수 부담 감소 · 검토에서 멈추던 시도를 일단 실행하고 판단**

05

다음 단계와 남은 교훈

무엇이 달라졌나

코드 변경 난이도별 리뷰 소요 시간과 비용

변경 난이도	리뷰 소요 시간 (사람 리뷰 / 리뷰 에이전트 / 단축분)	비용 (건당)
상 설계·하위 호환성까지 보는 변경 신규 기능, 구조 변경, 공용 인터페이스 수정	3~4시간 10~15분 94% 단축	$6~9 평균 $7.5
중 버그 가능성·개선 목표를 보는 변경 로직 수정, 리팩터링, 조건·예외 처리 변경	1~2시간 5~9분 92% 단축	$3~5 평균 $4
하 컴파일·기존 로직 영향만 보는 변경 설정값, 로그, 소규모 수정	30분 2~4분 90% 단축	$0.5~2 평균 $1.3
난이도 평균	110분 → 8분 93% 단축 · 약 15배	평균 약 $4

맞춰야 했던 세 시선

AX가 어려운 이유

조직이 원한 것

도입 속도와
눈에 보이는 성과

≠

개인이 할 수 있던 것

개인마다 다른
경험과 속도

→

기술 조직이 할 수 있던 것

팀과 개인의 격차를
줄이고 기준을 만들어
표준으로 제공하는 것

지난 시간이 남긴 다섯 가지

도구가 바뀌어도 남는 것

01	**코드에서 문서로 직행하는 길은 없다**	변경의 의미를 이해하는 단계를 둔다
02	**단일 에이전트가 아니라 역할 분리**	찾는 일과 판단하는 일을 나눈다
03	**AI의 기억에도 거버넌스가 필요하다**	확인된 문장만 출처·즉시 중지 경로와 함께 남긴다
04	**보안 리뷰는 비차단으로**	머지를 막지 않고 결과만 연결한다
05	**AX는 도입이 아니라 승격**	잘하는 팀의 관행을 조직 인프라로 올린다

AX는 도구의 수가 아닙니다

AX의 핵심

도구를 몇 개 쥐어주느냐가 아니라,
필요한 도구가 필요한 사람에게 가느냐



개인

각자 찾아낸 방식

→



팀

기준으로 합의

→



조직

인프라로 제공

aws FOR GAMES

Thank you

GS Neotek | MEGAZONE CLOUD