

AI Agent, 어디까지 **말길** 수 있는가

업무 시스템에 Agent를 올리기 위한
지식·권한·연동·운영의 설계 기준

Agentic AI

지식 오프로딩 · OKF

MCP · A2A

관측성 · 거버넌스

한눈에 보기

이 백서는 "AI를 도입했는데 성과가 보이지 않는다"는 질문에서 출발한다. 에스코어가 2024년부터 2026년까지 발행한 인사이드 리포트 9편을 토대로, Agent를 실제 업무 시스템에 올릴 때 조직이 먼저 정해야 할 네 가지 — 지식, 권한, 연동, 운영 — 을 정리했다.

이 백서의 구성

01

왜 파일럿은 늘었는데 업무는 그대로인가

도입 현실과 세 가지 실패 원인

03

읽기에서 쓰기로 — 권한을 넓히는 순서

권한 주체로서의 Agent, 승인 설계

05

무엇을 보아야 하는가 — 관측과 비용

AI 워크로드 지표, 평가 체계, 비용 구조

02

Agent가 일하려면 무엇이 준비되어야 하는가

지식 오프로딩, OKF, Agentic RAG, 검색 품질

04

연동의 표준 — MCP가 바꾸는 것

MCP 1.2세대, A2A, AGNTCY, Gateway

06

무엇부터 시작할 것인가

90일 실행 순서와 준비 항목

부록

AX 준비도 자가 점검표

5개 영역 20개 항목

88%

최소 한 개 업무 기능에서 AI를 쓰는
기업 비율
(2024년 78% → 2025년 88%)

1/3

파일럿을 넘어 확장 단계에 들어선
기업의 비율.
나머지는 여전히 실험 단계에 머문다

23%

조직 일부에서 에이전트 시스템을
확장 중이라고 답한 비율

출처: 맥킨지 2025 보고서. 에스코어 인사이드 리포트 「에이전틱 AI 구축 성공 전략」 (2026.01)에서 인용.

이 백서가 말하는 네 가지

1

Agent가 틀리는 이유는 대개 모델이 아니라 지식이다

조직이 합의한 지표 정의와 데이터 구조가 주어지지 않으면 Agent는 그것을 스스로 만들어낸다. 모델을 바꾸기 전에 조직의 지식을 어디에 어떤 형태로 적을지부터 정해야 한다.

2

Agent는 기능이 아니라 권한 주체다

사내 시스템에 접근하고 변경을 만든다면 그것은 자동화된 사용자다. 읽기 전용 작업으로 시작해 권한 경계와 감사 로그가 의도대로 작동하는지 확인한 뒤 쓰기 권한을 넓히는 것이 안전한 순서다.

3

연결 표준만 정해서는 운영이 되지 않는다

MCP와 A2A는 연결의 형식을 정할 뿐이다. 누구를 신뢰할지, 무엇을 허용할지, 문제가 생기면 어디서 끊을지는 발견·신원·정책·관측을 함께 설계해야 답이 나온다.

4

AI 워크로드의 운영 지표는 기존 시스템과 다르다

평균 응답시간만 보아서는 체감 성능을 설명할 수 없다. 토큰 처리량, 첫 토큰까지의 시간(TTFT), 이후 생성 속도(TPOT), KV 캐시, 그리고 인프라 비용과 토큰 비용의 이중 구조를 함께 보아야 한다.

핵심

해야 할 일은 **AI 프로젝트를 더 많이 시작하는 것이 아니라**, 조직의 업무와 책임 구조에 맞는 연결 기준을 세우고 그것을 안정적으로 운영하는 것이다.

왜 파일럿은 늘었는데 업무는 그대로인가

도입률은 올라갔다. 그런데 일하는 방식은 거의 바뀌지 않았다. 원인을 모델 성능에서 찾는 한 이 간극은 좁혀지지 않는다.

도입은 늘었고, 확장은 멈춰 있다

맥킨지 2025 보고서에 따르면 최소 한 개 업무 기능에서 AI를 사용한다고 답한 비율은 2024년 78%에서 2025년 88%로 올라갔다. 그러나 대다수 기업은 여전히 실험 또는 파일럿 단계에 머물러 있고, 확장 단계에 들어선 곳은 약 3분의 1에 그친다. 에이전트 시스템으로 좁혀 보면 조직 일부에서 확장 중이라는 응답이 23%, 실험을 시작했다는 응답이 39%다. 한편 EBIT의 5% 이상을 AI로 창출하는 기업은 에이전트를 더 자주, 더 다양한 분야에서 활용하는 경향을 보였다.

같은 시기 MIT 보고서는 AI를 도입한 기업의 95%가 기대에 못 미친다고 보고했다. 가트너 조사에서 AI 리더들이 실패 원인으로 꼽은 것은 모델이 아니라 **데이터 가용성과 품질 문제**였다.

용어 정리

생성형 AI는 지시가 있어야 움직이고 다단계 작업을 자율적으로 수행하지 못한다. **AI 에이전트**는 환경을 인식하고 판단하지만 사전에 정의된 규칙과 범위 안에서만 움직인다. **에이전틱 AI**는 스스로 목표를 세우고 계획을 짜며, 여러 에이전트가 협력하고 외부 도구를 활용한다. 생성형 AI가 "무엇을 만들어 달라"에 응답한다면, 에이전틱 AI는 목표 달성 자체에 응답한다.

세 가지 실패 원인

첫째, 조직 고유의 지식이 주어지지 않는다

에이전트는 우리가 제공하는 지식만큼만 똑똑하다. 모델 성능이 좋아져도 조직 고유의 데이터 구조와 지표 정의를 모르면 올바른 결과를 만들 수 없다. 문제는 이 지식이 메타데이터 카탈로그, 별도 문서, 코드 주석에 흩어져 있고 시스템마다 접근 경로도 다르다는 데 있다. 금융권 사례를 보면 정형 데이터가 전체의 약 80%를 차지하지만, 테이블 간 복잡한 조인 구조와 맥락 정보의 부재 때문에 AI가 컬럼명이나 코드값을 스스로 해석하기 어렵다.

둘째, 검증 체계 없이 출시된다

할루시네이션은 법률 분야 연구에서 69%에서 88%까지 나타났다. 2023년 미국의 한 변호사가 존재하지 않는 판례 6건을 인용해 제재를 받았고, 캐나다 항공사의 챗봇이 허위 환불 안내를 해 배상 판결이 난 사례도 있다. 에이전트는 구성 요소와 전체 시스템을 모두 검증해야 하며, 출시 후에도 성능 저하를 막기 위한 지속적인 모니터링이 필요하다.

셋째, 운영 기준이 없다

응답 속도는 곧바로 사용률로 이어진다. 구글은 5초 지연 시 반복 트래픽이 20% 줄었다는 사례를, 아마존은 100ms 지연마다 매출이 1% 감소했다는 사례를 제시한 바 있다. Forethought 보고서에 따르면 음성 AI 고객의 57%는 10분 이상 기다리지 않는다. 지연의 원인은 LLM 추론 비용, RAG 파이프라인의 누적 지연, 도구 호출의 네트워크 오버헤드처럼 여러 층에 흩어져 있어, 운영 지표를 따로 정의하지 않으면 원인을 짚어낼 수 없다.

짚어줄 점

빠를수록 항상 좋은 것은 아니다. 노력 휴리스틱에 따르면 적당한 지연이 신뢰감을 주기도 한다. 실제로 이메일 답변이 너무 빨라 "AI가 쓴 것 같다"는 불만을 받은 기업 사례가 보고되어 있다.

그래서 무엇을 정해야 하는가

위 세 가지는 각각 기술 과제처럼 보이지만, 실제로는 조직이 먼저 정해야 하는 **기준**의 문제다. 어떤 지식을 어디에 적을지(2장), 무엇까지 맡기고 어디서 사람이 승인할지(3장), 어떤 규격으로 연결할지(4장), 무엇을 측정해 성공과 실패를 판단할지(5장)가 정해져야 파일럿이 운영으로 넘어간다.

영역	정해야 할 것	정해지지 않으면 생기는 일
지식	지표 정의와 데이터 구조를 어디에 어떤 형식으로 기록할 것인가	Agent가 기준을 스스로 만들어내고, 부서마다 다른 숫자가 나온다
권한	어떤 작업까지 Agent가 직접 수행하고, 어디서 사람이 승인하는가	사고가 난 뒤에야 권한 범위를 논의하게 된다
연동	도구·에이전트 연결에 어떤 규격을 쓰고 누가 관리하는가	팀마다 서버와 키가 난립하고 교체 비용이 커진다
운영	어떤 지표로 품질·성능·비용을 판단하는가	"느리다"는 민원은 있는데 원인을 찾지 못한다

Agent가 일하려면 무엇이 준비되어야 하는가

Agent가 답을 내지 못하는 이유는 SQL 문법을 몰라서가 아니다. 어느 테이블과 컬럼을 골라야 하는지, 조직이 합의한 계산 기준이 무엇인지를 모르기 때문이다.

지식을 건네는 기존 방식의 한계

조직의 지식을 Agent에게 전달하는 방법은 크게 두 가지가 쓰여 왔다. 둘 다 한계가 분명하다.

- **전체 주입** — 지식이 늘면 컨텍스트 창을 넘긴다. 억지로 넣으면 무관한 내용이 섞여 정확도가 떨어지고, 호출마다 중복 주입되어 비용이 누적된다.
- **벡터 기반 RAG** — 문서를 규칙에 따라 자르므로 표 헤더와 본문이 분리되거나 단서 조항이 떨어져 나간다. 한 문서에 맞춘 분할 규칙이 다른 문서의 문맥을 깨뜨리기도 하고, 인덱싱과 재임베딩 비용이 계속 발생한다.

대안: 개념 단위로 지식을 미리 정리한다

문서 해석을 질의 시점이 아니라 **데이터 빌드 시점**에 끝내 두고, Agent는 정리된 결과를 읽게 하는 방식을 지식 오프로딩(Knowledge Offloading)이라 한다. 하나의 개념이 하나의 문서가 되고 경계를 사람이 정하므로, 기계적 청킹으로 문맥이 끊기지 않는다. 개별 파일이기 때문에 Git으로 변경 이력을 관리하고 코드 리뷰를 거칠 수 있으며, 무엇보다 **사람이 읽는 문서와 Agent가 읽는 문서가 같아진다.**

다만 조직마다 자체 규격을 쓰면 팀 간 지식 공유가 어렵다. 이 지점을 겨냥해 Google Cloud가 2026년 6월 공개한 개방형 규격이 **OKF(Open Knowledge Format)**다. 본질은 마크다운 파일이 담긴 디렉토리이며, 무거운 런타임이나 벡터 DB, 전용 SDK가 필요 없다. 규격이 정하는 것은 파일 구조뿐이라 생산자와 소비자가 서로의 시스템을 맞출 필요가 없다.

구성 요소	설명	대응되는 개념
bundle	개념 파일들이 모인 디렉토리 전체	git 레포 또는 디렉토리
concept	마크다운 개념 파일. 경로가 곧 식별자이며 한 개념의 모든 지식을 담는다	소스코드 파일 1개
index.md	디렉토리 내용을 링크와 한 줄 설명으로 정리한 진입점	README.md
frontmatter	파일 상단 YAML 블록. 요약 필드와 Type 필드로 문서 유형을 구분한다	코드 상단 주석
link	마크다운 링크로 다른 개념 파일을 가리키는 경로	문서 간 상호 참조

출처: 「OKF와 Agentic RAG: AI 에이전트에게 지식을 건네는 방법」(2026.08), 에스코어 방태연 프로.

정해진 구성 요소는 이것이 전부이고, 타입과 필드, 본문 내용은 작성자가 자유롭게 정한다. 번들은 한 번에 완성되지 않기 때문에 OKF는 소비자 쪽에 **best-effort** 원칙을 적용한다. 빈 필드는 건너뛰고 깨진 링크는 무시한다. "보낼 때는 엄격하게, 받을 때는 관대하게"라는 통신 원칙과 같다. 대신 소비 규칙이 관대한 만큼 지식의 품질은 시스템이 보장해 주지 않는다. 생명 주기 관리와 검수는 별도 프로세스로 엄격히 운영해야 한다.

읽는 방식도 함께 바뀌어야 한다 — Agentic RAG

지식을 정리해도 읽는 방식이 단발성 검색에 머물면 문서에 걸어 둔 링크는 쓰이지 않는다. Agent가 문서를 직접 열어 맥락을 파악하고 다음 탐색을 스스로 판단하는 방식을 Agentic RAG라고 한다. 네 지점에서 기존과 달라진다.

1 분해 (Decomposition)

복잡한 질문을 작고 명확한 단위로 나눈다. "지난 분기 지역별 매출 상위 고객"은 매출 지표, 지역 기준, 고객 식별 키워드는 세 질문으로 나뉘어 각각 탐색 경로를 갖는다.

2 진입 (Entry)

개별 문서를 열기 전에 index.md로 큰 그림을 파악하고 무엇을 열지 판단한다. 그래서 index.md의 한 줄 설명이 탐색 방향을 좌우한다.

3 이동 (Navigation)

링크를 따라 다음 문서로 이동한다. 본문의 링크가 이동 경로가 되고 주변 문장이 그 경로의 설명이 되므로, 별도의 관계 스키마가 필요 없다.

4 판단 (Evaluation)

수집한 문서로 답을 완성할 수 있는지 평가한다. 부족하면 탐색을 잇고, 충분하면 멈춘다. 찾지 못하면 그럴듯한 오답을 만드는 대신 무엇이 더 필요한지 되묻는다.

규모가 커질 때의 차이

문서가 늘어도 index.md는 큰 분류 몇 갈래만 남기고 세부는 하위 디렉토리로 나뉜다. 반면 벡터 인덱스는 모든 조각이 한 평면에 놓이고 질문과의 유사도로만 순위가 정해지므로, **규모가 커질수록 비슷한 조각이 늘어 정답을 가려내기 어려워진다.**

어떤 지식을 어느 방식에 맡길 것인가

두 방식은 대체재가 아니라 역할이 다르다. 운영 관점에서 비교하면 선택 기준이 분명해진다.

분류	벡터 기반 RAG	OKF + Agentic RAG
지식 생산	원문을 넣으면 자동 분할·저장. 초기 구축 비용이 낮다	개념 단위로 작성·검수해야 한다. 초기 구축 비용이 든다
잘 맞는 지식	로그, 회의록, 처리 이력처럼 경계가 흐린 대량의 자유 텍스트	스키마, 지표, 절차처럼 경계가 뚜렷하고 구조화 가능한 지식
변경 반영	수정 후 재임베딩과 배치 작업이 필요	파일 수정 후 PR 병합
변경 이력	인덱스 변경을 diff로 확인하기 어렵다	파일 diff와 log로 누가·언제·왜 바꿨는지 추적된다
모델 교체	인덱스 전체 재연산 및 재검증	원본 지식은 유지하고 인덱스만 재생성
평가 정답셋	분할 기준이 바뀔 때마다 정답 위치가 흔들린다	파일 경로 단위로 정답 위치가 고정된다
질의당 비용	한 번의 검색으로 끝나 지연 예측이 쉽다	여러 홉 탐색 시 호출 횟수와 지연이 늘어난다

정리하면 **스키마와 핵심 지표는 OKF로 오프로딩하고, 경계가 유동적인 텍스트는 기존 벡터 검색을 병행하는 것**이 현실적이다. 환각의 비즈니스 타격이 크고 전사 합의 기준이 필요한 엔터프라이즈 통합 검색이나 경영 의사결정 지원에는 지식 오프로딩이 유리하고, 문맥 경계가 유동적인 고객 문의·대화형 챗봇에는 기존 벡터 검색이 더 효율적이다.

검색 품질은 결합으로 올라간다

LLM은 확률에 기반해 답을 만들기 때문에 근거가 불분명한 출력을 낼 수 있다. 2021년 논문에서 비롯된 "확률적 앵무새"라는 표현은 그럴듯한 언어를 생성하되 그 의미를 이해하지는 못한다는 비유다. 이 한계를 줄이는 실무적 방법이 검색의 결합이다.

- **어휘 검색**(BM25 계열)은 정확한 명칭을 알 때 강하지만, 이름을 모르면 여러 번 시도해야 한다.
- **의미 검색**(벡터)은 특징만 알 때 유용하지만, 중요한 키워드를 놓치거나 원하지 않는 항목이 섞인다.
- **하이브리드 검색**은 둘을 결합한다. 결합 방식은 점수를 정규화해 가중 합산하는 CC(Convex Combination)와, 점수 대신 순위의 역수를 합산하는 RRF(Reciprocal Rank Fusion)가 쓰인다. CC는 세밀한 조정이 가능하고 RRF는 설정이 간단하다.

+8~12% 하이브리드 검색의 품질 개선폭 (어휘 검색 대비, nDCG@10)	+15% 하이브리드 검색의 품질 개선폭 (의미 검색 대비)	+0.024초 가장 큰 지연 증가를 보인 데이터셋의 p99 추가 소요 시간
--	---	--

출처: OpenSearch v2.10 벤치마크(7개 데이터셋). 「더 정확하고 유연하게 찾는 하이브리드 검색」(2025.06)에서 인용. 파인 튜닝한 하이브리드 검색은 어휘 검색 대비 평균 12.08% 높았다.

한 단계 더 정확도가 필요하면 2단계 검색을 쓴다. 하이브리드 검색으로 상위 k개를 추린 뒤 교차 인코더로 재순위화하는 방식이다. 교차 인코더는 정확하지만 비용이 크므로, 전체에 적용하지 않고 추려진 후보에만 적용해

속도와 정확도를 함께 얻는다.

관계를 묻는 질문에는 그래프가 필요하다

"팀 A가 속한 부서의 다른 팀이 담당하는 프로젝트 이름은?" 같은 다단계 질문에서 벡터 검색은 상위 K개 안에 연결 고리 문서가 빠지면 불확실한 답을 낸다. 데이터를 노드와 관계로 표현하는 지식 그래프를 함께 쓰면 팀→부서→팀→프로젝트라는 탐색 경로를 따라 근거가 분명한 답을 만들 수 있다. 다만 개체와 관계를 정확히 정의하고 속성을 정규화하지 않으면 의미 중복과 잘못된 관계가 생겨 결과가 왜곡된다. 자연어를 그래프 쿼리로 바꾸는 변환 품질과 쿼리 검증도 과제로 남는다.

임베딩 모델은 성능만으로 고르지 않는다

임베딩 모델 선택은 네 가지를 함께 본다. 디코더 기반 모델은 대개 고차원 벡터를 써서 품질이 좋지만 메모리와 연산 비용이 크고, 벡터 DB가 CPU에 의존하는 환경에서는 오히려 성능이 떨어질 수 있다. 독점 모델은 데이터를 외부로 보내야 할 수 있어 보안이 중요한 환경에는 부적합하다. 미세 조정이 필요 없다면 오픈 가중치 모델로 충분하고, 용도에 맞춘 조정이 필요하다면 오픈소스 모델이 유리하다.

한국어 환경

2024년 하반기 기준 MTEB 리더보드는 한국어 성능을 평가하지 않는다. Kor-IR 같은 사설 리더보드가 대안이지만 갱신이 비정기적이다. 검증되지 않은 모델을 쓰려면 공인 데이터셋을 번역하거나 자체 데이터셋을 정제해 여러 지표로 검증해야 한다. **최신 상태는 도입 시점에 다시 확인할 필요가 있다.**

정형 데이터를 AI가 읽게 만드는 일

기업 데이터의 대부분은 정형 데이터이지만, RDB의 스키마에는 비즈니스 로직이 적혀 있지 않다. 세션당 지출을 어떻게 계산하는지, 구매 없는 세션을 포함하는지, 환불을 어떻게 반영하는지는 조직의 합의이지 스키마의 정보가 아니다. 그래서 정형 데이터의 구조화는 세 단계로 진행된다.

- 1. AI 데이터 마트 구축** — 네이밍, 스키마 구조, 시간 차원처럼 RDB에는 없지만 AI 조회에 필요한 메타 정보를 더한다.
- 2. 지식 그래프 기반 정보 증강** — 벡터 검색이 놓치는 데이터 간 숨은 관계를 그래프로 보완한다.
- 3. 지능형 SQL 생성** — 자주 쓰는 질의는 미리 정의한 Static SQL로, 조건이 바뀌는 질의는 슬롯이 있는 Skeleton SQL로, 복잡한 자연어 질의는 Text2SQL로 처리해 속도와 유연성을 나눈다.

여기에 거버넌스가 따라붙는다. 온프레미스 환경에서의 암호화·접근 제어·감사 추적, 메타데이터를 부여한 데이터 카탈로그, 출처와 흐름을 추적하는 데이터 리니지, 변환·사용 이력 관리가 데이터 신뢰성의 기반이 된다.

2장 요약

모델과 에이전트는 계속 바뀐다. 그러므로 정제된 지식은 모델 바깥의 **독립적인 자산(Asset)**으로 남겨 두어야 시스템을 처음부터 다시 만드는 낭비를 피할 수 있다.

읽기에서 쓰기로 — 권한을 넓히는 순서

Agent가 사내 시스템에 접근하고 변경을 만든다면, 그것은 기능이 아니라 권한 주체다. 문제는 그 실행 경로가 결정적이지 않다는 데 있다.

Agent는 자동화된 사용자에게 가깝다

Agent는 문서를 검색하고, 코드를 쓰고, API를 호출하고, 티켓을 만든다. 성격으로 보면 서비스 계정에 가깝다. 그러나 일반 서비스 계정과 결정적으로 다른 점이 있다. **실행 경로가 LLM의 판단, 프롬프트 입력, 검색 컨텍스트, 외부 도구의 응답에 따라 매번 달라진다.** 같은 요청이라도 실행 경로와 비용, 위험이 달라질 수 있다는 뜻이다.

따라서 단순히 API 키를 하나 발급하는 것으로는 부족하다. Agent를 "AI 기능"이 아니라 권한 주체로 등록하고 신원(Identity), 권한(Permission), 정책(Policy), 감사(Audit)의 관점에서 관리해야 한다.

읽기와 쓰기는 다른 작업이다

장애 조사를 예로 들면 구분이 분명해진다. 메트릭과 로그, 쿠버네티스 이벤트, Git 변경 내역을 읽는 것은 읽기 작업이다. 그러나 롤백이나 설정 변경부터는 쓰기 작업이 되며, 별도의 권한 확인과 승인이 필요하다. 같은 시스템을 다루더라도 조회와 재시작은 위험 수준이 다르다.

단계	허용 범위	확인할 것	다음 단계로 넘어가는 조건
1단계	읽기 전용 Tool만 허용 (조회·검색·상태 확인)	권한 경계가 의도대로 작동하는가, 감사 로그가 빠짐없이 남는가	경계와 로그가 설계대로 동작함을 확인
2단계	되돌릴 수 있는 변경 (설정 제안, 초안 생성, 티켓 등록)	잘못된 실행을 사람이 즉시 알아차릴 수 있는가	오탐·오실행 시 복구 절차가 작동
3단계	운영 환경 변경 (배포, 롤백, 권한 변경, 삭제)	승인자가 판단에 필요한 근거를 함께 보는가	책임 주체와 복구 절차가 문서로 확정

「AI Agent 시대의 오픈소스 인프라 재편」 Part 1·2(2026.06, 2026.08), 에스코어 이현승 프로의 권고를 단계표로 재구성했다.

사람이 반드시 들어가야 하는 지점

쓰기, 삭제, 배포, 권한 변경은 human-in-the-loop 승인을 거치는 것이 원칙이다. 결제 승인이나 계약 체결처럼 되돌리기 어려운 작업, 그리고 모델의 확신도가 낮은 답변은 사람에게 에스컬레이션한다. AI는 사람을 대체하는 쪽이 아니라 사람의 능력을 높이는 쪽으로 설계할 때 운영에 안착한다.

Agent 실행 정책에 담겨야 할 것

- **도구와 권한의 범위** — 사용자의 직무와 대상 시스템에 따라 허용되는 Tool 목록이 달라진다. 같은 Tool이라도 누가 어떤 세션에서 호출하는지에 따라 허용 여부가 갈린다.
- **승인이 필요한 작업** — 어떤 작업이 승인 대상인지, 누가 승인자인지 미리 정한다.
- **감사 로그** — 요청 주체, Agent가 참고한 근거, 호출된 Tool, 운영 환경에 반영된 변경 사항이 **서로 연결된 형태로** 남아야 한다. 결과만 남으면 사후에 원인을 재구성할 수 없다.
- **결과 검사** — 응답에 비밀정보가 섞여 나가지 않는지 확인한다.

데이터가 밖으로 나가는 경로를 정의한다

사내 문서, 고객 정보, 소스코드, 장애 로그가 외부 모델 API로 전달되면 그 자체가 새로운 데이터 이동 경로가 된다. RAG와 Agent 환경에서는 검색 결과와 내부 API 응답까지 프롬프트에 섞이기 때문에 노출 범위가 더 넓어진다. 관리 방안은 다음과 같다.

- 전송 범위를 명시적으로 정의한다
- 민감정보를 자동 탐지·차단하고, 허용 목록 기반으로 필터링한다
- 모델 제공자의 데이터 보관·학습 정책을 확인한다
- 호출 로그를 남기고 Agent별로 정책을 분리한다

AI가 만든 코드에 대하여

AI가 생성한 코드에는 권한 검증 누락, 입력 검증 부재, 오래된 API 사용 같은 문제가 섞일 수 있다. 자동 수정 도구를 쓰더라도 그것은 검증의 대체가 아니라 **검증 가능한 보조**여야 하며, 기존 코드 스캐닝과 리뷰 프로세스 안에서 함께 작동해야 한다.

3장 요약

모든 자산을 한꺼번에 연결하려 하지 않는다. **연결 대상과 접근 경계를 먼저 파악하고, 읽기 전용 작업부터 검증한 뒤** 변경 권한을 넓히는 것이 사고를 줄이는 가장 확실한 순서다.

연동의 표준 — MCP가 바꾸는 것

연결의 형식을 표준화한다고 운영 문제가 풀리지는 않는다. 상대를 신뢰할지, 호출을 허용할지, 문제가 생기면 어디서 끊을지는 기업이 직접 설계해야 한다.

세 층으로 나누어 본다

Agent 연동은 하나의 규격으로 끝나지 않는다. 연결 규격, 발견과 인증, 운영과 관리의 세 층이 함께 있어야 작동한다. 연결 표준을 정하는 단계에서 나머지 두 층도 함께 정해야 한다는 뜻이다.

계층	구성 요소	역할
연결 규격	MCP, A2A	도구 접근과 Agent 간 메시지·작업 상태를 표현한다
발견·인증	Directory, Identity	Agent Card나 OASF 레코드를 검색하고, 상대를 식별해 자격 증명을 검증한다
운영·관리	Gateway, Observability	권한·승인 조건·사용량 제한을 적용하고 트레이스·메트릭·감사 로그를 모은다

MCP — Agent와 업무 도구의 연결

MCP(Model Context Protocol)는 AI 애플리케이션을 외부 데이터 소스와 도구, 워크플로에 연결하는 공개 표준이다. 메시지는 JSON-RPC 2.0 형식이며, 로컬은 stdio, 원격은 주로 Streamable HTTP로 통신한다. 서버가 제공하는 것은 세 가지다.

- **Tools** — API 호출, 데이터베이스 조회, 파일 변경 같은 실행 기능
- **Resources** — 모델에 제공할 수 있는 파일, 문서, 레코드 같은 데이터
- **Prompts** — 작업 지침과 입력을 미리 구성한 재사용 템플릿

인증 측면에서 HTTP 기반 MCP는 OAuth 2.1 흐름을 따른다. 2026년 6월 안정 버전으로 공개된 Enterprise-Managed Authorization(EMA)은 조직 IdP를 통해 접근을 중앙에서 관리하게 한다. 인사 이동이나 퇴사 시 기존 계정 절차로 MCP 권한까지 함께 조정할 수 있다는 점이 운영상 의미가 크다.

1세대와 2세대, 무엇이 달라졌나

초기 MCP는 인스턴스가 프로세스와 함께 상시 생성·유지되는 구조였다. 태스크 수와 MCP 수가 곱해지는 부하가 생기고, 상시 연결이라 서버리스 환경에 올리기가 어려웠다. 여기에 파일 시스템·웹 검색·셀 실행 같은 기능이 여러 툴에 내장되면서 MCP가 불필요한 중간 계층으로 인식되기 시작했다. 2세대 Stateless MCP는 이 구조를 바꾼다.

구분	Stateful (1세대)	Stateless (2세대)
연결 방식	stdio/SSE 상시 연결	단발성 HTTP POST
세션 상태	서버 메모리에 유지	무상태
단절 대처	핸드셰이크 재호출	요청 자체가 독립적으로 처리됨
세션 메모리	유휴 상태에도 지속 점유	대기 중 점유 없음
서버리스	적용 어려움	이식 및 호출 기반 실행 가능
적합 환경	단일 개발자의 로컬 IDE	분산 인프라, 대규모 원격 서비스

출처: 「새로운 2세대 MCP 규격 발표: Stateless MCP」 (2026.09), 에스코어 주경민 프로.

기술적으로는 요청에 메타데이터를 담는 **_meta** 필드가 도입되어 각 요청이 스스로를 설명하는 자가 설명적 형태가 되었고, W3C 분산 트레이싱 식별자를 함께 실어 여러 노드로 라우팅되어도 추적과 장애 분석이 가능해졌다.

도입 판단 시 유의

2세대는 1세대와 **하위 호환이 없다**. 구현체를 전면 수정하거나 호환 계층을 만들어야 하고, 서비스 제공자는 두 규격을 동시에 관리하게 될 수 있다. 2026년 9월 기준 주요 개발툴의 지원도 제한적이다. 당분간은 로컬 제어는 내장 도구가, 추적이 필요한 클라우드 연동은 Stateless MCP가, 기존 레거시는 OpenAPI가 맞는 **비대칭적 공존**이 현실적인 전망이다.

A2A — Agent끼리 일을 주고받는 규격

업무가 조직 경계나 전문 영역을 넘으면 여러 Agent가 나누어 처리한다. A2A는 이들이 메시지를 교환하고 작업 상태와 결과를 공유하는 방식을 정의한다. MCP와 A2A는 경쟁 관계가 아니라 한 작업 안에서 함께 쓰인다. MCP가 도구와 데이터 접근을 맡고, A2A가 Agent 간 작업 전달을 맡는다.

개념	역할
Agent Card	Agent의 엔드포인트, 기능, 지원 방식, 인증 요구사항을 외부에 알린다
Message	사용자나 Agent가 전달하는 통신 내용
Task	상태와 생명주기를 가진 작업 단위
Artifact	문서, 이미지, 구조화 데이터 같은 작업 결과물
Context	여러 Message와 Task를 하나의 업무 맥락으로 연결한다

도구 호출과 달리 Agent 작업은 오래 걸리고, 중간에 추가 정보나 승인이 필요하며, 결과물을 나누어 전달하기도 한다. 구매 Agent와 공급업체 Agent의 예를 보면, 구매 측은 상대의 내부 모델과 도구를 알 필요가 없다. 작업이 승인

대기 상태가 되면 같은 식별자를 담은 메시지로 승인 결과를 보내고, 승인 후 견적서를 결과물로 받아 작업을 완료한다.

AGNTCY와 Agent Gateway — 발견·신원·통제

Agent 수가 늘면 어떤 Agent가 있는지, 누가 관리하는지, 신뢰할 수 있는지, 어떤 경로로 연결할지를 알아야 한다.

구성요소	해결하는 문제	운영에서의 역할
OASF	Agent의 속성·기능을 설명하는 공통 형식	등록 정보의 표준화와 검증
Agent Directory	Agent 등록과 검색	소유자·기능·엔드포인트 탐색
Identity	Agent와 Tool의 식별·신원 확인	검증 가능한 자격 증명과 접근 정책
SLIM	Agent 간 안전한 메시징	Pub/Sub·스트리밍·암호화
Observability & Evaluation	다중 Agent 흐름의 분산 실행 정보	텔레메트리 수집과 성능·품질 평가

여기서 세 가지를 구분하는 것이 중요하다. **Directory 검색은 후보를 고르는 일, Identity 검증은 그 후보가 누구인지 확인하는 일, 권한 판단은 그 작업을 허용할지 정하는 일**이다. 셋을 하나로 뭉뚱그리면 "검증된 Agent이니 허용한다"는 잘못된 결론에 이른다.

Agent Gateway는 여러 경로에 흩어진 인증과 호출 로그를 하나의 공통 경로로 모은다. 사용자·Agent·MCP 서버의 신원과 권한을 확인하고, Agent와 Tool 단위의 허용 목록과 호출 제한, 승인 정책을 적용하며, Tool Call·A2A Task·모델 호출을 잇는 추적과 감사를 남긴다. 타임아웃, 재시도, 비용 한도 같은 운영 기준도 여기에 둔다. 게이트웨이가 없으면 팀별로 서버와 API 키가 난립한다.

관리 주체가 중립으로 옮겨가고 있다

2025년 12월 Linux Foundation이 출범시킨 Agentic AI Foundation(AAIF)에 MCP가 합류하면서 관리 주체가 한 기업에서 여러 참여자의 공개 절차로 옮겨졌다. 이후 A2A, AGNTCY, agentgateway도 2026년 6월에서 8월 사이에 차례로 합류했다. 도입 기업 입장에서는 특정 공급자의 결정에 덜 얽매이게 되고, 모델이나 플랫폼을 바꿀 때 연동을 처음부터 다시 설계할 부담이 줄어든다.

무엇을 보아야 하는가 — 관측과 비용

답변 품질이 나쁠 때 원인이 모델인 경우는 생각보다 적다. 검색 품질, 잘못된 도구 호출, 부적절한 모델 라우팅, 반복 호출이 더 흔한 원인이다.

AI 워크로드는 기존 애플리케이션과 다르게 움직인다

기존 웹 서비스의 CPU·메모리·QPS·지연 중심 운영으로는 AI 워크로드를 설명하기 어렵다. LLM은 가속기에 의존하고, 프롬프트 길이와 출력 토큰 수, KV 캐시 사용량에 따라 부하가 크게 달라진다. Agent는 검색·API 호출·코드 실행·도구 호출·재시도·응답 생성을 반복하므로 장애 지점도 모델 서버, 벡터 DB, 내부 API, 인증 시스템, 외부 SaaS로 흩어진다.

관점	기존 애플리케이션	AI 워크로드
자원	CPU / 메모리	GPU·TPU, KV cache
확장·성능	요청 수, CPU 사용률	토큰 처리량, TTFT / TPOT, 배치 효율
장애 분석	로그·메트릭·트레이스	프롬프트, 모델 호출, Tool 호출, 검색 결과
비용	인스턴스·Pod 단위	GPU 시간, 토큰 사용량, 모델별 단가
관측성	서비스 단위	Agent 실행 흐름과 Tool Call 트레이스

평균 지연으로는 설명되지 않는다

체감 성능을 보려면 **TTFT**(첫 토큰까지의 시간)와 **TPOT**(이후 토큰 생성 속도)를 함께 봐야 한다. 평균 응답시간이 같아도 TTFT가 길면 사용자는 "멈춰 있다"고 느낀다.

무엇을 수집할 것인가

영역	측정 항목
자원 사용	모델별 GPU·TPU 사용량, utilization, 메모리
추론 성능	토큰 처리량, TTFT, TPOT, 배치 효율
비용	토큰 사용량, GPU 시간, 모델별 비용, 요청당 비용
Agent 실행	단계별 실패 지점, 반복 Tool Call 여부
RAG 품질	검색 결과 품질, 문서 최신성, 벡터 DB 지연
감사·증적	Prompt, Model Call, Tool Call 트레이스와 API 결과
안정성	cold start와 scale-out 시간, endpoint 장애율

관측의 범위도 넓어졌다. 예전에는 개별 서비스의 호출을 보았다면, 이제는 Agent·Task·메시지·Tool Call·모델 호출이 하나의 실행 흐름으로 묶여 추적된다. 감사 기록에는 실행 결과만이 아니라 **요청 주체와 Agent가 참고한 근거, 호출된 Tool이 함께** 남아야 한다.

표준화 흐름

이 영역에서는 공통 기반을 만들려는 움직임이 진행 중이다. AI 워크로드를 위한 쿠버네티스 기준선과 검증 체계(AI Conformance), GPU와 가속기를 세밀하게 요청·공유하는 기반(Dynamic Resource Allocation), 프롬프트 길이와 예상 출력 토큰, 캐시 상태를 반영하는 LLM 인지형 라우팅(Gateway API Inference Extension), 벤더에 종속되지 않는 AI 관측 데이터 구조(OpenTelemetry GenAI Semantic Conventions)가 대표적이다.

품질은 평가 체계로 관리한다

- 1 오프라인 평가**
골든 데이터셋을 만들고 RAGAS·DeepEval 같은 도구로 검색 품질과 생성 품질을 측정한다. 배포 전에 회귀를 잡는 단계다.
- 2 온라인 평가**
사용자 피드백과 함께 재질문 빈도, 이탈률 같은 암묵적 신호를 본다. 사용자는 대개 "틀렸다"고 말하지 않고 그냥 떠난다.
- 3 LLM-as-a-Judge**
강력한 모델로 출력 품질을 자동 평가한다. 사람 검수의 범위를 좁혀 주는 보조 수단으로 쓴다.

평가 체계를 OKF 기반 지식과 함께 쓰면 진단이 쉬워진다. 정답을 파일 경로로 지정할 수 있어 **Agent가 올바른 문서를 읽었는지와 SQL 작성이 틀렸는지**를 분리해 볼 수 있기 때문이다.

비용은 두 겹이다

AI 운영 비용은 인프라 비용과 토큰 비용의 이중 구조다. 인프라 쪽은 GPU 사용률, 유휴 GPU, 배치 효율, 캐시 활용이 변수이고, 토큰 쪽은 프롬프트·완성 토큰, 모델별 단가, Agent의 반복 호출이 변수다.

비용 최적화에 대한 오해

비용 최적화는 값싼 모델을 고르는 일이 아니다. **요청 라우팅, 반복 호출 제한, 캐시 활용을 정책으로 관리하는 일**이다. 질의 복잡도에 따라 모델을 나누어 보내고, 같은 질문의 반복 호출을 캐시로 끊고, 호출 한도를 게이트웨이에서 거는 쪽이 훨씬 크게 작동한다.

실행 계층에서는 추론 엔진 최적화(PagedAttention, Continuous Batching), 시맨틱 캐싱, 모델 티어링, 스트리밍 응답을 함께 적용한다. 민감 데이터는 온프레미스 오픈소스 모델로, 복잡한 추론은 상용 모델 API로 보내고 그 사이를 라우팅 레이어가 중개하는 하이브리드 아키텍처가 보안과 비용을 함께 다루는 현실적인 형태다.

무엇부터 시작할 것인가

전면 도입은 실패 확률이 높다. 좁은 범위에서 빠르게 검증하고, 검증된 것만 넓히는 순서가 결국 가장 빠르다.

원칙 다섯 가지

- 1. 작게 시작하고 빠르게 반복한다. 부서별 FAQ 봇처럼 범위가 좁은 유스케이스로 시작해 검증 후 확장한다. 2~4주 단위의 MVP 사이클이 현실적이다.
- 2. 평가 체계를 먼저 세운다. 평가 기준 없이 만든 시스템은 좋아졌는지 나빠졌는지 말할 수 없다.
- 3. 사람을 설계에 넣는다. 고위험 작업은 사람이 확인하고, 확신도가 낮은 답변은 에스컬레이션한다.
- 4. 하이브리드로 간다. 민감 데이터는 내부에서, 복잡한 추론은 외부 API에서 처리하고 라우팅으로 나눈다.
- 5. 조직 역량을 함께 키운다. 특정 프레임워크에 깊이 의존한 코드는 마이그레이션 비용이 크다. 기술 선택을 되돌릴 수 있게 설계하고, 실패를 학습으로 전환하는 문화를 만든다.

90일 실행 순서

기간	할 일	끝났을 때 손에 남는 것
1~2주	AI 워크로드 인벤토리 작성(LLM API, RAG 앱, Agent runtime, 벡터 DB). 연결 대상과 접근 경계 파악	무엇이 어디에 연결되어 있는지에 대한 현황도
3~4주	유스케이스 1건 선정. 평가용 골든 데이터셋 작성. 읽기 전용 Tool만으로 MVP 구성	품질을 숫자로 말할 수 있는 기준선
5~8주	핵심 지표·스키마를 개념 문서로 오픈로딩. 하이브리드 검색 적용. 관측 항목 수집 시작	재사용 가능한 첫 번째 지식 번들
9~12주	Gateway 도입(인증·정책·호출 기록 중앙화). 승인 흐름 설계 후 되돌릴 수 있는 변경부터 쓰기 권한 부여	권한 경계와 감사 로그가 검증된 운영 체계
이후	A2A·발견/신원 체계 도입, 적용 범위 확대, 비용 배분 모델 수립	확장 가능한 Agent 운영 기반

플랫폼 조직이 준비할 항목

- AI 워크로드 인벤토리 — 무엇이 돌고 있는지 목록부터 만든다
- GPU·TPU 자원 관리 — quota, 우선순위, 예약, 유휴 자원 회수

- 추론 엔드포인트 표준화
- AI Gateway 도입 — 호출 정책, 인증, 라우팅, 호출 제한의 중앙화
- 관측성 확장 — Prompt, Model Call, Tool Call 추적
- 비용 배분 모델 — 팀·서비스·Agent 단위 가시화
- Agent 실행 정책 — 권한, 도구 범위, 승인 작업, 감사 로그

마지막으로

모델과 프레임워크는 계속 바뀐다. 그러므로 투자해야 할 곳은 특정 모델이 아니라 **바뀌어도 남는 것** — 정제된 지식, 권한과 승인 구조, 관측과 감사 체계 — 이다.

AX 준비도 자가 점검표

다섯 개 영역, 스무 개 항목이다. 해당하는 항목에 표시해 보면 지금 조직이 어느 단계에 있는지, 다음에 무엇을 해야 하는지가 드러난다.

1. 지식 — Agent에게 건넬 것이 정리되어 있는가

- ☒ 핵심 지표의 계산 기준이 문서로 합의되어 있고, 부서마다 같은 숫자가 나온다
- ☐ 데이터 스키마에 네이밍·시간 차원 등 조회에 필요한 메타 정보가 붙어 있다
- ☐ 지식 문서의 변경 이력을 추적할 수 있다(누가·언제·왜 바꿨는지)
- ☐ 사람이 읽는 문서와 AI가 읽는 문서가 같은 것을 쓴다

2. 권한 — 무엇까지 맡길지 정해져 있는가

- ☒ Agent가 신원을 가진 권한 주체로 등록되어 있다(공용 API 키가 아니다)
- ☐ 읽기 작업과 쓰기 작업이 구분되어 권한이 분리되어 있다
- ☐ 배포·삭제·권한 변경에 사람의 승인 단계가 있다
- ☐ 외부 모델로 나가는 데이터의 범위가 정의되어 있고 필터링이 작동한다

3. 연동 — 연결 방식이 통제되고 있는가

- ☐ 어떤 Agent와 도구가 어디에 연결되어 있는지 목록이 있다
- ☒ 인증과 호출 기록이 하나의 공통 경로(Gateway)를 지난다
- ☐ 연결 규격을 바꿀 때 영향 범위를 파악할 수 있다
- ☐ 조직 IdP와 연동되어 인사 이동·퇴사 시 권한이 함께 조정된다

4. 운영 — 무엇이 일어나는지 보이는가

- ☐ 토큰 처리량, TTFT, TPOT를 수집하고 있다
- ☒ Prompt·Model Call·Tool Call이 하나의 실행 흐름으로 추적된다
- ☐ 인프라 비용과 토큰 비용을 팀·서비스 단위로 나누어 볼 수 있다
- ☐ 장애 시 어느 단계에서 실패했는지 짚어낼 수 있다

5. 검증 — 좋아졌는지 말할 수 있는가

- ☒ 배포 전 품질을 측정하는 골든 데이터셋이 있다
- ☐ 재질문 빈도·이탈률 같은 사용 신호를 모으고 있다
- ☐ 검색이 틀린 것인지 생성이 틀린 것인지 분리해 진단할 수 있다
- ☐ 출시 후에도 품질 저하를 감지하는 모니터링이 돌고 있다

읽는 법

5개 이하라면 지식과 권한 정의부터 시작한다. **6~13개**라면 기반은 있으나 연결과 관측이 비어 있을 가능성이 높다. **14개 이상**이라면 적용 범위를 넓히면서 비용 구조를 설계할 단계다. 노란색으로 표시된 다섯 항목은 각 영역에서 가장 먼저 확보해야 할 항목이다.

에스코어가 하는 일

에스코어는 삼성SDS 자회사로, IT 경영 컨설팅과 소프트웨어 기술 서비스를 함께 보유한 B2B IT 서비스 기업이다. 오픈소스 도입부터 아키텍처 설계, 거버넌스, 구축, 기술 지원까지를 하나의 흐름으로 연결한다.

AI 영역에서는 RAG와 Agentic AI 구현 경험을 바탕으로 AI 서비스 구축을 지원하며, 온프레미스 기반 데이터 플랫폼(AI-DIVE)으로 기업 내부 데이터를 AI가 활용할 수 있는 형태로 준비하는 일을 함께 수행한다. Red Hat 제품군 기술지원 분야에서는 거버넌스 체계 확립, 라이선스 및 보안 취약점 관리, 기술지원을 포함한 원스톱 서비스를 제공한다.

AX 준비도 진단 | 위 점검표를 바탕으로 한 진단과 다음 단계 제안을 받아 보실 수 있습니다.

s-core.co.kr · 서비스 > AI Service > AX 준비도 진단 받기

출처

이 백서는 에스코어 인사이트 리포트 9편을 재구성한 것이다. 각 리포트는 s-core.co.kr 인사이트 메뉴에서 전문을 볼 수 있다.

기반이 된 인사이트 리포트

- [1] 새로운 2세대 MCP 규격 발표: Stateless MCP — 주경민 프로, 소프트웨어사업부 개발플랫폼팀 (2026.09)
- [2] OKF와 Agentic RAG: AI 에이전트에게 지식을 건네는 방법 — 방태연 프로, 소프트웨어사업부 플랫폼개발팀 (2026.08)
- [3] AI Agent 시대의 오픈소스 인프라 재편: Part 2. 관측성·연동·거버넌스 — 이현승 프로, 오픈소스사업부 (2026.08)
- [4] AI Agent 시대의 오픈소스 인프라 재편: 보안과 운영 관점에서 — 이현승 프로, 오픈소스사업부 (2026.06)
- [5] 에이전틱 AI 구축 성공 전략: AI 도입 현실과 Red Hat AI 활용 방안 (2026.01)
- [6] AI의 도메인 지식 활용을 위한 필수 도구: AI-Ready 데이터 플랫폼 (2025.10)
- [7] 데이터 속 숨은 관계를 찾아주는 Graph RAG의 힘 (2025.10)
- [8] 더 정확하고 유연하게 찾는 하이브리드 검색: LLM의 '확률적 앵무새'를 넘어 (2025.06)
- [9] 벡터 임베딩 모델의 이해와 활용: 최적의 임베딩 모델 찾기 (2024.11)

리포트가 인용한 외부 자료

- 맥킨지 2025 AI 현황 보고서 — 도입률 78%→88%, 확장 단계 약 1/3, 에이전트 확장 23%·실험 39% [5]
- MIT 보고서 — AI 도입 기업의 95%가 기대 미달 [6]
- 가트너 조사 — AI 리더가 꿈은 실패 원인으로서의 데이터 가용성·품질 [6]
- OpenSearch v2.10 하이브리드 검색 벤치마크 — 7개 데이터셋, nDCG@10 기준 [8]
- Bender et al.(2021) — '확률적 앵무새(Stochastic Parrots)' 개념 [8]
- Google Cloud — Open Knowledge Format(OKF), 2026년 6월 공개 [2]
- Linux Foundation — Agentic AI Foundation(AAIF), 2025년 12월 출범 [3]
- 응답 지연 관련 사례(구글 5초/20%, 아마존 100ms/1%), 법률 분야 할루시네이션 69~88%, Forethought 보고서(음성 AI 고객 57%) [5]

읽을 때 유의할 점

본문의 수치와 사례는 위 리포트가 인용한 값을 그대로 옮긴 것이다. 원 보고서의 발표 시점이 다르므로, 의사결정에 사용할 때는 **해당 시점의 최신 자료를 다시 확인할 것**을 권한다. 특히 MCP 2세대 지원 현황, 임베딩 모델 한국어 평가 체계, 각 표준의 관리 주체는 변화가 빠른 영역이다.

S-Core

기업 핵심 가치, 그 이상의 혁신

We Think Beyond Core

AI Service · AX 전략 수립부터 AI Agent 구축·운영까지

Biz Consulting · 일하는 방식의 재설계부터 데이터·마케팅까지

IT 플랫폼 구축 · 개발·배포·운영, 데이터, 클라우드를 하나의 체계로

Application 구축 · 노후 시스템의 클라우드 네이티브 재설계

오픈소스 Full Stack 서비스 · 거버넌스부터 전환 구축, 기술지원까지