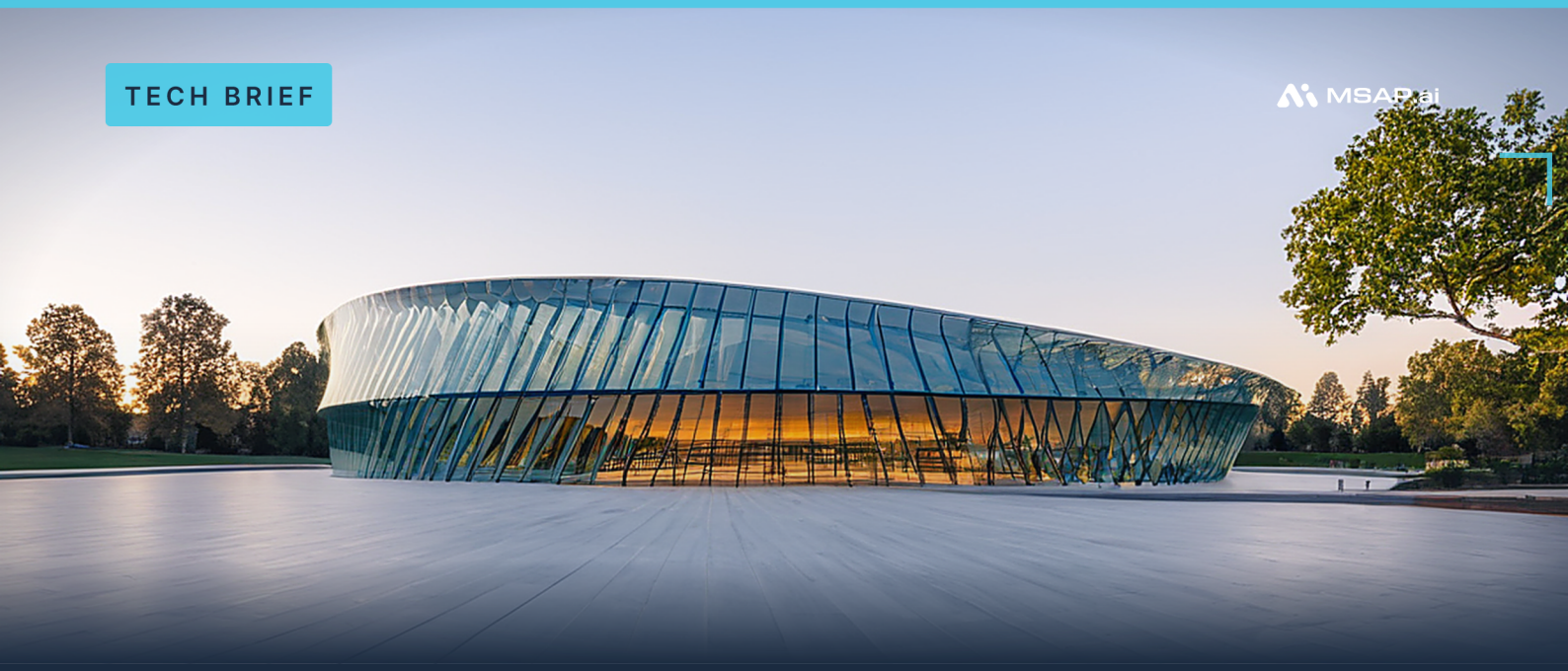
TECH BRIEF

MSAP.ai



AI Native Platform 이 필요한 이유

생성형 AI 의 활용 방식은 4년 만에

생성형 AI 의 활용 방식은 4년 만에 두 차례 크게 바뀌었습니다. 2022년 ChatGPT 가 만든 "프롬프트 시대" 는 단발성 자연어 질문으로 모델을 사용하는 단계였습니다. 사람이 한 문장을 입력하면 모델이 한 문장을 돌려주는 *질의 응답* 패턴이 표준이었습니다. 그러다가 2025년부터는 검색·문맥·메모리를 함께 묶어 모델에 *환경* 으로 제공하는 "컨텍스트 시대" 가 표준이 되었습니다.

2026-05-22 · 작성자: SMART CF Team · msap.ai

목차

AI Native Platform 이 필요한 이유 — 한국 IT 의사결정자를 위한 백서

본 백서의 1쪽 요약 (Executive Summary)

CNCF (Cloud Natvie Computing Foundation) 가 Kubernetes 를 AI 의 foundational infrastructure 로 공식 규정하고, GenAI 추론 운영 조직의 66%가 K8s 위에서 추론을 운영합니다 [S01, S02]. 그럼에도 엔터프라이즈의 86%가 파일럿 정체 에 머물러 있고, 2026년 엔터프라이즈 소프트웨어의 40%가 task-specific AI Agent 를 통합한다는 Gartner 전망과 본격 프로덕션 배포 2% 미만의 현실 사이에는 Agent 주변 인프라 의 격차가 존재합니다 [S08, S09, S13].

한국 공공기관과 기업이 개별 시스템 별로 그룹웨어 AI·홈페이지 AI·MIS AI·민원 챗봇·HR/복지 AI·콜센터 AI·법무 검토 AI·사내 위키 검색 AI 등 시스템별 AI 프로젝트 로 따로 발주하는 구조로는 이 격차를 메울 수 없습니다. DORA 2025 는 부실한 내부 플랫폼 위의 AI 도구 가 생산성을 오히려 떨어뜨림 을 정량적으로 보였고 [S07], 금융위원회는 「금융권 AI 플랫폼」 정책으로 오픈소스 AI 모델·데이터·PoC 인프라를 플랫폼 단위로 통합 제공 하는 방향을 공식화하여 공공·기업 전반에 동일한 정책 흐름이 확산되고 있습니다 [S26].

본 백서는 이 흐름을 AI Native Platform — K8s 위 13 컴포넌트로 묶인 표준 플랫폼 으로 정의하고, 4~5 계층 아키텍처, MCP 정식 표준 채택, 망분리 적합성, 13 컴포넌트 RFP 체크리스트를 의사결정 자산으로 제공합니다.

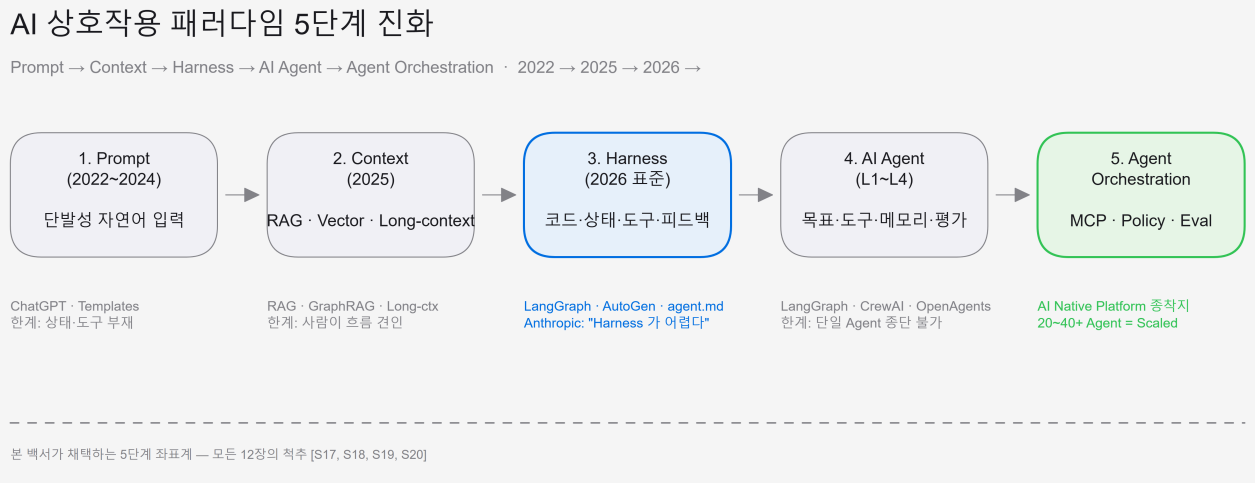
1장. AI 상호작용 패러다임의 5단계 진화

생성형 AI 의 활용 방식은 4년 만에 두 차례 크게 바뀌었습니다. 2022년 ChatGPT 가 만든 "프롬프트 시대" 는 단발성 자연어 질문으로 모델을 사용하는 단계였습니다. 사람이 한 문장을 입력하면 모델이 한 문장을 돌려주는 *질의 응답* 패턴이 표준이었습니다. 그러다가 2025년부터는 검색·문맥·메모리를 함께 묶어 모델에 *환경* 으로 제공하는 "컨텍스트 시대" 가 표준이 되었습니다. RAG(Retrieval Augmented Generation, 검색 증강 생성) 가 대표적인 사례입니다.

2026년 한국 공공기관과 기업의 AI TF 리더가 가장 자주 받는 질문은 더 이상 "어떤 프롬프트가 좋은가" 가 아닙니다. *우리 Agent 가 어떤 환경에서 살아야 하는가* 라는 *환경 설계* 의 질문입니다. 모델이 단발성 답을 돌려주는 것이 아니라 도구를 호출하고 상태를 기억하고 피드백을 받아 다음 행동을 결정하는 *행동 주체* 로 작동하기 시작했기 때문입니다.

본 장은 AI 상호작용을 다섯 단계로 정렬합니다. Prompt → Context → Harness → AI Agent → Agent Orchestration 의 5 단계입니다. 단계가 하나 진행될 때마다 *모델이 다루는 환경* 의 폭이 커지고, *조직이 책임지는 인프라* 의 깊이가 깊어집니다. 단계 정의가 분명해야 한국 IT 의사결정자가 자기 조직의 현재 위치를 진단할 수 있고, 다음 단계 투자 우선순위도 분명해집니다.

본 백서는 이 5 단계 좌표계를 출발점으로 삼습니다. 본 장의 핵심 표 ([FIGURE: paradigm-5stage-table]) 는 12 장 전체의 척추 역할을 합니다. 4 장 (AI Native Platform 의 4~5 계층 아키텍처) 과 5 장 (13 컴포넌트 매트릭스) 이 이 좌표계 위에 정렬됩니다. 본 장에서 분명히 한 정의는 이후 모든 장에서 재사용되며 일관성을 유지합니다.



캡션: AI 상호작용 패러다임 5단계 — 정의·대표 기술·한계·다음 단계 전이 이유

5 단계 정의·대표 도구·한계·다음 단계로 가는 이유를 행으로 정렬한 통합 표입니다. 첫 행은 Prompt Engineering 의 정의와 ChatGPT·Claude·Gemini 단발 호출, Prompt Templates 를 대표 도구로 명시하고, 한계로 상태·도구·메모리 부재와 결과 재현성 한계를 적습니다. 두 번째 행은 Context Engineering 으로 RAG·GraphRAG·Vector DB·Long-context model 을 대표 기술로 표기합니다. 세 번째 행은 Harness Engineering, 네 번째 행은 AI Agent (L1~L4 자율성 단계 포함), 다섯 번째 행은 Agent Orchestration 으로

MCP·Agent Fabric·Policy Engine·Eval Pipeline 결합을 표기합니다. 각 행의 *다음 단계로 가는 이유* 열은 한국 IT 의사결정자가 자기 조직 위치를 진단한 뒤 다음 단계 투자 근거로 활용 가능한 형식으로 작성됩니다.

1.1 Prompt → Context — 환경 묶음의 등장

1.1.1 Prompt Engineering (2022~2024) — 단발성 자연어의 시대

Prompt Engineering 은 모델에 자연어 한 줄 또는 짧은 템플릿을 입력해 결과를 얻는 단계입니다. ChatGPT 의 등장이 이 시대를 열었고, 한국에서도 2023년부터 2024년 사이에 대부분의 사내 GenAI 도입 시도가 이 단계에서 시작되었습니다. 하지만 이 단계의 본질적 한계는 분명했습니다. 모델이 *상태* 를 기억하지 못하고, *외부 도구* 를 호출하지 못하며, *결과가 재현되지 않는* 세 가지입니다.

도메인 컨텍스트를 매번 사람이 수동으로 입력해야 하는 부담은 특히 한국 공공·기업 환경에서 도입 정체의 주요 원인이 되었습니다. 동일한 보고서 형식을 다섯 번 만들면 다섯 번 같은 컨텍스트를 다시 적어 넣어야 하는 운영상의 비효율이 누적되었기 때문입니다. Prompt 단계의 우수 사례조차도 *동일 작업의 반복 자동화* 측면에서는 큰 한계를 보였습니다 [S17].

한국 RFP 가 여전히 "프롬프트 엔지니어링" 만 요구하는 경우가 있다면, 이는 4 년 전의 패러다임에 머무른 발주 형식입니다. 본 백서가 5 단계 좌표계를 먼저 제시하는 이유는, 현재 RFP 발주 단계가 *어떤 패러다임* 에 정렬되어 있는지를 의사결정자가 자가 진단할 수 있게 하기 위함입니다. Prompt 단계에 머무른 발주는 *시스템별 챗봇* 의 한계를 그대로 안게 됩니다 — 다음 단계로 이행하지 않으면 본격 프로덕션 진입이 사실상 불가능합니다 [S18].

1.1.2 Context Engineering (2025) — RAG·Vector·Long-context 환경 묶음

Context Engineering 단계에서는 검색·메모리·도구 출력을 *환경* 으로 묶어 모델에 함께 제공합니다. 가장 대표적인 사례가 RAG 입니다. 사내 문서를 임베딩으로 벡터화하여 Vector DB 에 저장하고, 사용자 질문에 대해 관련 문서를 검색해 모델에 함께 전달하는 패턴입니다. GraphRAG, Long-context 모델, 메모리 시스템도 이 단계의 변형으로 분류됩니다.

이 단계는 한국 공공기관과 기업의 챗봇 PoC (민원 응대·그룹웨어·홈페이지·HR·콜센터·사내 위키 검색 등) 가 가장 많이 머무는 단계이기도 합니다. RAG 구조 자체는 정착했지만, *여전히 사람이 흐름을 끌고 가는* 한계가 남습니다. 다단계 의사결정이나 외부 도구 호출이 필요한 워크플로우는 RAG 외부에서 별도 코드로 구현해야 했기 때문입니다 [S17]. 즉 모델은 *답* 을 잘 만들지만, *행동* 을 스스로 결정하지는 못하는 단계입니다.

본 단계의 핵심 가치는 *데이터와 모델의 분리* 입니다. 사내 지식을 모델 외부에 두고 검색으로 가져오는 구조는, 모델 교체 시에도 데이터를 재활용할 수 있는 자산화의 시작점이 됩니다. 본 백서 5 장에서 정의하는 13 컴포넌트 매트릭스의 Vector DB · Embedding · Knowledge Repo 영역이 바로 이 Context 단계의 자산화를 가능하게 하는 인프라입니다 [S18].

다음 단계로 이행하는 이유는 분명합니다 — 자동화된 다단계 의사결정과 도구 실행을 모델 안에서 처리할 필요가 생기기 때문입니다. 사람이 흐름을 매번 끌고 가는 구조로는 *20~40 개 이상의 Agent 를 동시 운영하는 Scaled 단계* 에 도달할 수 없습니다 (본 백서 7 장에서 상세 진단).

1.2 Harness → Agent — 모델이 사는 환경의 표준화

1.2.1 Harness Engineering — 코드·상태·도구·피드백 루프의 환경

Harness 라는 단어는 마구(馬具) 를 뜻하는 영단어에서 가져왔습니다. 말을 마차에 연결하는 *장비 일체* 처럼, Harness 는 모델을 *작업이 가능한 환경* 에 연결하는 *환경 일체* 를 가리킵니다. 구체적으로는 네 가지 요소로 구성됩니다 — *코드 (작업 흐름 정의), 상태 (현재 단계 추적), 도구 (외부 호출 인터페이스), 피드백 루프 (도구 출력의 모델 재주입)* 입니다. 이 네 요소가 결합된 환경 안에서 모델은 비로소 *Agent* 가 됩니다 [S17, S19].

Anthropic 의 엔지니어링 블로그는 이 단계의 본질을 한 줄로 정리했습니다. *"Harness 가 어려운 것이지 Agent 가 어려운 것이 아니다."* 이 결론은 한국 RFP 어휘에 그대로 적용 가능한 통찰입니다. 단일 Agent 의 모델 선택이나 프롬프트 최적화에 시간을 쓰기 전에, *모델이 사는 환경* 인 Harness 의 표준화가 먼저라는 의미입니다 [S19, S20].

잘못 설계한 Harness 는 오히려 모델 성능을 *떨어뜨립니다*. 도구 출력의 토큰이 폭증해 컨텍스트 한계를 초과하거나, 잘못된 종료 조건으로 무한 루프에 빠지거나, 부적절한 컨텍스트 압축으로 핵심 정보가 누락되는 사례가 보고됩니다. 즉 Harness 자체가 *공학적 제품* 으로 다뤄져야 하며, 사내 인프라 팀의 정식 운영 대상이 되어야 합니다 [S19].

본 백서 4 장의 *Execution Layer* 와 5 장의 *Agent Builder · MCP Server · LLM Gateway* 컴포넌트가 바로 이 Harness 의 구성 요소를 표준화한 영역입니다. LangGraph 의 상태 머신, AutoGen 의 Runtime, MCP 의 표준 도구 인터페이스 — 모두 Harness Engineering 단계의 표준화 시도로 분류됩니다 [S18, S22].

1.2.2 AI Agent — 목표·도구·메모리·평가의 자율 실행체

Agent 는 *목표*, *도구*, *메모리*, *평가 루프* 의 네 요소를 가진 자율 실행체입니다. 목표는 사람이 부여하는 작업의 정의이고, 도구는 Agent 가 호출 가능한 외부 인터페이스 (예: 데이터베이스 조회, 이메일 발송, API 호출) 이며, 메모리는 단기·장기 상태를 저장하는 영역, 평가 루프는 결과의 품질을 자동 점검하는 메커니즘입니다. 이 네 요소가 모두 갖추어진 시스템이 Agent 의 최소 정의입니다 [S18, S21].

Agent 의 자율성은 보통 L1 부터 L4 의 4 단계로 구분됩니다. L1 은 *반응형 Agent* — 사람이 지시한 한 단계를 수행하고 멈춥니다. L2 는 *순차적 Agent* — 미리 정의된 단계 흐름을 따라갑니다. L3 는 *적응형 Agent* — 상황에 따라 도구를 선택하고 흐름을 조정합니다. L4 는 *자율 생태계* — 여러 Agent 가 협업하며 사람의 개입 없이 작업을 완수합니다 [S13, S14].

한국 공공·기업 조직의 *생산성 Agent* (예: 보고서 자동 작성, 회의록 요약, 메일 분류) 는 대부분 L1~L2 단계에 머무릅니다. L3 이상으로 진입하려면 도구 카탈로그, 정책 엔진, 평가 자동화가 *플랫폼 수준* 으로 갖춰져야 하기 때문입니다. 즉 *단일 Agent 만 잘 만든다고 L3·L4 로 가지 않습니다* — Agent 의 *주변 인프라* 가 단계 진입의 전제 조건입니다 [S13].

단일 Agent 의 한계는 분명합니다. 다부서·다시스템에 걸친 종단(end-to-end) 업무 자동화는 한 Agent 가 혼자 처리할 수 없습니다. 거버넌스·관찰가능성·재사용은 더더욱 단일 Agent 의 범위를 넘어섭니다. 그래서 *Agent Orchestration* 이라는 5번째 단계가 자연스럽게 다음 의제로 등장합니다 [S15, S16, S21].

1.3 Agent Orchestration — 다 Agent 협업 런타임

1.3.1 Multi-Agent Orchestration 의 정의와 구성 요소

Agent Orchestration 은 여러 Agent·도구·인간을 *정책 하에* 라우팅·협업·감독하는 런타임을 가리킵니다. 단순히 두세 개의 Agent 를 동시에 실행하는 것이 아니라, *정책 엔진* 이 어떤 Agent 가 어떤 도구를 언제 호출할 수 있는지를 통제하고, *Agent Fabric* 이 Agent 의 라이프사이클을 관리하며, *Eval Pipeline* 이 결과 품질을 자동 평가하고, *MCP* 가 Agent 간·Agent 와 도구 간 표준 인터페이스를 제공하는 결합 시스템입니다 [S09, S10, S24].

이 단계는 *AI Native Platform 의 종착지* 입니다. 본 백서 4 장에서 다루는 Ampcome 4 계층 모델과 Digital Applied 5 계층 모델 모두 이 Multi-Agent Orchestration 을 *가로축* 으로 정의합니다. 즉 Orchestration 은 별도의 계층이 아니라, 모든 계층을 가로지르는 *런타임 환경* 입니다 [S09, S10, S11].

한국 공공·기업 의사결정자가 RFP 에 *Multi-Agent Orchestration* 을 필수 표준 요구 항목으로 명시하는 흐름이 2026년부터 본격화되었습니다. KOFIA 의 LLM ETF 챗봇 RFP 가 단일 챗봇 RFP 임에도 *LLM·VectorDB·검색·평가·운영* 을 묶음으로 요구한 것이 그 신호입니다 (본 백서 10 장에서 상세 분석) [S28]. 즉 시스템별 단발성 발주가 사실상 *플랫폼급 Orchestration* 사양을 내포하는 형태로 진화 중입니다.

본 백서가 *AI Native Platform 의 13 컴포넌트 매트릭스* 를 RFP 항목 형식으로 정렬한 이유도 여기 있습니다. Orchestration 의 표준 구성 요소를 점검표 형식으로 제공해야, 한국 의사결정자가 *어느 컴포넌트가 누락되어 있는가* 를 한눈에 진단할 수 있기 때문입니다.

1.3.2 5 단계 통합 표 — 본 백서가 채택하는 좌표계

본 백서가 채택하는 5 단계 좌표계 (Prompt → Context → Harness → AI Agent → Agent Orchestration) 는 단순한 진화 순서 정렬이 아닙니다. 각 단계가 *모델이 책임지는 범위, 환경이 책임지는 범위, 조직이 책임지는 인프라* 의 세 축에서 명확히 달라지는 *경계 정의* 입니다. 의사결정자는 자기 조직의 위치를 이 좌표계 위에 직접 표시할 수 있어야 합니다 [S17, S18, S19].

각 단계의 *모델 책임 범위* 는 다음과 같이 정리됩니다. Prompt 는 *답 생성*, Context 는 *답 생성 + 검색 결과 통합*, Harness 는 *답 생성 + 도구 호출 + 상태 추적*, AI Agent 는 *목표 분해 + 도구 선택 + 평가 자기 점검*, Agent Orchestration 은 *Agent 라우팅 + 정책 시행 + 협업 조정* 입니다. 단계가 진행되면 모델이 책임지는 영역이 넓어지면서 동시에 *환경의 정교함* 도 따라 올라가야 합니다.

조직이 책임지는 인프라는 단계마다 다음과 같이 누적됩니다. Prompt 는 *모델 API* 만 있으면 됩니다. Context 는 *Vector DB + 임베딩 + 검색 파이프라인* 이 추가됩니다. Harness 는 *상태 저장소 + 도구 카탈로그 + 피드백 루프 코드* 가 더해집니다. AI Agent 는 *Agent Runtime + 메모리 + 평가 파이프라인* 이 결합됩니다. Agent Orchestration 은 *정책 엔진 + Agent Fabric + Tool Registry + MCP + Observability* 가 모두 갖춰져야 합니다.

본 5 단계 표 ([FIGURE: paradigm-5stage-table]) 는 본 백서 전체 좌표계의 척추입니다. 의사결정 보고서에 발췌해 사용해도 좋습니다 — 임원이 *우리 조직의 현재 단계와 다음 단계의 인프라 격차* 를 한 페이지로 파악할 수 있도록 정렬된 형식입니다. 다음 2 장에서는 이 5 단계 모두가 *동일한 기반* 위에서 돌아간다는 사실을 다룹니다 — 그 기반이 바로 Kubernetes 입니다 [S01, S02].

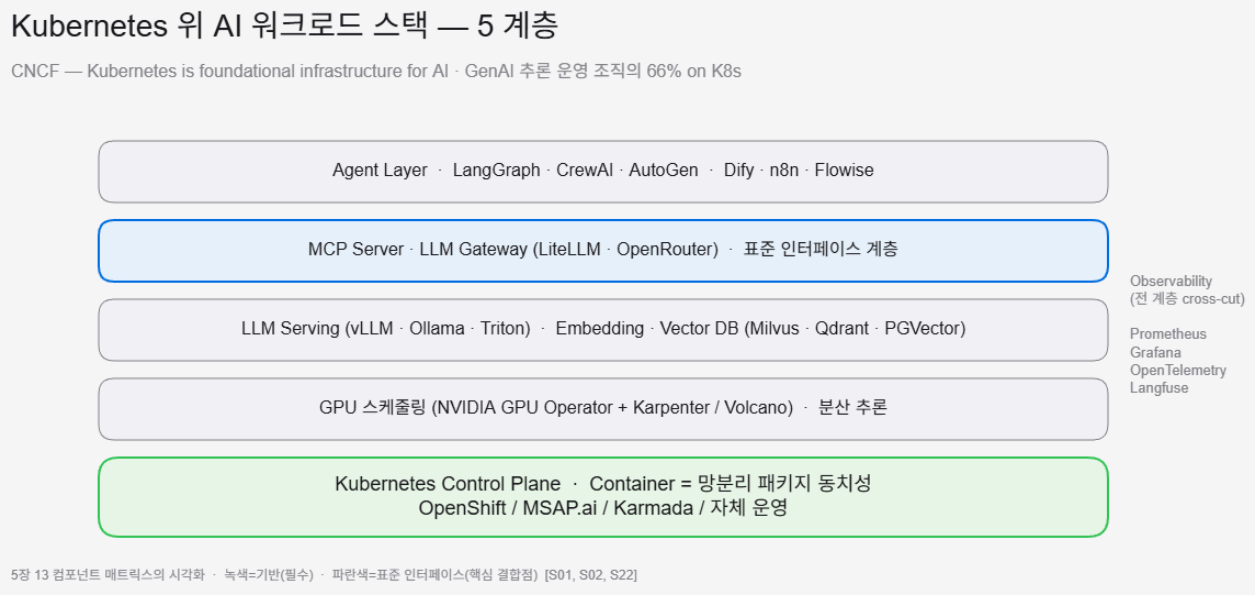
2장. Kubernetes 는 AI 의 OS 입니다

CNCF (Cloud Native Computing Foundation) 의 한 줄 메시지가 2026년 엔터프라이즈 AI 아키텍처 합의를 바꿨습니다. *"Kubernetes is foundational infrastructure for AI."* 이 선언은 단순한 마케팅 슬로건이 아닙니다. CNCF Annual Survey 2025 와 KubeCon EU 2026 키노트가 동일한 메시지를 반복하며, 정량 데이터로 뒷받침된 산업 합의입니다 [S02].

본 장은 이 합의를 한국 IT 의사결정자의 좌표로 옮겨 적습니다. *Cloud Native 까지는 우리도 했다. AI Native 는 그 위에 한 계층 더 쌓는 일이다* — 이 mental model 이 본 장의 핵심 결론입니다. 이미 K8s 자산을 가진 한국 공공·기업 조직은 매몰 비용이 아니라 *AI Native 의 출발점* 을 가진 셈입니다 [S01].

KubeCon EU 2026 발표가 제시한 두 정량 데이터를 먼저 확인합니다. 컨테이너 사용자의 82%가 Kubernetes 를 프로덕션에서 운영하고 있고, 그중 GenAI 추론을 운영하는 조직의 66%가 K8s 위에서 추론 워크로드를 운영합니다. 즉 *추론 운영 조직의 3분의 2 가 이미 K8s 위에 있는* 정량 현실입니다 [S01].

본 장의 핵심 시각 자산은 K8s 위 AI 워크로드 스택 다이어그램 ([FIGURE: k8s-ai-stack]) 입니다. GPU 스케줄링, LLM Serving, Vector DB, MCP Server, Observability 의 5 계층이 K8s 표준 운영 모델 위에서 어떻게 결합되는지를 시각화한 다이어그램입니다.



캡션: K8s 위 AI 워크로드 스택 — GPU 스케줄링·LLM Serving·Vector·MCP·Observability 의 5계층

K8s control plane 위에 GPU node pool 이 그려지고, GPU 스케줄링 계층 (NVIDIA GPU Operator + Karpenter) 이 그 위에 배치됩니다. 그 위에 LLM Serving 계층 (vLLM·Ollama·Triton·TGI 의 Helm Chart 또는 Operator) 과 Embedding · Vector DB 계층 (Milvus·Qdrant·Weaviate Operator) 이 병렬로 그려집니다. 더 위에는 MCP Server 풀과 LLM Gateway (LiteLLM·OpenRouter) 가 배치되고, 가장 윗 계층에 Agent Builder (LangGraph·CrewAI·AutoGen 의 컨테이너) 와 Visual Workflow (Flowise·n8n) 가 배치됩니다. 측면에는 Observability 스택 (Prometheus·Grafana·OpenTelemetry·Langfuse) 이 모든 계층의 메트릭·로그·트레이스를 수집하는 형태로 그려집니다. 다이어그램의 가장 아래에는 *컨테이너 = 망분리 패키지* 라는 부속 설명 박스가 한국 공공·기업 적합성을 명시합니다.

2.1 CNCF 공식 메시지의 함의

2.1.1 *Kubernetes is foundational infrastructure for AI* 선언의 위치

CNCF 가 공식적으로 K8s 를 *AI 의 foundational infrastructure* 로 규정한 것은 한국 의사결정자에게 두 가지 의미를 갖습니다. 첫째, *Cloud Native ↔ AI Native* 의 어휘 연속선이 확립되었다는 점입니다. 한국 공공·기업 시장에서 *Cloud Native* 는 이미 어느 정도 인지도가 있는 용어이며, 행정안전부의 클라우드 네이티브 가이드도 정착 단계입니다. *AI Native* 라는 신조어는 이 기존 인지의 직관적 확장으로 받아들여집니다 [S02].

둘째, CNCF 의 메시지는 K8s 가 AI 워크로드의 *대체 가능한 옵션 중 하나* 가 아니라 *기반 인프라 표준* 임을 못 박는 효과를 가집니다. 즉 AI 도입 시 K8s 가 아닌 다른 컨테이너 오케스트레이션을 채택하는 것은 *표준에서 이탈하는 의사결정* 이 됩니다. KubeCon EU 2026 의 키노트 발표가 이 메시지를 *Cloud Native + AI* 의 합집합 생태계로 명확히 정렬했습니다 [S01, S02].

본 백서는 이 선언을 *AI Native Platform 의 출발점* 으로 채택합니다. 5 장에서 정의하는 13 컴포넌트 매트릭스의 *1 번 영역 = Kubernetes* 가 바로 이 선언의 의사결정 자산화입니다. K8s 영역이 13 컴포넌트의 *맨 앞* 에 배치된 이유는, 이 영역의 부재가 다른 12 영역 전체의 일관된 운영을 불가능하게 만들기 때문입니다 [S02].

2.1.2 한국 의사결정자의 K8s 도입 누적 — Cloud Native 위 AI Native

한국 공공·기업·대기업의 K8s 도입은 2020년 전후부터 본격화되었습니다. OpenShift, MSAP.ai, 자체 운영 K8s 가 주요 선택지였고, 2024년 무렵에는 망분리 환경에서도 K8s 운영이 정착 단계로 진입했습니다. 행정안전부의 클라우드 네이티브 가이드와 금융감독원의 클라우드 가이드라인이 이 흐름을 정책 측면에서 뒷받침했습니다 [S03].

기존 K8s 자산은 AI 워크로드의 *출발점* 입니다. 매몰 비용으로 보지 않고 *AI Native 의 1 계층* 으로 재정의하면, 이미 가진 운영 인력·모니터링·CI/CD 가 그대로 활용 가능한 자산이 됩니다. 본 백서가 *AI Native Platform* 이라는 용어를 채택한 이유 중 하나는, 이 *기존 자산의 재활용* 메시지를 한국 의사결정자에게 직관적으로 전달하기 위함입니다 [S01, S04].

한국 K8s 도입률은 글로벌 평균과 유사하거나 일부 영역에서는 더 높은 수준으로 알려져 있습니다. 다만 *GenAI 추론 워크로드를 K8s 위에서 운영하는 비율* 은 글로벌 평균 66% 보다 낮을 가능성이 큽니다. 한국에서는 GenAI 추론을 *별도 서버* 또는 *외부 API* 형태로 운영하는 사례가 여전히 다수이기 때문입니다. 이 격차는 *AI Native Platform* 도입의 잠재 시장이 큽니다 [S01, S03].

본 장 이후 4 장 (4~5 계층 아키텍처) 과 5 장 (13 컴포넌트 매트릭스) 에서 이 K8s 기반 위에 *어떤 컴포넌트를 어떤 순서로 올릴 것인가* 를 정렬합니다. 본 장은 그 기반의 *정합성과 정책적 안정성* 을 확인하는 단계입니다.

2.2 정량 데이터 — 컨테이너 82% / GenAI 추론 66% on K8s

2.2.1 KubeCon EU 2026 정량 — 82% / 66% 의 출처와 해석

KubeCon EU 2026 에서 발표된 정량 데이터의 두 핵심 숫자는 다음과 같습니다. 82% — 컨테이너 사용자 중 *프로덕션에서 K8s 를 운영* 하는 비율. 66% — GenAI 추론 워크로드를 운영하는 조직 중 *K8s 위에서 추론을*

운영 하는 비율. 두 숫자는 모두 CNCF 의 산업 조사 데이터를 기반으로 한 KubeCon EU 2026 키노트 발표에서 인용되었습니다 [S01].

82% 라는 컨테이너 사용자 K8s 도입률은 *컨테이너 = K8s* 가 사실상 1:1 관계임을 보여줍니다. 컨테이너를 쓰면서 K8s 를 안 쓰는 18% 는 대부분 *Docker Compose 단독 운영의 소규모 사례* 이거나 *전환 중 인 과도기 상태* 로 분류됩니다. 한국 공공·기업처럼 *프로덕션 운영 안정성* 이 핵심 요건인 환경에서는 사실상 K8s 가 유일한 선택지입니다 [S01].

66% 라는 GenAI 추론 K8s 운영률은 더 의미 있는 숫자입니다. GenAI 가 본격 도입되기 시작한 시점이 2023 년 무렵임을 고려하면, 3년 만에 추론 운영 조직의 3 분의 2가 K8s 위로 이동한 것은 *놀라운 표준화 속도* 입니다. 이 속도는 vLLM·Ollama·Triton 등 LLM Serving 도구가 K8s Operator 또는 Helm Chart 형태로 표준 배포되는 흐름이 만들어 낸 결과입니다 [S01, S22].

본 두 숫자는 의사결정 보고서에 *글로벌 정량 한 줄 인용* 으로 활용 가능합니다. 예를 들어 *"GenAI 추론 운영 조직의 66%가 K8s 위에 있습니다 — 한국도 동일한 표준에 합류해야 합니다"* 라는 한 줄 결론은, 임원 보고용으로 충분히 강력합니다 [S01, S02].

2.2.2 GPU 스케줄링·LLM Serving·Distributed Inference 의 K8s 표준 패턴

K8s 위 AI 워크로드의 표준 패턴은 다음과 같이 정리됩니다. GPU 스케줄링은 NVIDIA GPU Operator 가 표준이고, GPU 자원의 multi-tenant 관리는 Karpenter 또는 Volcano 가 표준입니다. LLM Serving 은 vLLM 의 KV 캐시 최적화와 PagedAttention 이 사실상 표준이 되었고, Ollama 는 경량 배포의 표준, Triton 은 NVIDIA 의 공식 추론 서버 표준입니다 [S22].

분산 추론 (Distributed Inference) 의 표준화도 빠르게 진행되었습니다. K8s 의 StatefulSet 과 Headless Service 를 활용한 *모델 샤딩* 패턴, vLLM 의 *PagedAttention 분산 구조*, NVIDIA Dynamo 의 *추론 라우팅* 이 표준 패턴으로 자리잡았습니다. 한국 망분리 환경에서도 이 표준 패턴이 그대로 적용 가능합니다 — 모든 컴포넌트가 컨테이너로 패키지화되어 있기 때문입니다 [S01, S22].

본 백서 5 장의 13 컴포넌트 매트릭스에서 이 K8s 표준 패턴이 그대로 *RFP 항목* 으로 변환됩니다. 예를 들어 *"GPU 스케줄링 : NVIDIA GPU Operator + Karpenter / Volcano 중 1 종 이상 채택"* 같은 형식으로 의사결정 자산화가 가능합니다. K8s 위에서의 표준 패턴이 분명할수록 RFP 응답의 비교 가능성도 분명해집니다.

K8s Operator 모델의 또 다른 강점은 *자동 운영* 입니다. Helm Chart 로 배포한 LLM Serving 컴포넌트는 K8s 의 자동 복구, 자동 확장, 자동 업그레이드 메커니즘을 그대로 사용합니다. 즉 *AI 워크로드 운영 인력* 이 기존 K8s 운영 인력으로 흡수 가능하며, 이는 한국 IT 인력난 환경에서 핵심 가치입니다 [S22].

2.3 한국 망분리·온프레미스 환경에서의 K8s = AI OS

2.3.1 컨테이너 = 망분리 패키지 동치성

한국 공공·기업의 *망분리* 요건은 우회할 수 없는 전제입니다. 그리고 망분리 환경에서 *AI 워크로드 운영* 의 단일 합리적 선택지가 *컨테이너 + K8s* 입니다. 그 이유는 컨테이너 이미지가 본질적으로 *오프라인 패키지* 의 성격을 가지기 때문입니다. Container Registry 에 한 번 적재한 이미지는 외부 네트워크 없이 모든 의존성을 포함한 채로 배포 가능합니다 [S01].

이 *컨테이너 = 망분리 패키지 동치성* 은 한국 의사결정자에게 *단순한 mental model* 을 제공합니다. 망분리 환경에서 AI 를 운영하려면 *내부 Container Registry + 자체 K8s + 컨테이너로 패키징된 모든 AI 컴포넌트* 의 세 가지가 필요합니다. 이 모델은 추가 설명 없이도 한국 IT 운영 팀이 즉시 이해할 수 있는 구조입니다.

vLLM, Milvus, Qdrant, MCP Server 등 모든 AI Native 컴포넌트가 *공식 컨테이너 이미지를 제공* 한다는 사실이 이 모델을 가능하게 만듭니다. 즉 *오픈소스 AI Native Platform 의 모든 13 컴포넌트가 컨테이너 = 망분리 패키지* 의 동치성을 만족합니다. 본 백서 11 장의 망분리 적합성 점검 매트릭스가 이 동치성을 13 컴포넌트별로 정렬합니다 [S22, S23].

본 동치성의 또 다른 가치는 *벤더 락인 회피* 입니다. 컨테이너 이미지는 표준 형식이므로 *어떤 K8s 위에서도 동일하게 동작* 합니다. 즉 OpenShift 위에서 운영하다가 자체 K8s 로 이전하더라도, 컨테이너 이미지 자체는 변경할 필요가 없습니다. 이는 *모델 교체 비용 절감* 과 함께 본 백서 6 장의 6 축 비교에서 한국 의사결정자에게 강력한 근거가 됩니다.

2.3.2 온프레미스 K8s + AI 운영 플랫폼 후보 — OpenShift AI · MSAP.ai · 자체 운영

한국 공공·기업 환경에서 *온프레미스 K8s + AI 운영 플랫폼* 의 주요 후보는 다음 세 가지입니다. 첫째, *Red Hat OpenShift / OpenShift AI* — 엔터프라이즈 K8s 와 그 위의 AI 운영 플랫폼이 결합된 Red Hat 상용 제품으로, 한국 공공·기업의 도입 사례가 누적되어 있습니다. 둘째, *MSAP.ai* — 오픈마루와 투라인클라우드가 공동 개발한 *K8s 위 완전한 AIaaS 플랫폼* 으로, LLM Serving·Vector DB·Embedding·Agent Builder·MCP Server·LLM Gateway·Observability·Governance 의 13 컴포넌트를 사전 통합하여 공공기관과 기업의 망분리 환경에 적합한 형태로 제공합니다. 셋째, *자체 운영 K8s* — kubeadm 으로 직접 구축한 표준 K8s 위에 AI Native 컴포넌트를 직접 조립하는 방식입니다 [S01, S03].

각 후보의 *운영 인력 요건과 도입 특성* 은 다음과 같이 정리됩니다. OpenShift / OpenShift AI 는 Red Hat 의 공식 지원으로 운영 인력 부담이 낮지만 Red Hat 상용 구독 비용이 발생합니다. MSAP.ai 는 K8s 위에 AIaaS 13 컴포넌트를 사전 통합한 형태로 제공되어 *AI 운영을 위한 별도 조립·통합 비용이 거의 0* 에 가깝고, 국내 기술 지원과 한국어 LLM 통합·망분리 환경 사전 검증·RFP 응답 표준화가 기본 옵션으로 포함됩니다. 자체 운영 K8s 는 라이선스 자유도가 가장 높지만 *AI Native 13 컴포넌트의 조립·통합·운영 인력* 부담이 가장 큽니다.

한국 공공·기업의 의사결정 시 흔한 패턴은 *OpenShift AI / MSAP.ai (AI 운영 핵심) + 자체 운영 K8s (부속 워크로드) 하이브리드* 입니다. AI 운영의 핵심 워크로드는 OpenShift AI 또는 MSAP.ai 의 통합 패키지를 활용하고, 부속 시스템은 자체 K8s 위에 가벼운 컴포넌트를 추가하는 형태입니다. 이 패턴은 *모델 교체 비용 절감* 과 *벤더 락인 회피* 를 동시에 달성하는 합리적 선택지로 평가됩니다 [S01].

본 3 후보의 의사결정은 *AI 워크로드 적합성* 한 축으로만 결정되지 않습니다. 기존 K8s 운영 인력의 숙련도, 기존 라이선스 계약, 망분리 환경의 보안 인증 요건, 데이터센터 자원 가용성 등 여러 축이 결합됩니다. 본 백서는 3 후보의 RFP 형식 비교를 부록 A 의 13 컴포넌트 체크리스트의 *1 번 영역* 으로 정렬합니다. 의사결정 보고서에 그대로 첨부할 수 있는 형식으로 설계되었습니다.

다음 3 장에서는 이 K8s 기반 위에서도 *86% 가 파일럿에 머무르는 격차* 의 정체를 분해합니다. 격차는 *기반 인프라* 가 아니라 *기반 위의 인프라 통합* 에서 발생합니다 [S08, S13].

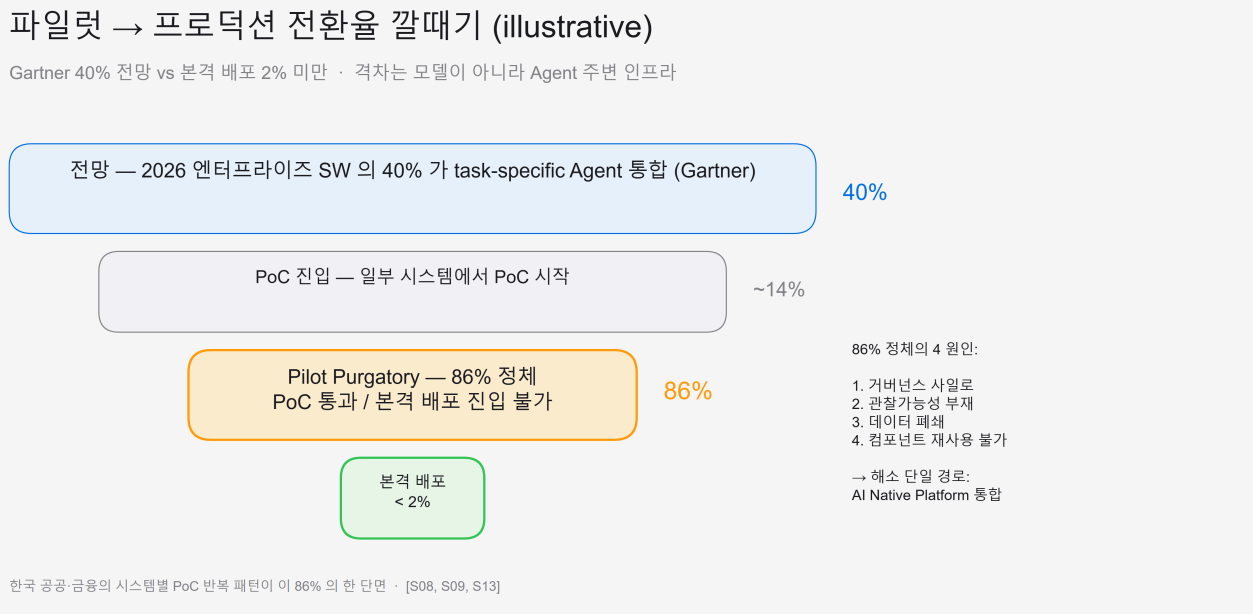
3장. 86% 가 파일럿에 머무릅니다 — 격차의 정체

Gartner 는 2026년 말까지 엔터프라이즈 소프트웨어 애플리케이션의 40%가 task-specific AI Agent 를 통합한다고 전망했습니다 [S08]. 그러나 같은 시점에 본격 프로덕션 규모로 배포된 비율은 2% 미만입니다 [S13]. 한국의 한 공공기관 AI TF 가 1년째 *민원 챗봇 PoC* 만 반복하고 있거나, 한 대기업 IT 부서가 *그룹웨어 챗봇·HR 챗봇·콜센터 AI PoC* 를 시스템별로 따로 진행하고 있다면, 이는 글로벌 86% 의 한 단면입니다.

본 장의 핵심 질문은 다음과 같습니다. *전망과 현실의 격차는 모델 성능이 만든 것인가, 아니면 Agent 주변 인프라가 만든 것인가?* 본 백서는 이 격차의 정체가 *후자* 임을 4 가지 구조적 원인으로 분해합니다. 거버넌스, 관찰 가능성, 데이터, 컴포넌트 재사용의 4 원인은 단일 시스템 발주로는 *동시에* 해소할 수 없는 항목들입니다.

본 장의 결론은 한국 IT 의사결정자에게 *위안과 동기를 동시에* 제공합니다. 위안은 *우리만 정체된 것이 아니다 — 글로벌 평균도 동일하게 86%* 라는 사실이고, 동기는 *격차 메우기 자체가 경쟁 우위* 라는 점입니다. 본 백서가 5 장 (13 컴포넌트 매트릭스) 과 6 장 (6 축 비교) 에서 제시하는 *AI Native Platform* 이 이 격차의 단일 경로 해소책입니다.

본 장의 시각 자산은 파일럿 → 프로덕션 전환율 깔때기 ([FIGURE: pilot-purgatory-funnel]) 입니다. 시장 전망의 상부 너비와 실측 본격 배포의 하부 너비의 격차를 한눈에 보여주는 형식입니다.



캡션: 파일럿 → 프로덕션 전환율 깔때기 (illustrative)

깔때기의 상부에 *Gartner 전망 40%* 가 넓게 표기됩니다. 다음 단계로 *PoC 진입 14%* — 시장 전망의 35% 가 PoC 까지 진입했음을 표시. 그 다음 *Pilot Purgatory 86% 정체* — 86% 가 PoC 단계에서 본격 배포로 진입하지 못함. 가장 아래 *본격 배포 2% 미만* 의 좁은 출구가 표시됩니다. 깔때기 옆 측면에는 4 가지 정체 원인 (거버넌스 사일로 / 관찰가능성 부재 / 데이터 폐쇄 / 컴포넌트 재사용 불가) 이 각 정체 지점에서 끌어내는 영향선으로 표시됩니다. 다이어그램의 하부에는 *격차 해소의 단일 경로 = AI Native Platform 통합* 이라는 부속 결론이 표기됩니다.

3.1 Gartner 40% vs 본격 배포 2% — 정량의 모순

3.1.1 Gartner 40% 전망의 출처와 해석

Gartner 의 Strategic Trends in Platform Engineering 보고서는 *2026년 말까지 엔터프라이즈 소프트웨어 애플리케이션의 40%가 task-specific AI Agent 를 통합* 한다고 전망했습니다. 2025년 5% 미만 수준에서 1년 만에 8배로 점프하는 전망입니다 [S08]. 이 숫자는 산업 전반의 *AI Agent 도입 의지* 가 강력함을 보여줍니다.

같은 보고서는 *Platform Engineering* 의 의무화도 함께 강조합니다. AI Agent 도입이 폭증하는 시점에 *내부 플랫폼* 이 부실하면, AI 도구는 오히려 운영 부담을 가중시킨다는 경고입니다. 즉 Gartner 의 40% 전망 자체가 *플랫폼 엔지니어링 성숙도* 를 전제 조건으로 한 숫자입니다 [S05, S08].

한국 시장에서도 이 전망은 그대로 적용됩니다. 한국 공공·기업·대기업의 AI Agent 도입 의지는 글로벌 평균보다 낮지 않습니다. 금융위 「금융권 AI 플랫폼」 정책, KOFIA 의 LLM 챗봇 RFP, 자본시장연구원의 통합 관리 권고 — 모두 *플랫폼 단위 AI 도입* 의 강한 의지를 보여줍니다 [S26, S27, S28]. 다만 의지가 본격 배포로 전환되는 비율은 글로벌 평균과 유사하게 낮을 가능성이 큽니다.

본 백서가 *왜 지금* 인가의 질문에 답하는 가장 직접적인 근거가 이 Gartner 전망입니다. *2026년 말까지 1년 안에 40%가 task-specific Agent 를 통합한다면, 한국 의사결정자는 그 시점까지 자기 조직의 위치를 어디에 둘 것인가* 라는 의사결정 압박이 작동하기 때문입니다. 본격 배포 진입 여부가 향후 3 년의 경쟁 우위를 가릅니다.

3.1.2 2% 미만 본격 배포 — Pilot Purgatory 의 정량

AgentMarketCap 의 2026 보고서는 *86%가 pilot purgatory 에 갇혀 있고 본격 프로덕션 배포는 2% 미만* 이라는 정량 데이터를 제시합니다 [S13]. *Pilot Purgatory* 라는 표현은 산업 합의된 용어입니다 — PoC 단계는 통과하지만 본격 배포로 진입하지 못한 채 *정체 상태* 에 머무는 조직들의 누적을 가리킵니다.

이 86% 정체의 본질을 분해하면 다음과 같습니다. *PoC 자체는 모델 API 호출 + 간단한 RAG 만 있으면 가능* 합니다. 그러나 본격 프로덕션 배포로 진입하려면 *거버넌스 정책, 감사 로그, 평가 자동화, 다부서 통합, 망분리 적합성, 모델 교체 대응, 운영 인력 확보* 등 *PoC 가 다루지 않는 7 가지 영역* 이 동시에 갖춰져야 합니다 [S09, S13].

한국 공공·기업에서 이 7 영역의 갖추기는 *시스템별 발주* 형태로는 사실상 불가능합니다. 그룹웨어 챗봇·홈페이지 챗봇·민원 챗봇·HR 챗봇·콜센터 AI·법무 검토 AI·사내 위키 검색 AI 같은 시스템별 발주 1 건에 거버넌스, 감사, 평가, 운영 인력의 7 영역을 모두 넣을 수 없기 때문입니다. 시스템별로 일부 영역만 채우면, 각 시스템이 자기 거버넌스와 자기 평가를 따로 운영하는 *사일로* 가 누적됩니다 [S05, S07].

이 정량의 함의는 분명합니다. *모델 성능을 더 높이는 것* 으로는 본격 배포 진입이 가능하지 않습니다. 모델 자체는 이미 충분히 좋습니다 — GPT-4 급 모델이 오픈소스로 제공되고 한국어 LLM 도 빠르게 발전했습니다. 격차를 만드는 것은 *모델 주변* 의 인프라이며, 이 인프라의 통합 단위가 *AI Native Platform* 입니다 [S13, S22].

3.2 격차의 4 원인 — 거버넌스 / 관찰가능성 / 데이터 / 재사용

3.2.1 거버넌스 사일로 + 관찰가능성 부재

첫 번째 원인은 *거버넌스 사일로* 입니다. 시스템별 AI 발주는 시스템마다 별도의 정책·RBAC·감사 로그를 운영하게 만듭니다. 한국 공공·기업의 *내부 감사 응대* 시각에서 보면, 시스템마다 다른 감사 로그 형식을 통합 분석하는 부담이 누적됩니다. 감사 응대가 *N 번* 으로 늘어나는 구조가 됩니다 [S09, S25].

거버넌스 사일로의 또 다른 폐해는 *정책 시행의 불일치* 입니다. 한 시스템에서는 LLM 호출 시 PII (개인식별정보) 마스킹을 강제하지만, 다른 시스템에서는 이 마스킹이 누락되는 상황이 빈번하게 발생합니다. 사용자가 어느 시스템을 쓰는지에 따라 *동일한 데이터의 다른 보호 수준* 이 적용되는 비일관성이 누적됩니다 [S25].

두 번째 원인은 *관찰가능성 (Observability) 부재* 입니다. LLM 호출의 추적·비용 분석·품질 측정은 *기존 APM 도구만으로는 부족* 합니다. Langfuse, Helicone, Phoenix 같은 *LLM 전용 Observability* 도구가 필요한데, 시스템별 발주 형태에서는 이 도구의 통합 운영이 어렵습니다 [S22, S25].

LLM 호출 추적의 부재는 본격 배포의 *직접적 장애물* 입니다. 한국 공공·기업의 *AI 의사결정 감사 요건* (공공기관 정보보호 인증, 금융권 내부 통제, 일반 기업의 컴플라이언스) 은 LLM 호출 시점, 입력, 출력, 모델 버전, 호출 비용, 호출자 ID 모두를 로그에 남기는 것을 요구합니다. 이 로그가 시스템마다 다른 형식으로 분산되면 감사 응대 자체가 *불가능에 가까운* 부담이 됩니다 [S25].

거버넌스와 관찰가능성의 두 원인이 결합하면 *PoC 통과는 가능하지만 본격 배포는 불가능* 한 정체 상태가 만들어집니다. PoC 단계에서는 감사·로그 요건을 *미루어 두고* 모델 동작만 검증하기 때문입니다. 본격 배포 시점에 이 두 영역의 부재가 일제히 드러납니다. 이것이 86% 정체의 핵심 메커니즘입니다 [S09, S13].

3.2.2 데이터 폐쇄 + 컴포넌트 재사용 불가

세 번째 원인은 *데이터 폐쇄* 입니다. 시스템별 AI 사업은 각 시스템 내부에 데이터를 폐쇄적으로 가둡니다. 그룹웨어 AI 가 자기 Vector DB 를 가지고, 홈페이지 AI 가 또 다른 Vector DB 를 가지며, MIS AI·민원 AI·HR AI·콜센터 AI·법무 검토 AI·사내 위키 검색 AI 가 각자 별도 Vector DB 를 운영하는 식입니다. 즉 사내 지식 자산이 *통합 활용 불가능* 한 형태로 누적됩니다 [S22, S23].

데이터 폐쇄의 폐해는 *재 PoC 비용 누적* 입니다. 새 Agent 를 만들 때마다 임베딩, 인덱싱, 정합성 검증을 다시 수행해야 하기 때문입니다. 1 차 Agent 의 데이터 정비에 3 개월이 걸렸다면, 2 차 Agent 에서도 동일한 3 개월이 다시 들고, N 차 Agent 까지 동일한 부담이 반복됩니다. *데이터 자산화의 부재* 가 도입 정체의 주요 원인입니다 [S22].

네 번째 원인은 *컴포넌트 재사용 불가* 입니다. 시스템별 AI 사업은 동일한 LLM Serving, 동일한 Vector DB, 동일한 MCP Server, 동일한 Observability, 동일한 Governance 컴포넌트를 *시스템마다 별도 구매* 하게 만듭니다. 즉 동일 컴포넌트의 N 회 중복 구매가 누적됩니다 [S05, S07, S09].

컴포넌트 재사용 불가의 함의는 *TCO 증가* 와 *벤더 락인 누적* 두 가지입니다. 시스템마다 다른 벤더 컴포넌트를 채택하면 *모델 교체 비용* 이 시스템 수만큼 늘어나고, *조직 차원의 협상력* 도 시스템별로 분산됩니다. 한국 공공·기업의 RFP 응답 비교 단계에서 *컴포넌트 표준화* 가 점점 더 중요한 평가 항목이 되고 있는 이유가 여기 있습니다 [S04, S22].

이 4 원인 (거버넌스 사일로, 관찰가능성 부재, 데이터 폐쇄, 컴포넌트 재사용 불가) 은 *서로를 강화* 합니다. 거버넌스가 사일로화되면 데이터 통합도 어려워지고, 데이터가 폐쇄되면 컴포넌트 재사용도 막힙니다. 즉 4 원인 모

두를 *동시에* 해소하지 않으면 본격 배포 진입이 불가능합니다. 이 *동시 해소* 의 유일한 단위가 *플랫폼* 입니다 [S09, S13].

3.3 한국 공공·기업의 도입 정체 패턴

3.3.1 시스템별 PoC 반복 패턴 — 동일 LLM·Vector·MCP 의 중복 발주

한국 공공·기업에서 관찰되는 *PoC 반복* 패턴의 5 가지 특성은 다음과 같이 정리됩니다. 첫째, 시스템마다 *별도 RFP* 로 발주됩니다. 그룹웨어 AI 챗봇, 홈페이지 AI 챗봇, MIS AI 챗봇, HR AI 챗봇, 민원 응대 AI, 콜센터 AI, 법무 검토 AI, 사내 위키 검색 AI 가 각각 다른 시점에 다른 부서가 발주합니다. 둘째, 각 RFP 는 *PoC 단위* 로 계약되어 본격 배포 진입의 책임이 RFP 외부에 있습니다 [S13, S26].

셋째, 각 PoC 는 *모델 API + RAG 구조* 의 동일 패턴을 반복합니다. 즉 *기술 스택의 70%가 중복* 됩니다. 넷째, 거버넌스·감사·평가 요건이 *PoC 범위 외* 로 미뤄집니다. 다섯째, 본격 배포 단계에 진입 시 *시스템 통합* 의 부담이 일제히 드러나 *재 PoC* 가 발생합니다. 이 5 단계 반복이 한국 공공·기업의 *플랫폼 통합 이전 단계 정체* 의 구조입니다 [S13].

이 패턴의 본질적 한계는 *각 PoC 가 자기만의 비용을 발생시키면서, 자산화는 0 에 가까운* 점입니다. 5 차례의 PoC 에 동일한 3 개월 × 5 = 15 개월이 들었다면, 15 개월의 인력·예산이 사내 자산으로 누적되지 않고 *각 시스템의 폐쇄 영역* 에 갇히는 셈입니다. 한국 공공기관과 기업의 *PoC 누적 사례* 가 이 구조를 보여줍니다 [S26, S28].

본 백서가 5 장 (13 컴포넌트 매트릭스) 과 부록 A (RFP 체크리스트) 에서 *플랫폼 단위 발주* 의 형식을 제공하는 이유가 여기 있습니다. 시스템별 RFP 가 *컴포넌트 일체를 요구하는* 형식으로 진화하지 않으면, 위 5 단계 반복은 끊어지지 않습니다. KOFIA 의 LLM ETF 챗봇 RFP 가 이미 *시스템별 RFP 가 사실상 플랫폼 컴포넌트 일체를 요구하는* 형태로 진화한 것이 그 신호입니다 [S28].

3.3.2 격차 해소의 단일 경로 — AI Native Platform 통합

위 4 원인 (거버넌스, 관찰가능성, 데이터, 컴포넌트) + 한국 패턴 5 특성을 모두 분석하면, *동일한 해결책* 이 도출됩니다. *통합 플랫폼 단위 의사결정* 입니다. 거버넌스를 한 곳에서, 관찰가능성을 한 곳에서, 데이터를 한 곳에서, 컴포넌트를 한 곳에서 통합 운영하면, 4 원인이 동시에 해소됩니다 [S09, S26].

이 *통합 단위* 의 정확한 정의가 다음 4 장의 주제입니다. Ampcome 의 4 계층 모델과 Digital Applied 의 5 계층 모델이 이 통합의 *아키텍처 reference* 를 제공합니다. 본 백서 4 장은 두 모델을 통합 매트릭스로 정렬하여 한국 의사결정자가 *어느 계층의 어느 컴포넌트가 누락되어 있는가* 를 한눈에 진단할 수 있게 합니다 [S09, S10].

격차 해소의 *시간 비용* 도 무시할 수 없습니다. 본격 배포 진입까지의 평균 시간이 *시스템별 발주 누적 시* 3~5 년으로 추정되는 반면, *플랫폼 통합 시* 1~2 년으로 단축 가능합니다 (illustrative — 실제 워크로드별 변동). 본격 배포 진입 시점이 곧 *경쟁 우위 확보 시점* 이므로, 시간 비용은 직접적인 사업 가치로 환산됩니다 [S07, S08].

본 장의 결론은 한 줄로 정리됩니다. *86% 정체의 정체는 모델이 아니라 인프라이며, 인프라의 통합 단위는 플랫폼이다.* 본 백서 4 장이 이 플랫폼의 정확한 정의를, 5 장이 13 컴포넌트의 매트릭스를, 6 장이 시스템별 발주 대

비 6 축 비교를 제공합니다. 4·5·6 장은 본 장의 결론을 의사결정 자산으로 전환하는 본론입니다 [S09, S13, S22].

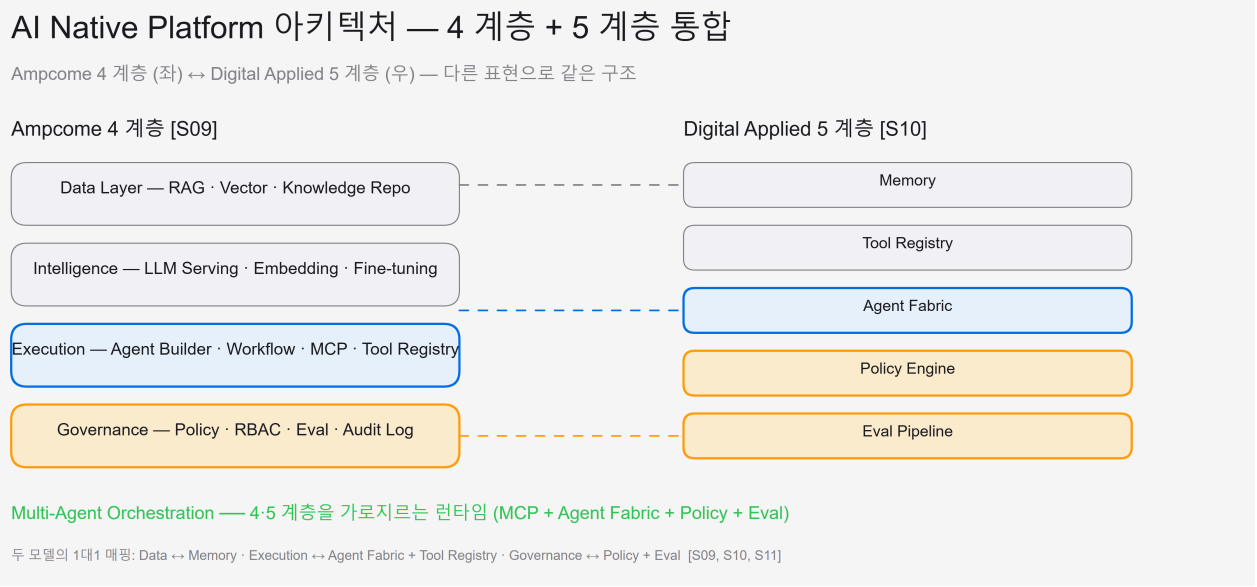
4장. AI Native Platform 의 정의와 4~5 계층 아키텍처

AI Native Platform 을 *정확히 무엇* 이라고 정의하는가에 따라 RFP 항목과 의사결정 척도가 달라집니다. 정의가 모호한 채로 발주하면, 응답 RFP 의 비교 자체가 불가능합니다. 본 장은 시장에 존재하는 두 reference architecture 를 통합 매트릭스로 정렬하여 한국 의사결정자가 *어느 계층의 어느 컴포넌트가 누락되어 있는가* 를 한눈에 점검할 수 있게 합니다.

2026년 엔터프라이즈 Agentic AI 아키텍처는 두 가지 reference 로 수렴합니다. 첫째, Ampcome 의 *Data → Intelligence → Execution → Governance + Multi-Agent Orchestration* 의 4 계층 모델 [S09]. 둘째, Digital Applied 의 *Agent Fabric / Tool Registry / Memory / Policy Engine / Eval Pipeline* 의 5 계층 모델 [S10]. McKinsey QuantumBlack 의 미래 대비 Agentic Platform 아키텍처도 두 모델과 정합합니다 [S11].

두 모델은 *다른 표현으로 같은 구조* 를 가리킵니다. Ampcome 의 *Execution Layer* 가 Digital Applied 의 *Agent Fabric + Tool Registry* 를 포괄하고, Ampcome 의 *Governance Layer* 가 Digital Applied 의 *Policy Engine + Eval Pipeline* 을 포괄하는 식의 1대1 매핑이 가능합니다. 본 백서는 두 모델의 통합 매트릭스를 4 장의 핵심 자산으로 제시합니다.

본 장의 시각 자산은 4 계층 + 5 계층 통합 아키텍처 다이어그램 ([FIGURE: ainp-4plus5-layer]) 입니다. 좌측에 Ampcome 4 계층이 그려지고, 우측에 Digital Applied 5 계층이 병렬로 그려진 뒤, 가운데 매핑 화살표가 1대1 정렬을 표시하는 형식입니다.



캡션: 4 계층 (Ampcome) + 5 계층 (Digital Applied) 통합 아키텍처 다이어그램

다이어그램의 좌측에 Ampcome 4 계층이 아래에서 위로 그려집니다 — Data Layer (RAG·Vector·Knowledge Repo), Intelligence Layer (LLM Serving·Embedding·Fine-tuning), Execution Layer (Agent Builder·Workflow·MCP·Tool Registry), Governance Layer

(Policy·RBAC·Eval·Audit Log). 우측에 Digital Applied 5 계층이 별도로 그려집니다 — Memory, Tool Registry, Agent Fabric, Policy Engine, Eval Pipeline. 가운데에는 두 모델 간 1대1 매핑 화살표가 그려집니다 — Ampcome Data Layer ↔ Digital Applied Memory, Ampcome Execution Layer ↔ Digital Applied Agent Fabric + Tool Registry, Ampcome Governance Layer ↔ Digital Applied Policy Engine + Eval Pipeline. 다이어그램 가로축의 상단에는 Multi-Agent Orchestration 이 모든 계층을 가로지르는 런타임으로 표시되고, 그 아래에 MCP 가 Agent ↔ Tool / Agent ↔ Agent 의 universal adapter 로 표시됩니다. 다이어그램 하단에는 13 컴포넌트 매트릭스 (5 장) 와의 매핑 박스가 부속 설명으로 그려집니다.

4.1 Ampcome 4 계층 모델

4.1.1 Data → Intelligence Layer — 데이터·모델·임베딩의 계층

Ampcome 의 *Data Layer* 는 AI Agent 가 활용하는 모든 데이터 자산을 포괄합니다. 구체적으로는 RAG 의 검색 대상이 되는 사내 문서 저장소, Vector DB 에 적재된 임베딩, GraphRAG 의 지식 그래프, Knowledge Repo 의 markdown 자산이 모두 이 계층에 포함됩니다. 한국 공공·기업 환경에서 *데이터 주권* 의 핵심이 이 계층입니다 [S09, S22].

Data Layer 의 핵심 컴포넌트는 다음과 같이 정리됩니다. Vector DB (Milvus, Qdrant, Weaviate, PGVector), Knowledge Repo (Obsidian Sync, Outline, BookStack), GraphRAG 엔진, RAG 파이프라인 (LangChain Retrievers, LlamaIndex). 이 컴포넌트들이 *각 시스템 폐쇄* 가 아니라 *조직 공통 자산* 으로 운영되는가가 본격 배포 진입의 핵심 조건입니다 [S22, S23].

Intelligence Layer 는 모델 자체의 운영을 다룹니다. LLM Serving (vLLM, Ollama, Triton, TGI 등 셀프 호스팅 도구 또는 OpenAI·Anthropic·Bedrock API 같은 매니지드 서비스), Embedding 모델 서빙 (sentence-transformers, bge-m3, Instructor 등), Fine-tuning 파이프라인 (LoRA, QLoRA, full fine-tuning) 이 모두 이 계층에 포함됩니다 [S22].

한국 시장에서 Intelligence Layer 의 의사결정은 *데이터 주권 + 모델 주권* 의 결합으로 진행됩니다. 외부 API 사용은 *모델 주권 부재* 의 리스크를 안고, 셀프 호스팅은 *운영 인력 부담* 을 안습니다. 본 백서 5 장의 13 컴포넌트 매트릭스에서 LLM Serving 영역의 의사결정 가이드가 이 trade-off 를 정렬합니다 [S22, S26].

Data + Intelligence 두 계층의 결합 패턴이 *RAG* 입니다. RAG 가 본격 도입의 가장 일반적인 진입점인 이유는, 이 두 계층만 갖춰지면 *Context Engineering* 단계 (본 백서 1 장 §1.1.2) 까지 진입 가능하기 때문입니다. 그러나 본격 배포로의 진입은 다음 두 계층 (Execution + Governance) 이 함께 갖춰져야 가능합니다 [S09, S13].

4.1.2 Execution → Governance Layer — 실행·정책의 계층

Execution Layer 는 Agent 가 *실제로 행동* 하는 영역을 다룹니다. Agent Builder (LangGraph, CrewAI, AutoGen 등 코드 기반 프레임워크), Visual Workflow (Dify, n8n, Flowise 등 노코드 도구), MCP Server (Agent ↔ Tool 표준 인터페이스), Tool Registry (도구 카탈로그) 가 모두 이 계층에 포함됩니다. LLM Gateway (LiteLLM, OpenRouter, Cloudflare AI Gateway) 도 이 계층의 부속 컴포넌트입니다 [S09, S22, S24].

Execution Layer 의 핵심 가치는 *Harness Engineering* (본 백서 1 장 §1.2.1) 의 표준화입니다. Agent 가 *어떤 도구를 어떤 순서로* 호출할지를 정의하는 흐름이 이 계층에 박제됩니다. LangGraph 의 상태 머신, AutoGen 의 Conversation Loop, CrewAI 의 Role-based Multi-agent 가 각각 다른 Harness 패턴을 제공합니다. 워크로드 복잡도에 따라 선택이 달라집니다 [S22].

Governance Layer 는 *통제와 평가* 의 영역을 다룹니다. Policy Engine (OPA, Kyverno), RBAC, 감사 로그 (Audit Log), 평가 파이프라인 (Promptfoo, DeepEval, Ragas, LLM-as-judge), PII 마스킹·콘텐츠 필터링 이 모두 이 계층에 포함됩니다. 한국 공공·기업의 *AI 의사결정 감사 요건* (공공기관 정보보호 인증, 일반 기업의 컴플라이언스·내부 통제) 의 직접적 대응 영역입니다 [S09, S25].

Governance Layer 가 PoC 단계에서는 *미뤄지지만* 본격 배포 단계에서는 *반드시 필요* 한 이유가 분명합니다. PoC 는 *모델이 답을 잘 만드는가* 만 검증해도 통과 가능하지만, 본격 배포는 *PII 마스킹이 누락되지 않는가, 감사 로그가 완전한가, 평가 점수가 임계값 이상인가* 의 세 질문을 모두 통과해야 합니다 [S25].

Execution + Governance 의 결합은 *Agent 의 본격 배포 진입* 의 두 축입니다. Execution 만 갖춰지면 Agent 는 작동하지만 *통제되지 않으므로* 본격 배포 불가입니다. Governance 만 갖춰지면 *정책은 있지만 Agent 가 없으므로* 본격 배포 불가입니다. 두 계층의 *동시 결합* 이 본격 배포 진입의 정확한 조건입니다 [S09, S13].

4.2 Digital Applied 5 계층 모델

4.2.1 Agent Fabric + Tool Registry — Agent 와 도구의 표준

Digital Applied 의 5 계층 모델은 *Agentic 측면* 을 강조한 reference architecture 입니다. 첫 번째 계층 *Agent Fabric* 은 Agent 의 라이프사이클 (생성·실행·중단·재시작·삭제) 을 표준 인터페이스로 관리합니다. Agent 가 K8s 컨테이너로 패키지화되고, Agent Fabric 이 그 컨테이너들의 운영 표준을 제공하는 구조입니다 [S10].

Agent Fabric 의 가치는 *Agent 의 자산화* 에 있습니다. 사내에서 만들어진 Agent 가 *재사용 가능한 자산* 으로 누적되려면, 표준 라이프사이클 관리가 전제 조건입니다. *어느 부서의 어느 Agent 가 누가 만들었고, 무엇을 입력으로 받고, 무엇을 출력하는가* 를 한 곳에서 조회 가능해야 합니다 [S10, S11].

두 번째 계층 *Tool Registry* 는 Agent 가 호출 가능한 도구들의 카탈로그를 관리합니다. MCP Server 의 등장으로 도구 등록의 표준 인터페이스가 확립되었고 (본 백서 9 장 §9.1), Tool Registry 는 이 MCP Server 들의 통합 카탈로그 역할을 합니다. 새 도구가 추가되면 모든 Agent 가 즉시 호출 가능한 구조입니다 [S10, S24].

Tool Registry 가 *한국 RFP 의 새 필수 항목* 으로 떠오르는 이유는 분명합니다. 사내 도구의 통합 카탈로그가 부재하면, 새 Agent 를 만들 때마다 *도구 통합 코드를 다시 작성* 해야 하기 때문입니다. Tool Registry 를 도입하면 한 번 등록한 도구를 N 개의 Agent 가 모두 재사용합니다. 이는 *Time-to-First-Agent* 단축의 핵심 메커니즘입니다 [S10].

Agent Fabric + Tool Registry 의 결합은 *Composable AI Platform* 의 핵심입니다. Agent 와 도구가 각각 표준 인터페이스로 등록되면, 새 워크로드에 대해 *조합* 만으로 새 시스템이 만들어집니다. 이는 본 백서 6 장의 *Time-to-First-Agent 단축* 의 정량적 근거가 됩니다 [S10, S23].

4.2.2 Memory + Policy + Eval — 기억·정책·평가의 3 축

세 번째 계층 *Memory* 는 Agent 의 단기·장기 상태 저장을 다룹니다. Vector DB 의 임베딩 기억, Knowledge Repo 의 markdown 자산, 세션별 conversation history, 사용자별 개인화 상태가 모두 이 계층에 포함됩니다. Ampcome 4 계층 모델의 Data Layer 와 사실상 동일한 영역입니다 [S10].

Memory 계층의 핵심 의사결정은 *공통화* 입니다. 시스템별로 별도 메모리를 가지면 *데이터 폐쇄* (본 백서 3 장 §3.2.2) 의 원인이 됩니다. 조직 공통 메모리를 두면 새 Agent 가 *기존 메모리 자산을 즉시 활용* 가능합니다. 이는 *Scaled 단계 진입* 의 전제 조건입니다 [S10, S13].

네 번째 계층 *Policy Engine* 은 Agent 의 행동을 정책으로 통제합니다. OPA (Open Policy Agent) 의 Rego 정책, Kyverno 의 K8s native 정책이 표준 도구입니다. *어떤 Agent 가 어떤 도구를 어떤 사용자에 대해 호출 가능한가* 를 단일 정책 언어로 정의 가능합니다 [S25].

Policy Engine 의 결합 가치는 *정책 시행의 일관성* 입니다. 시스템별로 정책이 흩어지면 동일 사용자에 대해 다른 보호 수준이 적용되는 *비일관성* 이 발생합니다 (본 백서 3 장 §3.2.1). Policy Engine 한 곳에서 통제하면 모든 Agent 에 동일 정책이 즉시 적용됩니다. 정책 변경의 *시간 비용* 도 시스템 수만큼 줄어듭니다 [S25].

다섯 번째 계층 *Eval Pipeline* 은 Agent 출력의 품질을 자동 평가합니다. Promptfoo, DeepEval, Ragas, LLM-as-judge 가 표준 도구입니다. *PoC 에서 본격 배포로* 진입하려면 *Eval Pipeline 의 자동화* 가 전제 조건입니다 — 사람이 매 배포마다 수동 평가하는 구조로는 본격 배포 진입 자체가 불가능하기 때문입니다 [S10, S25].

Memory + Policy + Eval 의 3 축이 *동시에 운영* 되어야 *Scaled 단계 진입* 가능합니다. 본 백서 7 장에서 다루는 *프로덕션 Agent 20~40+* 임계값은, 이 3 축의 자동화 없이는 도달 불가능합니다. 단일 Agent 의 모델 품질이 아니라, 3 축의 *플랫폼 수준 자동화* 가 핵심 경쟁 우위입니다 [S10, S13].

4.3 통합 매트릭스 + Multi-Agent Orchestration

4.3.1 4 계층 ↔ 5 계층 통합 매핑표

Ampcome 4 계층과 Digital Applied 5 계층은 *다른 표현으로 같은 구조* 를 가리킵니다. 정확한 1대1 매핑은 다음과 같이 정리됩니다. Ampcome Data Layer 는 Digital Applied Memory 에 정합합니다. Ampcome Intelligence Layer 는 Digital Applied 의 명시적 계층이 아니지만 *Agent Fabric 의 모델 슬롯* 으로 흡수됩니다. Ampcome Execution Layer 는 Digital Applied Agent Fabric + Tool Registry 를 포괄합니다. Ampcome Governance Layer 는 Digital Applied Policy Engine + Eval Pipeline 을 포괄합니다 [S09, S10].

매핑이 1대1 로 떨어지지 않는 부분은 두 모델의 *강조점 차이* 를 반영합니다. Ampcome 은 *데이터 흐름의 수직 누적* 을 강조 (Data → Intelligence → Execution → Governance 의 순서) 하고, Digital Applied 는 *Agent 라이프사이클의 모듈화* 를 강조합니다. 두 시각은 보완적입니다 — 같은 아키텍처를 다른 축에서 본 결과일 뿐입니다 [S09, S10, S11].

본 백서의 통합 매트릭스는 두 모델의 *합집합* 으로 정렬됩니다. 즉 Ampcome 4 계층 + Digital Applied 5 계층 = *통합 5+1 계층* (Data, Intelligence, Execution-Agent Fabric, Execution-Tool Registry,

Governance-Policy, Governance-Eval) 의 6 영역으로 펼쳐집니다. 본 백서 5 장의 13 컴포넌트 매트릭스는 이 6 영역을 다시 13 개의 *RFP 항목* 으로 펼친 형식입니다 [S09, S10].

두 reference 의 통합은 한국 의사결정자에게 *안심* 을 제공합니다. *어느 reference 를 채택해도 동일한 컴포넌트가 도출* 된다는 사실이 RFP 응답 비교의 안정성을 보장하기 때문입니다. 글로벌 reference 의 *수렴* 흐름이 한국 시장에서도 그대로 적용됩니다.

4.3.2 Multi-Agent Orchestration — 가로축의 런타임

Multi-Agent Orchestration 은 4 계층 또는 5 계층을 *가로지르는 런타임* 입니다. 별도의 계층이 아니라, 모든 계층 위에 걸쳐서 작동하는 *통합 운영 환경* 입니다. Ampcome 4 계층 다이어그램과 Digital Applied 5 계층 다이어그램 모두 이 Orchestration 을 가로축으로 표시합니다 [S09, S10, S11].

Orchestration 의 핵심 구성 요소는 다음과 같이 정리됩니다. *Agent 간 라우팅* (어떤 사용자 요청을 어떤 Agent 에 보낼지 결정), *Agent 간 협업* (한 Agent 가 다른 Agent 의 출력을 입력으로 받는 흐름), *정책 시행* (Policy Engine 의 실시간 적용), *상태 추적* (현재 Agent 흐름의 메타데이터), *결과 평가* (Eval Pipeline 의 실시간 호출) [S11].

MCP 가 이 Orchestration 의 *표준 인터페이스* 입니다. Agent ↔ Tool, Agent ↔ Data, Agent ↔ 다른 Agent 의 모든 호출이 MCP 표준 위에서 일어나면, Orchestration 의 라우팅·협업·정책 시행이 *단일 추적 가능 단위* 로 통합됩니다. 본 백서 9 장에서 MCP 의 표준화 경로와 한국 도입 의사결정을 상세 다룹니다 [S24, S25].

Multi-Agent Orchestration 이 *한국 RFP 의 다음 필수 항목* 으로 등장하는 흐름은 분명합니다. KOFIA 의 LLM ETF 챗봇 RFP 가 *단일 챗봇 사업* 임에도 LLM·VectorDB·검색·평가·운영을 묶음으로 요구한 것이 첫 신호입니다 [S28]. 향후 1~2 년 안에 *Orchestration 까지 한 묶음으로 발주* 하는 RFP 패턴이 표준화될 가능성이 큽니다.

본 4 장의 결론은 한 줄로 정리됩니다. *AI Native Platform = 4 계층 또는 5 계층의 수렴 아키텍처 + Multi-Agent Orchestration 의 가로축 런타임.* 본 정의를 기반으로 다음 5 장에서 13 컴포넌트 매트릭스를 RFP 형식으로 펼치고, 6 장에서 시스템별 발주 대비 6 축 비교를 진행합니다. 4·5·6 장이 본 백서의 *정의 + 매트릭스 + 비교* 의 3 자산을 완성하는 본론입니다 [S09, S10, S11, S22].

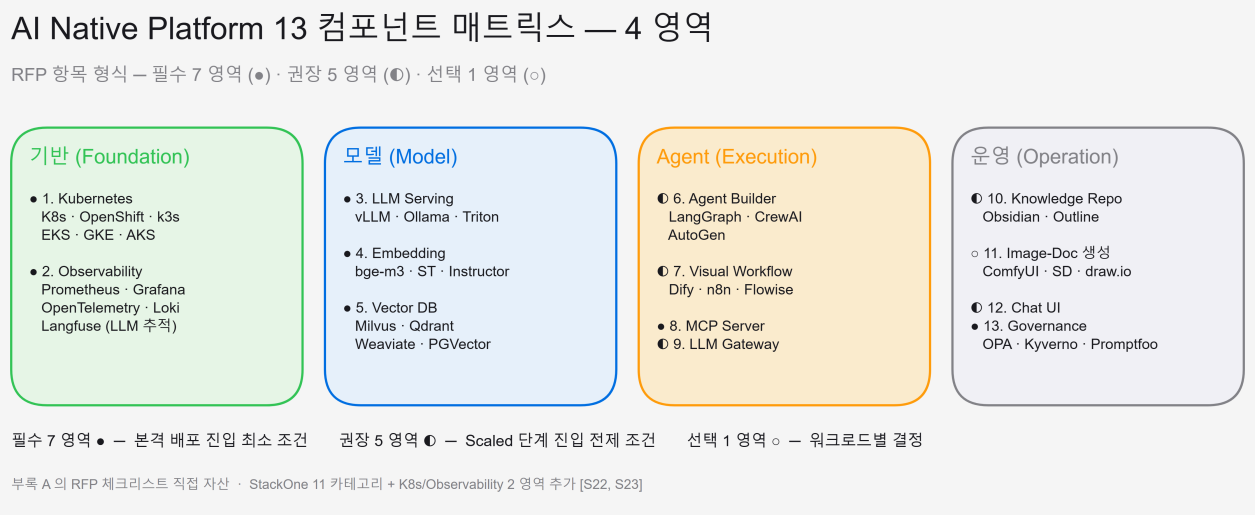
5장. AI Native Platform 13 컴포넌트 매트릭스

한국 RFP 의 *AI 항목* 이 1~2 줄로 끝나는 시대는 끝났습니다. KOFIA 의 LLM ETF 챗봇 RFP 가 단일 챗봇 사업임에도 LLM·VectorDB·검색·평가·운영을 묶음으로 요구한 것이 그 신호입니다 [S28]. 본 장의 13 컴포넌트 매트릭스는 *RFP 항목으로 그대로 옮길 수 있는* 형식으로 설계되었습니다.

13 영역은 다음과 같이 정리됩니다. 기반 영역 (Kubernetes, Observability) 의 2 컴포넌트, 모델 영역 (LLM Serving, Embedding, Vector DB) 의 3 컴포넌트, Agent 영역 (Agent Builder, Visual Workflow, MCP Server, LLM Gateway) 의 4 컴포넌트, 운영 영역 (Knowledge Repo, Image-Doc Generation, Chat UI, Governance) 의 4 컴포넌트. 본 13 영역은 4 장의 *4 계층 또는 5 계층* 통합 매트릭스를 RFP 형식으로 펼친 결과입니다 [S09, S10, S22, S23].

StackOne 의 2026 landscape 보고서는 Agentic AI 도구를 *11 카테고리, 120+ 도구* 로 매핑합니다 [S23]. 본 백서가 13 영역으로 확장한 차이는 *Kubernetes (컨테이너 오케스트레이션) + Observability* 의 2 영역을 명시 추가한 점입니다. K8s 와 Observability 는 *AI Native 의 Native 정의* 를 강제하는 컴포넌트이므로 본 백서에서는 이 둘을 1·2 영역으로 명시했습니다 [S01, S02].

본 장의 시각 자산은 13 영역 × (필수 컴포넌트 / 대표 오픈소스 / 대표 상용 매니지드 / 출처) 의 매트릭스 표 ([FIGURE: ainp-13-components-matrix]) 입니다. 부록 A 의 RFP 체크리스트가 본 매트릭스의 직접적 확장입니다.



캡션: AI Native Platform 13 컴포넌트 매트릭스 — 4 영역 × 13 컴포넌트 × 의사결정 축

매트릭스의 좌측에 4 영역 라벨이 그려집니다 — 기반, 모델, Agent, 운영. 우측에는 각 영역의 컴포넌트가 행으로 정렬됩니다 — 1: Kubernetes, 2: Observability (기반 영역), 3: LLM Serving, 4: Embedding, 5: Vector DB (모델 영역), 6: Agent Builder, 7: Visual Workflow, 8: MCP Server, 9: LLM Gateway (Agent 영역), 10: Knowledge Repo, 11: Image-Doc Generation, 12: Chat UI, 13: Governance (운영 영역). 매트릭스의 가로축에는 의사결정 축이 정렬됩니다 — 필수 컴포넌트 (예: GPU 스케줄링, KV 캐시, 하이브리드 검색), 대표 오픈소스/통합 패키지 (예: K8s+OpenShift+MSAP.ai, vLLM+Ollama, Milvus+Qdrant), 대표 상용 매니지드 (예: EKS+GKE+AKS, OpenAI+Anthropic+Bedrock, Pinecone),

망분리 적합 점수 (○/△/×). 매트릭스의 하단에는 *부록 A RFP 체크리스트* 연결 화살표가 표시되어, 본 매트릭스가 의사결정 자산으로 직접 활용 가능함을 시각화합니다.

5.1 기반 영역 — Kubernetes + Observability

5.1.1 Kubernetes — 배포·자동확장·GPU 스케줄링

Kubernetes 영역의 의사결정 축은 *기존 자산 활용성, 운영 인력, GPU 스케줄링, 망분리 적합성* 의 4 가지입니다. 대표 후보는 오픈소스 측 (자체 운영 K8s, OpenShift Origin), 온프레미스 통합 AIaaS 패키지 측 (Red Hat OpenShift / OpenShift AI, MSAP.ai), 상용 매니지드 측 (AWS EKS, Google GKE, Azure AKS) 으로 정리됩니다 [S01, S02].

기존 K8s 자산 활용성은 한국 공공·기업의 의사결정에서 *가장 큰 가중치* 를 갖는 축입니다. 이미 OpenShift 또는 자체 K8s 를 운영하는 조직은 *별도 도입 없이* AI Native Platform 의 1 계층을 가진 셈입니다. 매몰 비용으로 보지 않고 *AI Native 출발점* 으로 재정의하면, 기존 운영 인력·모니터링·CI/CD 가 그대로 활용 가능합니다 [S01].

GPU 스케줄링은 AI 워크로드의 *특수 요건* 입니다. NVIDIA GPU Operator 가 K8s 표준이고, Karpenter 또는 Volcano 가 GPU 자원의 multi-tenant 관리 표준입니다. RFP 응답 시 *GPU Operator + Karpenter/Volcano 중 1 종 이상 채택* 이 사실상 필수 항목이 되었습니다 [S01, S22].

망분리 적합성은 한국 공공·기업에서 *우회할 수 없는 전제* 입니다. 컨테이너 이미지의 오프라인 패키지 특성과 K8s 의 망분리 환경 운영 표준이 결합되어, *컨테이너 + K8s = 망분리 패키지* 의 동치성이 성립합니다 (본 백서 2 장 §2.3.1). OpenShift, MSAP.ai, 자체 K8s 모두 망분리 환경에서 운영 가능하며, 매니지드 K8s (EKS·GKE·AKS) 는 망분리 적용 불가입니다 [S01, S26].

5.1.2 Observability — Prometheus·Grafana·OTel·Langfuse

Observability 영역의 의사결정 축은 *일반 Observability 표준 + LLM 호출 추적의 신규성* 의 결합입니다. 일반 Observability 는 Prometheus (메트릭), Grafana (시각화), OpenTelemetry (트레이스), Loki (로그) 가 표준입니다. LLM 호출 추적은 Langfuse, Helicone, Phoenix 같은 *LLM 전용 도구* 가 추가로 필요합니다 [S22, S25].

LLM 호출 추적의 신규성이 한국 RFP 의 *새 필수 항목* 으로 떠올랐습니다. 기존 APM (Application Performance Monitoring) 도구만으로는 LLM 호출 시점, 입력, 출력, 모델 버전, 호출 비용, 호출자 ID 를 모두 추적할 수 없기 때문입니다. 공공기관과 기업의 *AI 의사결정 감사 요건* 이 이 추적의 직접적 동기입니다 [S25].

Langfuse 가 오픈소스 측에서 가장 빠르게 표준화된 도구입니다. K8s Helm Chart 로 배포 가능하고, OpenAI·Anthropic·Bedrock·자체 LLM 모두 추적 가능한 SDK 를 제공합니다. 망분리 환경에서도 셀프 호스팅 가능하며, 한국 도입 사례도 누적 단계입니다 [S22].

본 영역의 RFP 표현은 *일반 Observability 4 종 + LLM 추적 1 종 = 최소 5 컴포넌트* 형식으로 정리 가능합니다. *기존 Observability 에 LLM 추적 추가 1 항* 의 단순 mental model 이 한국 의사결정자에게 직관적으로 닿습니다.

5.2 모델 영역 — LLM Serving + Embedding + Vector

5.2.1 LLM Serving — vLLM·Ollama·Triton·TGI vs API

LLM Serving 영역의 의사결정 축은 *데이터 주권, 라이선스, TCO* 의 3 가지입니다. 셀프 호스팅 후보는 vLLM (KV 캐시·PagedAttention 표준), Ollama (경량 배포 표준), NVIDIA Triton (공식 추론 서버), HuggingFace TGI (Text Generation Inference) 가 있습니다. 매니지드 API 후보는 OpenAI, Anthropic, AWS Bedrock, Google Vertex AI 입니다 [S22].

데이터 주권 축은 한국 공공·기업에서 *결정적* 입니다. 외부 API 사용 시 *프롬프트와 출력이 외부로 전송* 되므로 *PII·기밀 정보 노출 리스크* 가 발생합니다. 금융위 「금융권 AI 플랫폼」 정책이 *오픈소스 AI 모델 공동 인프라* 를 강조한 이유가 이 데이터 주권 우려와 정합합니다 [S26].

라이선스 축은 *vLLM 의 Apache 2.0, Ollama 의 MIT, Triton 의 BSD-3 라이선스* 가 모두 *상업 활용 가능* 한 자유로운 형태입니다. HuggingFace TGI 는 일부 모델 라이선스 제약이 있으므로 *호스팅하는 모델* 의 라이선스를 별도 점검해야 합니다 [S22, S23].

TCO 축은 *셀프 호스팅 vs API 사용* 의 핵심 비교입니다. 일정 호출량 이상에서는 셀프 호스팅이 TCO 우위인 분기점이 존재합니다 (illustrative — 워크로드별 정밀 계산 필요). 한국 공공·기업의 *3 년 누적 TCO* 분석은 본 백서 부록 B 의 6 축 비교에서 정렬됩니다 [S04, S22].

망분리 환경에서의 LLM Serving 의사결정은 *셀프 호스팅 필수* 로 좁혀집니다. vLLM + Ollama + Triton 의 3 종이 컨테이너 이미지로 배포 가능하므로, *컨테이너 + K8s = 망분리 패키지* 의 동치성을 그대로 활용합니다 (본 백서 2 장 §2.3.1) [S01, S22].

5.2.2 Embedding + Vector DB — Milvus·Qdrant·Weaviate·PGVector

Embedding 영역은 *모델 선택* 의 의사결정입니다. sentence-transformers (다국어 표준), bge-m3 (다국어 + 다양한 차원), Instructor (instruction-aware embedding), OpenAI Embeddings (매니지드 API) 가 주요 후보입니다. 한국어 임베딩의 품질은 bge-m3 와 sentence-transformers 의 한국어 fine-tuning 모델이 글로벌 평균을 상회합니다 [S22, S23].

Vector DB 영역은 *데이터 주권, 운영 인력, 하이브리드 검색* 의 3 축으로 정리됩니다. 4 대 오픈소스 후보의 특징은 다음과 같이 정합됩니다. Milvus 는 *대규모 분산 처리* 강점, Qdrant 는 *Rust 기반 단일 노드 성능* 강점, Weaviate 는 *GraphQL + 모듈 시스템* 강점, PGVector 는 *기존 PostgreSQL 자산 활용성* 강점입니다 [S22, S23].

PostgreSQL 을 이미 운영하는 한국 공공·기업 조직은 PGVector 를 우선 검토할 가치가 큽니다. 기존 DB 운영 인력과 도구를 그대로 활용하면서 Vector 검색 기능을 추가할 수 있기 때문입니다. *기존 자산 활용성* 의 의사결정 가중치가 큰 환경에서는 합리적 선택지입니다.

대규모 임베딩 (1억 건 이상) 의 분산 운영이 필요한 환경에서는 Milvus 가 우위입니다. K8s Operator 와 분산 인덱싱 구조가 표준화되어 있어 운영 인력 부담이 상대적으로 낮습니다. 단일 노드 고성능이 필요한 환경에서는 Qdrant 의 Rust 기반 성능이 매력적입니다 [S22].

하이브리드 검색 (vector + keyword) 의 표준화는 4 후보 모두 지원합니다. BM25 또는 sparse vector 와 dense vector 의 결합 검색이 *임베딩만으로는 부족한 검색 정확도* 를 보강합니다. 한국어 환경에서는 BM25 의 형태소 분석기 (nori, mecab-ko) 결합이 추가 의사결정 항목입니다 [S22, S23].

매니지드 측 후보 (Pinecone, Elasticsearch Vector) 는 *데이터 주권 우려* 로 한국 공공·기업에서 도입 비율이 낮습니다. 단 일부 시범 사업에서는 매니지드 활용 사례도 있으며, 망분리 환경 외부에서는 합리적 선택지가 될 수 있습니다.

5.3 Agent 영역 — Builder + Workflow + MCP + Gateway

5.3.1 Agent Builder — LangGraph·CrewAI·AutoGen

Agent Builder 영역의 의사결정 축은 *워크로드 패턴 + 인력 스킬* 의 2 가지입니다. 3 대 오픈소스 후보의 패턴은 다음과 같이 정합됩니다. LangGraph 는 *상태 머신 + 그래프 흐름* 패턴, CrewAI 는 *역할 기반 Multi-agent* 패턴, AutoGen 은 *대화형 Multi-agent* 패턴입니다 [S21, S22].

워크로드 패턴 축의 의사결정은 다음과 같이 정리됩니다. *복잡한 분기·재시도 흐름* 이 필요한 워크로드 (예: 금융 거래 분석, 컴플라이언스 점검) 는 LangGraph 의 상태 머신이 적합합니다. *여러 역할의 협업* 이 명확한 워크로드 (예: 컨설팅 보고서 작성, 시장 분석) 는 CrewAI 의 역할 팀이 직관적입니다. *자유로운 대화형 협업* 이 필요한 워크로드 (예: 고객 응대, 내부 협업) 는 AutoGen 의 Conversation Loop 이 강점입니다 [S22].

인력 스킬 축은 *Python + LangChain 친숙도* 의 측면에서 정합됩니다. 3 도구 모두 Python 기반이고 LangChain 과의 통합이 표준입니다. 한국 IT 인력의 Python·LangChain 친숙도는 글로벌 평균과 유사하며, 3 도구의 학습 곡선은 *2~4 주 이내* 가 일반적입니다 [S22].

본 영역의 *단일 패턴 선택* 보다는 *3 패턴 모두 운영 가능한 플랫폼* 권장이 합리적입니다. 워크로드별로 적합 패턴이 다르므로, 한 패턴만 채택하면 *워크로드별 최적화 불가* 의 한계가 생깁니다. AI Native Platform 의 *Composable* 정의 (본 백서 4 장 §4.2.1) 가 이 *N 패턴 동시 지원* 을 가능하게 합니다 [S22, S23].

OpenAgents 와 LangSmith Agents 같은 추가 후보도 있지만, 2026년 한국 시장 도입 비율은 3 대 후보 (LangGraph, CrewAI, AutoGen) 가 압도적입니다. 본 백서는 3 대 후보를 1 등급 후보로, 그 외를 보조 후보로 분류합니다 [S21].

5.3.2 MCP Server + LLM Gateway — 표준 인터페이스의 결합

MCP Server 영역은 *Agent ↔ Tool / Data / Agent* 의 universal adapter 입니다. Anthropic 발 (2024년 11월) → Linux Foundation 거버넌스 (2025년 12월) → OpenAI·Google·Microsoft·AWS 4 사 채택의 표준화 경로가 완성되었습니다 (본 백서 9 장 상세). 월간 SDK 다운로드 9,700만 회 규모입니다 [S24].

MCP Server 의 운영 의사결정은 *자체 호스팅 vs 외부 매니지드* 의 단순 구조로 정합됩니다. 망분리 환경에서는 자체 호스팅이 필수이며, GitHub·Slack·Postgres·AWS·Atlassian·Notion 등 주요 SaaS 의 공식 MCP Server 가 컨테이너 이미지로 배포 가능합니다 [S24, S25].

LLM Gateway 영역은 *멀티 모델 라우팅 + 키 관리 + 캐시 + 비용 통제* 의 4 가지를 단일점에서 처리합니다. LiteLLM, OpenRouter (오픈소스 측), OneAPI (중국 기원), Portkey (상용) 가 대표 후보이고, Cloudflare

AI Gateway 가 매니지 측의 대표입니다 [S04, S22].

LLM Gateway 의 핵심 가치는 *모델 교체 비용 0* 입니다. 시스템마다 별도 코드를 수정하는 대신 *Gateway 한 곳에서 모델 라우팅 설정만 변경* 하면 됩니다. GPT-4 → Claude → Gemini → 한국어 LLM 의 교체가 *분 단위 운영 작업* 으로 단축됩니다. 본 백서 6 장 §6.3.1 의 *모델 교체 비용* 축이 이 가치를 정량화합니다 [S04, S22].

MCP + Gateway 의 결합은 *벤더 락인 방어* 의 단일 결합점입니다. MCP 는 *도구 표준화*, Gateway 는 *모델 표준화* 를 제공하므로, 두 컴포넌트가 결합되면 *Agent 가 사용하는 모든 외부 자원의 표준 인터페이스화* 가 완성됩니다. 이는 본 백서 6 장의 6 축 비교에서 *벤더 락인 회피* 의 정량적 근거가 됩니다 [S04, S22, S24].

5.4 운영 영역 — Knowledge + Image-Doc + Chat UI + Governance

5.4.1 Knowledge Repo + Image-Doc 생성

Knowledge Repo 영역은 *조직 지식의 원천 저장소* 입니다. Obsidian Sync (markdown + 백링크), Outline (위키 형식), BookStack (책 형식), MediaWiki (전통적 위키) 가 오픈소스 측 대표이고, Confluence·Notion·SharePoint 가 상용 측 대표입니다 [S23].

Knowledge Repo 의 핵심 의사결정은 *기존 자산과의 정합* 입니다. Confluence 또는 SharePoint 를 이미 운영하는 조직은 *마이그레이션 비용 vs 통합 운영* 의 trade-off 가 필요합니다. RAG 의 검색 대상으로 활용하는 측면에서는 markdown 형식의 Obsidian, Outline, BookStack 이 *임베딩 친화적* 입니다 [S22, S23].

Image-Doc Generation 영역은 *이미지·다이어그램·문서 생성* 을 컨테이너로 운영합니다. ComfyUI (Stable Diffusion 워크플로우), Stable Diffusion (이미지 생성 표준), draw.io (다이어그램), Marp (markdown → 슬라이드), excalidraw (스케치형 다이어그램) 가 오픈소스 측 대표입니다 [S23].

망분리 환경에서 이미지 생성의 의사결정은 *Stable Diffusion 셀프 호스팅* 으로 좁혀집니다. ComfyUI 와 ForgeUI 가 K8s 컨테이너로 배포 가능하고, GPU 자원 활용도가 LLM Serving 과 *동일한 GPU pool 에서 공유* 가능합니다. 이는 *기존 GPU 자산 활용도 극대화* 의 측면에서 가치가 있습니다 [S23].

문서 자동 생성의 의사결정은 *makeexcali, draw.io, Marp* 의 결합 패턴이 표준입니다. 다이어그램은 excalidraw 의 자연어 → SVG 변환이 빠르게 표준화되고 있고, 슬라이드는 Marp 의 markdown → 슬라이드 변환이 표준입니다. 본 백서 자체도 makeexcali 의 인라인 figure 생성으로 본 장의 다이어그램들이 만들어집니다.

5.4.2 Chat UI + Governance — 직원 접점과 정책의 결합

Chat UI 영역은 *직원 접점* 의 통합 인터페이스를 다룹니다. OpenWebUI, AnythingLLM, LibreChat 이 오픈소스 측 대표이고, Microsoft Copilot Studio, Glean, Slack AI 가 상용 측 대표입니다. 한국 공공·기업에서는 OpenWebUI 와 AnythingLLM 의 도입 사례가 가장 빠르게 누적 중입니다 [S22, S23].

Chat UI 의 핵심 가치는 *직원 채팅 = 정책 게이트* 의 단일점 mental model 입니다. 모든 직원의 AI 사용이 *단일 Chat UI* 를 통하면, PII 마스킹·콘텐츠 필터링·감사 로그가 *그 한 곳* 에서 통합 운영됩니다. 시스템별 챗봇이

흩어진 구조에서는 불가능한 *정책 시행의 일관성* 이 가능해집니다 [S25].

Governance 영역은 *정책·평가의 통합 운영* 입니다. OPA (Open Policy Agent), Kyverno (K8s native policy) 가 정책 엔진 표준이고, Promptfoo, DeepEval, Ragas 가 평가 파이프라인 표준입니다. 상용 측에서는 Cisco AI Defense, Lakera, Protect AI 가 대표입니다 [S09, S25].

OPA + Kyverno 의 결합은 *K8s 위 AI 의 정책 통제* 의 단일점입니다. K8s 가 이미 OPA·Kyverno 를 표준 정책 엔진으로 채택하고 있으므로, *AI 컴포넌트의 정책 통제 = K8s 정책 통제* 의 재사용이 가능합니다. 즉 기존 K8s 거버넌스 인력의 *스킬 재활용* 이 그대로 성립합니다 [S25].

Promptfoo + DeepEval + Ragas 의 평가 파이프라인 결합은 *PoC → 본격 배포 전환점* 의 핵심 메커니즘입니다. 사람이 매 배포마다 수동 평가하는 구조로는 본격 배포 진입 자체가 불가능합니다 (본 백서 4 장 §4.2.2). 자동 평가 파이프라인이 *Scaled 단계 진입* 의 전제 조건입니다 [S10, S25].

본 13 컴포넌트 매트릭스의 결론은 한 줄로 정리됩니다. *AI Native Platform 의 13 영역은 K8s 위에서 컨테이너로 운영 가능한 표준 컴포넌트의 집합이며, RFP 항목으로 그대로 활용 가능합니다.* 부록 A 의 RFP 체크리스트가 본 매트릭스의 직접적 확장이고, 부록 C 의 망분리 점검표가 본 13 영역의 한국 적합성 검증입니다. 다음 6 장에서 *왜 이 매트릭스가 시스템별 발주 대비 우위인가* 를 6 축으로 정량 비교합니다 [S22, S23, S25].

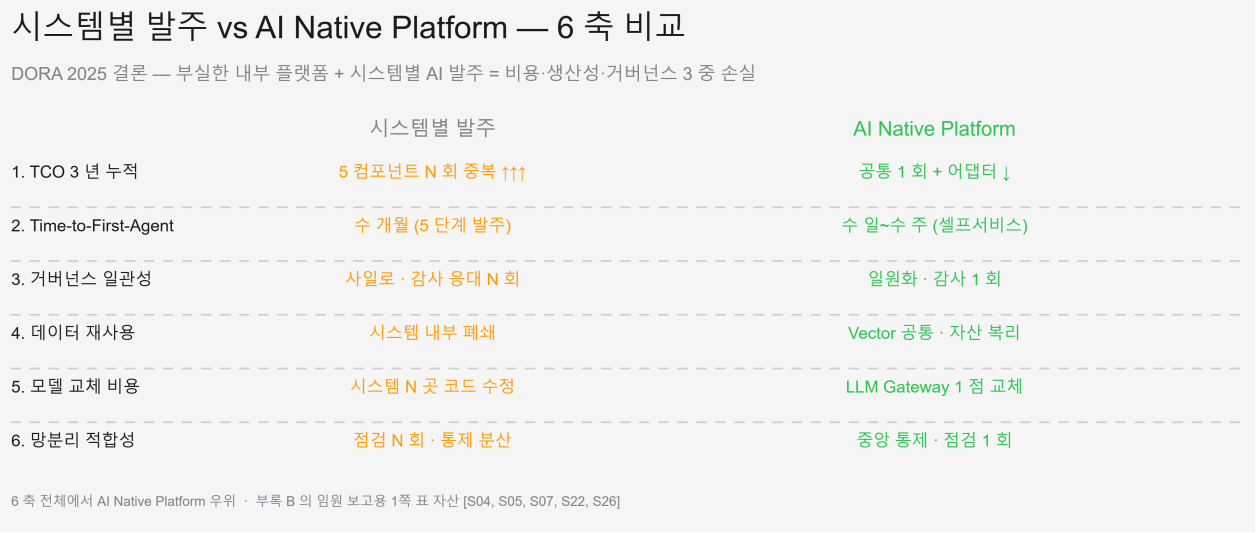
6장. 플랫폼 없는 AI vs AI Native Platform — 6축 비교

DORA 2025 의 결론을 한국 시장에 옮기면 이렇게 됩니다 — *부실한 내부 플랫폼 위에 시스템별 AI 발주를 누적하면 동일 컴포넌트를 반복 구매하면서 거버넌스가 사일로화되어 비용·생산성·거버넌스 3중 손실이 발생합니다* [S07]. 본 장의 6 축 비교 매트릭스는 이 결론을 6 개의 정량·정성 축으로 분해하여, 한국 의사결정자가 *왜 플랫폼이 먼저인가* 의 1쪽 결론을 즉시 추출할 수 있게 정렬합니다.

6 축은 다음과 같이 구성됩니다. TCO 3년 누적 (1 축), Time-to-First-Agent (2 축), 거버넌스 일관성 (3 축), 데이터 재사용 (4 축), 모델 교체 비용 (5 축), 망분리 적합성 (6 축). 각 축에서 시스템별 발주와 AI Native Platform 의 차이가 정량·정성으로 분명히 갈립니다.

본 6 축은 한국 공공·기업의 *3 년 IT 투자 계획* 수립 시 직접 활용 가능한 형식으로 정렬되었습니다. 임원 보고서의 첨부 자료로 본 매트릭스가 1 쪽으로 발췌 가능합니다. 부록 B 에서 본 매트릭스를 *보고서 첨부 형식* 으로 재구성합니다.

본 장의 시각 자산은 6 축 비교 매트릭스 + TCO 시뮬레이션 illustrative 그래프 ([FIGURE: silo-vs-ainp-6axis]) 입니다.



캡션: 시스템별 발주 vs AI Native Platform 6 축 비교 매트릭스 (illustrative)

매트릭스의 행은 6 축 (TCO 3년 누적, Time-to-First-Agent, 거버넌스 일관성, 데이터 재사용, 모델 교체 비용, 망분리 적합성) 으로 구성됩니다. 열은 *시스템별 AI 발주* 와 *AI Native Platform* 의 2 열로 정렬되고, 가장 우측에 *차이의 의사결정 함의* 열이 추가됩니다. 다이어그램의 하단에는 TCO 3 년 누적 비교 그래프 (illustrative) 가 별도 첨부됩니다 — 시스템별 발주는 *N 시스템 × 5 컴포넌트 중복 구매 + 시스템별 운영 인력 누적* 의 우상향 곡선, AI Native Platform 은 *공통 컴포넌트 1 회 + 시스템별 어댑터 가벼운 추가* 의 완만한 곡선으로 표시. 매트릭스 우측 하단에는 *DORA 2025 결론 인용 박스* 가 부속 자산으로 표시됩니다.

6.1 TCO + Time-to-First-Agent (1·2 축)

6.1.1 TCO 3년 누적 — 중복 구매의 정량 (illustrative)

TCO 3 년 누적 비교는 시스템별 발주가 만드는 *중복 구매의 정량 부담* 을 핵심으로 합니다. 한 시스템에 LLM Serving + Vector DB + MCP Server + Observability + Governance 의 5 컴포넌트가 필요한데, 3 개 시스템을 시스템별로 발주하면 *동일 5 컴포넌트를 3 번 구매* 하는 셈입니다. AI Native Platform 통합 시 5 컴포넌트를 *1 번 구매 + 시스템별 어댑터* 만 추가합니다 [S05, S07].

3 년 누적 시각에서 보면 격차는 더 분명해집니다. 시스템별 발주는 *시스템마다 별도 운영 인력* 도 누적시킵니다. 5 명 인력 × 3 시스템 = 15 명 인력 vs. AI Native Platform 의 5 명 인력 + 시스템별 1 명 = 8 명 인력 (illustrative). 인건비 차이가 *컴포넌트 라이선스 차이* 보다 훨씬 큰 비중을 차지하는 것이 일반적입니다 [S05].

운영 도구 중복 구매도 누적 부담입니다. Observability 의 Datadog 라이선스, LLM Gateway 의 Portkey 구독, 거버넌스 의 Lakera 등 *벤더별 구독료* 가 시스템마다 별도 계약되면 *조직 협상력 분산* 의 추가 손실까지 발생합니다. 통합 발주 시 *대량 협상력* 으로 단가 절감도 가능합니다.

본 TCO 비교의 *illustrative* 수치는 실제 워크로드·라이선스·인건비에 따라 정밀 재계산이 필요합니다. 본 백서는 *3 년 누적 비용 차이의 방향성과 메커니즘* 을 정확히 제시하되, 정확한 금액은 의사결정자 자체 분석을 권장합니다. 한국 공공·기업의 실제 도입 사례별 정량 데이터는 후속 Customer Case Study 백서에서 다룹니다 [S05, S07].

본 1 축의 핵심 메시지는 다음과 같이 정리됩니다. *시스템별 발주는 컴포넌트 N 회 중복 구매 + 운영 인력 N 배 누적 + 협상력 N 등분의 3 중 손실을 만든다. AI Native Platform 통합은 이 3 중 손실을 동시에 해소한다.* 본 메시지는 임원 보고서의 1 쪽 요약으로 직접 활용 가능합니다.

6.1.2 Time-to-First-Agent — 셀프서비스의 시간 단축

Time-to-First-Agent (TTFA) 는 *새 워크로드에 새 Agent 를 만들어 본격 배포까지 가는 시간* 의 정량 지표입니다. 시스템별 발주는 *발주 → 계약 → 구축 → 검증 → 배포* 의 5 단계가 평균 *수 개월* 걸립니다. AI Native Platform 의 셀프서비스 구조는 *기존 컴포넌트 조합 → 배포* 의 2 단계로 *수 일~수 주* 까지 단축 가능합니다 [S07, S08].

이 시간 단축의 메커니즘은 *컴포넌트 재사용* 입니다. AI Native Platform 위에서는 LLM Serving·Vector DB·MCP Server·Observability·Governance 가 모두 *기존 운영 자산* 이므로, 새 Agent 는 *워크로드 정의 + 도구 등록 + 정책 적용* 만 추가하면 됩니다. 발주·계약·구축의 5 단계가 *시민 개발자 셀프서비스* 로 압축됩니다 [S08].

시간 단축의 직접 가치는 *경쟁 우위 확보 시점* 의 단축입니다. *2026년 말까지 40%가 task-specific Agent 통합* 한다는 Gartner 전망 (본 백서 3 장 §3.1.1) 의 시간 압박 안에서, *수 일~수 주* 의 TTFA 는 *수 개월* 의 TTFA 보다 *수 배 더 많은 Agent 를 만들 수 있는* 시간을 확보합니다 [S08, S13].

본 TTFA 단축이 *Scaled 단계 진입* 의 직접적 메커니즘이기도 합니다. 본 백서 7 장에서 다루는 *프로덕션 Agent 20~40+* 임계값은, Agent 한 개당 *수 개월* 의 발주가 누적되면 도달 불가능합니다. 셀프서비스 TTFA *수 일~수 주* 가 임계값 도달의 전제 조건입니다 [S13, S14].

본 2 축의 핵심 메시지는 다음과 같이 정리됩니다. *시스템별 발주의 수 개월 TTFA 는 Scaled 단계 도달을 구조적으로 차단한다. AI Native Platform 의 셀프서비스 TTFA 가 한국 조직의 경쟁 우위 확보 시점을 단축한다.*

본 메시지는 의사결정 보고서의 시간 비용 분석에 직접 활용 가능합니다.

6.2 거버넌스 + 데이터 재사용 (3·4 축)

6.2.1 거버넌스 일관성 — 정책·로그·평가의 일원화

거버넌스 일관성 축은 *정책·로그·평가의 일원화 여부* 를 비교합니다. 시스템별 발주는 시스템마다 별도의 정책·RBAC·감사 로그를 운영하게 만듭니다 (본 백서 3 장 §3.2.1). AI Native Platform 은 OPA + Kyverno 정책 엔진 한 곳, Audit Log 한 곳, Promptfoo + DeepEval 평가 파이프라인 한 곳에서 통합 운영합니다 [S09, S10, S25].

거버넌스 일관성의 직접 가치는 *내부 감사 응대* 의 단순화입니다. 한국 공공·기업의 감사 응대 시각에서, 시스템마다 다른 감사 로그 형식을 통합 분석하는 부담이 *N 시스템 × N 감사 응대* 로 누적되는 구조를 *1 감사 응대* 로 압축할 수 있습니다 [S25, S26].

정책 시행의 일관성도 핵심 가치입니다. 시스템마다 다른 PII 마스킹 정책, 다른 콘텐츠 필터링 룰, 다른 호출 제한 정책이 흩어지면 *동일 사용자에 대해 다른 보호 수준* 이 적용되는 *비일관성* 이 누적됩니다 (본 백서 3 장 §3.2.1). 통합 Policy Engine 의 *룰 한 번 정의 → 모든 Agent 즉시 적용* 이 이 비일관성을 해소합니다 [S25].

평가 파이프라인의 일관성은 *PoC → 본격 배포 전환* 의 직접적 메커니즘입니다. 시스템별로 별도 평가 기준이 적용되면 *조직 차원의 품질 표준* 이 부재합니다. 통합 평가 파이프라인은 *조직 표준 품질 임계값* 을 한 곳에서 운영하고, 모든 Agent 가 그 임계값을 통과해야 본격 배포 진입 가능한 게이트 역할을 합니다 [S10, S25].

본 3 축의 핵심 메시지는 다음과 같이 정리됩니다. *시스템별 발주는 거버넌스의 N 회 중복 운영 + N 회 감사 응대 + 정책 비일관성의 3 중 부담을 만든다. AI Native Platform 의 통합 거버넌스는 단일 정책 + 단일 감사 응대 + 일관 정책 시행의 3 중 가치를 만든다.*

6.2.2 데이터 재사용 — Vector·Knowledge Repo 의 공통화

데이터 재사용 축은 *데이터 자산의 폐쇄 vs 공통화* 를 비교합니다. 시스템별 발주는 데이터를 시스템 내부에 폐쇄적으로 가둡니다 (본 백서 3 장 §3.2.2). 그룹웨어 AI 가 자기 Vector DB 를 가지고, 홈페이지 AI·민원 AI·HR AI·콜센터 AI·사내 위키 검색 AI 가 각각 또 다른 Vector DB 를 가지는 식입니다. AI Native Platform 은 공통 Vector DB + 공통 Knowledge Repo 로 *조직 전체* 가 데이터를 활용합니다 [S22, S23].

데이터 공통화의 직접 가치는 *재 PoC 비용의 해소* 입니다. 새 Agent 를 만들 때마다 임베딩·인덱싱·정합성 검증을 다시 수행하는 부담이 사라집니다. *조직 공통 임베딩 자산* 이 한 번 갖춰지면 새 Agent 는 *검색 쿼리만 정의* 하면 즉시 활용 가능합니다 [S22].

데이터 자산의 *복리 효과* 도 핵심 가치입니다. 시스템별 폐쇄 데이터는 각 시스템의 폐쇄 영역에서 *복리 없이 0 으로 수렴* 합니다. 공통화 데이터는 N 개 Agent 가 모두 활용하므로 *복리 효과로 자산 가치가 누적* 됩니다. 이는 본 백서 12 장의 결론 자산화에서 *조직 차원의 누적 가치* 로 설명됩니다 [S22, S23].

조직 차원의 데이터 정합성도 향상됩니다. 시스템별 폐쇄 데이터는 *어느 시스템의 데이터가 진실인가* 의 충돌이 빈번하게 발생합니다. 공통화 데이터는 *조직 단일 진실 (Single Source of Truth)* 로 정합성이 유지됩니다. 한국 공공·기업의 *데이터 거버넌스* 요건과도 정합합니다 [S26, S27].

본 4 축의 핵심 메시지는 다음과 같이 정리됩니다. *시스템별 발주는 데이터 자산의 N 회 폐쇄 + 재 PoC 비용 N 배 + 데이터 충돌의 3 중 손실을 만든다. AI Native Platform 의 공통화 데이터는 자산 복리 효과 + 재 PoC 비용 0 + 단일 진실의 3 중 가치를 만든다.*

6.3 모델 교체 비용 + 망분리 적합성 (5·6 축)

6.3.1 모델 교체 비용 — LLM Gateway 1점 교체

모델 교체 비용 축은 *시스템별 코드 수정 vs Gateway 1점 교체* 의 비교입니다. 시스템별 발주는 시스템마다 LLM 호출 코드가 분산되어 있으므로, GPT-4 → Claude → Gemini → 한국어 LLM 의 교체가 *시스템 수만큼의 코드 수정* 을 요구합니다. AI Native Platform 의 LLM Gateway 는 *한 곳에서 모델 라우팅 설정만 변경* 하면 모든 Agent 가 자동으로 새 모델을 사용합니다 [S04, S22].

모델 교체 비용의 핵심 가치는 *벤더 락인 방어* 입니다. 단일 벤더 (예: OpenAI) 에 모든 시스템이 코드 수준 종속되면, 벤더의 가격 인상이나 정책 변경에 *조직 차원의 대응 불가* 가 됩니다. LLM Gateway 의 단일 추상화는 *벤더 의존성을 코드 수준에서 분리* 하여 협상력을 유지합니다 [S04, S22].

한국어 LLM 의 빠른 발전을 고려하면 이 5 축의 중요성은 더 커집니다. 2026년 한국어 LLM 의 품질은 글로벌 모델과 빠르게 격차를 좁히고 있으며, *국산 LLM 채택* 의 정책 압박도 누적되고 있습니다 [S26]. 시스템별 코드 수정으로는 이 빠른 교체에 대응 불가합니다.

모델 교체의 *시간 비용* 도 정량적 차이를 만듭니다. Gateway 설정 변경은 *분 단위 운영 작업* 이지만, 시스템별 코드 수정은 *수 주의 개발 + 검증 + 배포* 의 사이클이 필요합니다. 시간 비용이 *시스템 수만큼 곱해지는* 시스템별 발주는, 본격 배포 단계의 교체 의사결정 자체를 *불가능에 가까운* 부담으로 만듭니다 [S04].

본 5 축의 핵심 메시지는 다음과 같이 정리됩니다. *시스템별 발주는 벤더 락인 + 시스템별 교체 비용 + 빠른 모델 시장 대응 불가의 3 중 손실을 만든다. AI Native Platform 의 LLM Gateway 는 벤더 락인 방어 + 1점 교체 + 빠른 대응의 3 중 가치를 만든다.*

6.3.2 망분리 적합성 — 점검 포인트 다수 vs 중앙 통제

망분리 적합성 축은 한국 공공·기업의 *우회할 수 없는 전제* 입니다. 시스템별 발주는 시스템마다 별도의 망분리 점검이 필요합니다. 즉 *N 시스템 × N 망분리 점검* 의 부담이 누적됩니다. AI Native Platform 은 *중앙 통제점 1 개* 로 망분리 점검이 통합됩니다 [S26].

망분리 점검의 통합화 가치는 *내부 인증 응대* 의 단순화입니다. 한국 공공·기업의 정보보호 인증 (ISMS-P 등) 응대 시, 시스템별 점검 포인트가 흩어지면 *각 시스템마다 동일 점검을 반복* 하는 부담이 누적됩니다. 통합 플랫폼은 *1 회 점검* 으로 인증 응대가 종료됩니다.

K8s 컨테이너의 *망분리 패키지 동치성* (본 백서 2 장 §2.3.1) 이 이 6 축의 핵심 메커니즘입니다. AI Native Platform 의 13 컴포넌트가 모두 컨테이너 이미지로 패키지되어 있으므로, *Container Registry 1 개 + 자체 K8s 1 개* 만 갖춰지면 망분리 환경에서의 운영이 가능합니다. 별도 컴포넌트마다 망분리 적합성을 다시 검증할 필요가 없습니다 [S01, S22].

망분리 환경의 보안 통제도 통합 점에서 운영됩니다. Container Registry 의 이미지 스캔, K8s 의 admission control, OPA 의 정책 시행, Audit Log 의 통합 분석 — 모두 *단일 K8s 운영 체계* 안에서 통합 운영됩니다. 시스템별 발주에서는 이 4 가지 통제가 *시스템마다 별도 도구 + 별도 운영* 으로 흩어집니다 [S25, S26].

본 6 축의 핵심 메시지는 다음과 같이 정리됩니다. *시스템별 발주는 망분리 점검 N 회 + 인증 응대 N 회 + 보안 통제 N 종 분산의 3 중 부담을 만든다. AI Native Platform 의 K8s 위 통합 운영은 점검 1 회 + 인증 1 회 + 보안 통제 통합점의 3 중 가치를 만든다.*

본 6 장의 결론은 한 줄로 정리됩니다. *6 축 전체에서 AI Native Platform 이 시스템별 발주 대비 우위입니다. DORA 2025 의 결론이 한국 공공·기업에서 그대로 작동하며, 시스템별 발주는 더 이상 합리적 선택지가 아닙니다* [S05, S07, S26]. 다음 7 장에서 *AI 성숙도의 새 지표* (Microsoft / Salesforce / nexocode / AgentMarketCap 의 4 프레임워크 합의) 를 통해 한국 조직의 *현재 위치 + 목표 위치 격차* 를 정량 진단합니다.

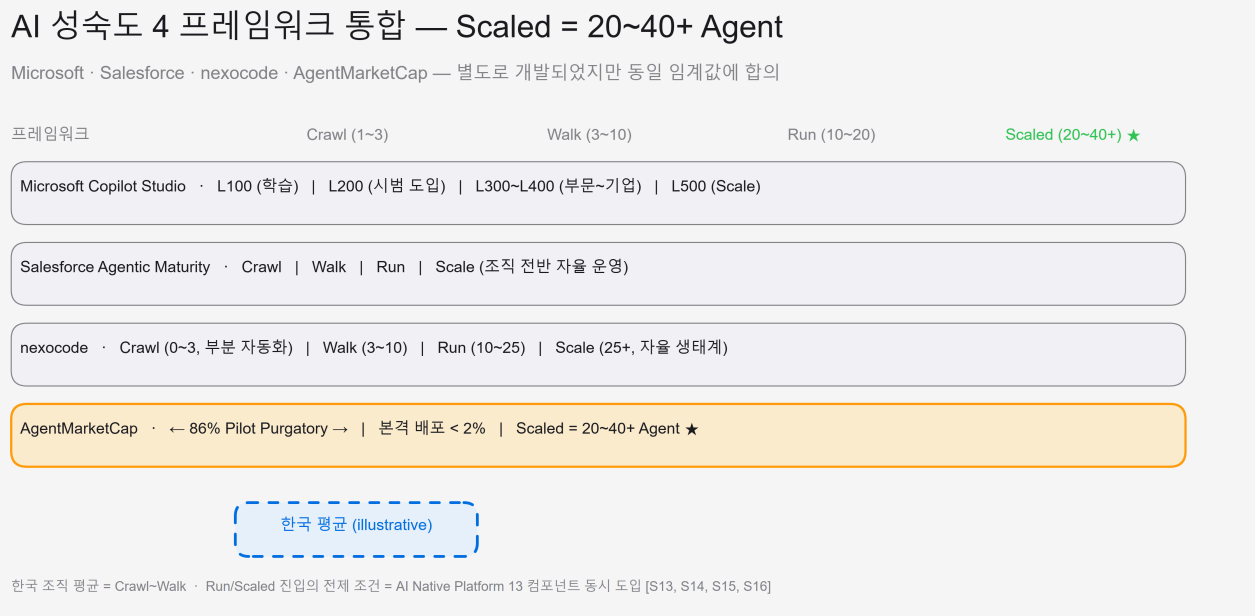
7장. AI 성숙도의 새 지표 — 20~40+ Agent = Scaled 단계

우리 조직의 AI 성숙도는 어디인가 는 한국 IT 의사결정자가 가장 자주 묻는 질문 중 하나입니다. 본 장은 글로벌 4 프레임워크가 합의한 *Scaled = 프로덕션 20~40+ Agent* 임계값으로 답을 정렬하고, 한국 조직 평균을 illustrative 로 위치시킵니다.

4 프레임워크는 다음과 같이 정리됩니다. Microsoft Copilot Studio 의 L100~L500 [S14], Salesforce 의 Agentic Maturity 4 단계 (Crawl/Walk/Run/Scale) [S16], nexocode 의 Crawl/Walk/Run/Scale [S15], AgentMarketCap 의 Pilot/Production/Scaled 정량 모델 [S13]. 4 프레임워크가 *별도로 개발되었음에도 동일한 임계값에 합의* 한 정황이 가장 강력한 산업 합의입니다.

본 장의 결론은 분명합니다. *프로덕션 Agent 20~40+ = Scaled 단계* 는 AI Native Platform 의 13 컴포넌트가 *동시에 운영* 되어야 도달 가능합니다. 즉 성숙도 임계값 도달의 *인프라 전제 조건* 이 AI Native Platform 입니다. 본 백서 5 장 (매트릭스) + 6 장 (6 축 비교) 의 결론을 *성숙도 측면* 에서 강화하는 역할을 본 7 장이 담당합니다.

본 장의 시각 자산은 4 프레임워크 통합 매트릭스 + 한국 조직 평균 위치 (`[FIGURE: maturity-4framework-matrix]`) 입니다.



캡션: 4 프레임워크 통합 매트릭스 — Microsoft L100~L500 / Salesforce Crawl-Walk-Run-Scale / nexocode / AgentMarketCap 정합

매트릭스의 행은 4 프레임워크 (Microsoft, Salesforce, nexocode, AgentMarketCap) 로 정렬됩니다. 열은 4 단계 (Crawl/L100, Walk/L200, Run/L300~400, Scaled/L500) 로 정렬됩니다. 각 셀에는 *각 프레임워크가 정의한 단계 이름 + 프로덕션 Agent 수 임계값 + 평가 척도* 가 표기됩니다. 가장 우측 열에는 *통합 합의* — Pilot Purgatory (Crawl/Walk), 본격 배포 진입 (Run), Scaled (20~40+ Agent) — 가 정렬됩니다. 다이어그램의 하단에는 *한국 조직 평균 위치 (illustrative)* 가 별도 박스로 표시됩니다 — 대부분 Crawl ~ Walk

단계 (1~5 Agent) 에 위치하며, AI Native Platform 도입이 Run/Scaled 진입의 전제 조건임을 명시. 매트릭스 우상단에는 *AgentMarketCap 의 86% Pilot Purgatory 정량* 인용 박스가 부속 자산으로 표시됩니다.

7.1 4 프레임워크 통합 비교

7.1.1 Microsoft L100~L500 + Salesforce Crawl/Walk/Run/Scale

Microsoft Copilot Studio 의 성숙도 모델은 *L100, L200, L300, L400, L500* 의 5 단계로 정의됩니다. L100 은 *학습 단계* (개별 Copilot 사용), L200 은 *시범 도입* (한정 부서 도입), L300 은 *부문 확장* (여러 부서 Copilot 운영), L400 은 *기업 전반* (조직 차원 통합), L500 은 *Scale 단계* (자율 Agent 운영 + 표준화) 입니다 [S14].

Microsoft 의 L500 정의가 본 백서의 *Scaled* 와 정합합니다. L500 의 명시적 임계값은 *프로덕션 Agent 다수 운영 + 표준화 + 거버넌스 자동화* 의 3 가지이고, 정량적으로는 *20~40+ Agent* 가 일반적입니다. 한국에서 Microsoft Copilot Studio 도입 사례 중 L300 이상으로 진입한 조직은 *공공·기업 상위 10%* 정도로 추정됩니다 [S14].

Salesforce 의 Agentic Maturity 모델은 *Crawl, Walk, Run, Scale* 의 4 단계입니다. Crawl 은 *학습 + PoC* (1~3 Agent), Walk 은 *시범 도입* (3~10 Agent), Run 은 *부문 확장* (10~20 Agent), Scale 은 *조직 전반 자율 운영* (20+ Agent) 입니다 [S16].

Salesforce 의 Scale 정의도 Microsoft L500 과 사실상 동일한 임계값입니다. *20+ Agent 의 조직 전반 자율 운영* 이라는 정량과 정성의 결합이 두 프레임워크 합의 지점입니다. Salesforce 가 *Crawl/Walk/Run/Scale* 의 직관적 단어를 채택한 점이 한국 의사결정자에게 *진입·진행·확장·도달* 의 단계를 mental model 로 전달하기에 효과적입니다 [S16].

본 두 프레임워크의 결합은 한국 RFP 어휘에 즉시 적용 가능합니다. *L300 이상 진입 목표* 또는 *Run/Scale 단계 진입 목표* 가 RFP 의 *현실적 목표 단계* 표현으로 활용 가능합니다. 본격 배포 진입 후 1~2 년 안에 *Run* 단계 도달 + 3~5 년 안에 *Scale* 단계 도달이 한국 공공·기업의 일반적 목표 시나리오입니다 [S14, S16].

7.1.2 nexocode Crawl/Walk/Run/Scale + AgentMarketCap Pilot/Production

nexocode 의 Agentic AI Maturity 모델은 Salesforce 와 동일한 *Crawl/Walk/Run/Scale* 의 4 단계를 채택합니다. nexocode 가 강조하는 차별점은 *각 단계의 정량 임계값을 더 구체적으로 명시* 한 점입니다. Crawl 은 *0~3 Agent + 부분 자동화*, Walk 은 *3~10 Agent + 단일 워크로드 자동화*, Run 은 *10~25 Agent + 다 워크로드 자동화*, Scale 은 *25+ Agent + 자율 생태계* 입니다 [S15].

nexocode 의 *25+ Agent = Scale* 임계값이 Microsoft L500 의 *20~40+* 와 사실상 동일한 범위 안에 있습니다. 임계값의 정확한 숫자는 프레임워크마다 미세하게 다르지만 *20~40 의 범위* 에서 합의되는 패턴입니다 [S14, S15, S16].

AgentMarketCap 의 *Pilot/Production/Scaled* 정량 모델은 *프로덕션 진입 비율* 의 측면에서 가장 명확한 기준을 제공합니다. *Pilot Purgatory 86%* 라는 정량 데이터가 산업 표준 인용으로 자리잡았고, *프로덕션 진입 2% 미만 + Scaled = 20~40+ Agent* 의 두 임계값이 본 백서 3 장 §3.1.2 에서도 핵심 인용되었습니다 [S13].

4 프레임워크 모두 *Scaled = 20~40+ Agent* 라는 동일 임계값에 합의한 정황은, 이 임계값이 *우연한 숫자가 아니라 조직 차원 자율 운영의 구조적 임계값* 임을 시사합니다. 즉 20~40 Agent 미만에서는 시스템별 별도 운영이 가능하지만, 그 이상에서는 *플랫폼 차원의 통합 운영* 이 필수가 된다는 뜻입니다 [S13, S14, S15, S16].

본 4 프레임워크 합의는 한국 의사결정자에게 *벤더 중립 안전성* 을 제공합니다. *어느 프레임워크를 채택해도 동일 목표 임계값* 이 도출되므로, 특정 벤더에 종속되지 않은 *조직 자체 목표 설정* 이 가능합니다. 4 프레임워크의 *합집합 정의* 가 한국 RFP 의 *Scaled 단계 정의* 로 활용 가능한 표준입니다.

7.2 Scaled 임계값 — 20~40+ Agent 의 의미

7.2.1 20~40+ Agent 의 정량 근거 — 자산화·재사용·운영 인력

20~40+ Agent 임계값의 정량 근거는 *조직 단위 자산화의 임계 질량* 에 있습니다. 1~5 Agent 만 운영하면 각 Agent 가 *별도 인력 + 별도 검증* 으로 운영 가능하지만, 20+ Agent 가 되면 *Agent 라이프사이클 표준화 + Tool Registry 공통화 + Eval Pipeline 자동화* 가 *필수* 가 됩니다 [S10, S13].

자산화의 임계 질량을 정량으로 분해하면 다음과 같습니다. Agent 한 개당 *코드 자산 + 도구 통합 + 메모리 + 평가 룰* 의 4 종 자산이 필요한데, 20 개 Agent 면 *80 자산* 이 누적됩니다. 이 80 자산을 *각 시스템 폐쇄로* 운영하면 인력 부담이 N 배 증가하지만, *플랫폼 공통화* 하면 *재사용 가능 자산 80 개 + 신규 Agent 추가 비용 거의 0* 의 구조가 됩니다 [S10].

운영 인력 측면에서도 임계값이 분명합니다. 시스템별 폐쇄 운영의 경우, Agent 5 개 당 *전담 인력 1 명* 이 일반적입니다. 20 Agent = 4 명, 40 Agent = 8 명이 *시스템별 운영 부담* 의 정량입니다. 플랫폼 운영 시 *플랫폼 운영 인력 5 명 + Agent 추가 비용 거의 0* 의 구조로 *고정 인력 + 가변 추가 0* 의 효율적 운영이 가능합니다 [S07, S13].

재사용 가능 자산의 *복리 효과* 가 임계값의 핵심 메커니즘입니다. 첫 번째 Agent 의 도구 통합이 *80% 재사용 가능* 으로 누적되면, 두 번째 Agent 는 *20% 신규 + 80% 재사용*, 세 번째 Agent 는 *15% 신규 + 85% 재사용* 의 식으로 *신규 Agent 당 추가 비용이 점점 줄어드는* 복리 곡선이 만들어집니다. 이 곡선의 변곡점이 *대략 20 Agent* 부근입니다 [S10, S22].

거버넌스 측면에서도 임계 질량은 명확합니다. 5 Agent 까지는 *사람이 직접 정책 검토* 가 가능하지만, 20+ Agent 가 되면 *자동 정책 시행 (OPA, Kyverno) + 자동 평가 (Promptfoo, DeepEval)* 가 *필수* 가 됩니다. 자동화 부재 시 거버넌스 부담이 *조직 차원의 본격 배포 차단점* 이 되기 때문입니다 [S25].

7.2.2 20~40+ 미달 단계 — Pilot Purgatory 의 정의

20~40+ Agent 미달 단계 는 *Pilot Purgatory* 로 정의됩니다 [S13]. Crawl 단계 (1~3 Agent), Walk 단계 (3~10 Agent) 가 모두 이 정체 상태에 포함됩니다. AgentMarketCap 의 *86% Pilot Purgatory* 정량이 이 단계 정체의 산업 현실을 보여줍니다.

Pilot Purgatory 의 구조적 특징은 다음과 같이 정리됩니다. 첫째, *각 Agent 의 PoC 는 통과* 하지만 *조직 차원 통합* 이 부재합니다. 둘째, *데이터·도구·정책이 시스템별 폐쇄* 되어 *재사용 불가* 합니다. 셋째, *본격 배포 진입 시 거버넌스·감사·평가의 부족이 일제히 드러나* PoC 단계로 회귀합니다 [S13].

한국 조직의 평균 위치가 이 *Pilot Purgatory* 단계에 있을 가능성이 큽니다 (illustrative 추정). 그룹웨어 챗봇 PoC, 홈페이지 챗봇 PoC, MIS 챗봇 PoC, 민원 챗봇 PoC, HR 챗봇 PoC, 콜센터 AI PoC, 사내 위키 검색 AI PoC 등이 시스템별로 진행되었지만 본격 배포로 진입한 사례는 *제한적* 인 한국 시장 현실이 이 추정의 근거입니다 [S13, S26].

Pilot Purgatory 에 머무는 *기회 비용* 은 정량적으로 큽니다. 시스템별 PoC 가 *3 년간 누적* 되어도 본격 배포 진입이 없다면, *3 년의 인력·예산이 자산화 0 으로 소진* 된 셈입니다. 본 백서 6 장의 6 축 비교에서 분석한 *3 년 누적 TCO 차이* 가 이 기회 비용의 직접적 정량화입니다 [S05, S07].

Pilot Purgatory 탈출의 단일 경로는 분명합니다. *AI Native Platform 의 13 컴포넌트 동시 도입* 이 그것입니다. 4 원인 (거버넌스 사일로, 관찰가능성 부재, 데이터 폐쇄, 컴포넌트 재사용 불가) 을 *동시에* 해소해야 본격 배포 진입이 가능하기 때문입니다 (본 백서 3 장 §3.2). 단일 컴포넌트만 추가하는 *부분 해소* 로는 정체 탈출이 어렵습니다 [S09, S13, S22].

7.3 한국 조직 평균 위치 — illustrative 추정과 함의

7.3.1 한국 평균 위치 illustrative — Crawl~Walk 단계

한국 공공·기업·대기업의 *평균 프로덕션 Agent 수* 는 illustrative 추정으로 *1~5 Agent* 수준입니다. 본 추정은 직접 통계가 아니라 *시스템별 PoC 반복 패턴* (본 백서 3 장 §3.3.1) + *Pilot Purgatory 글로벌 평균 86%* [S13] + *한국 시장 일반적 도입 정체* 의 종합 추정입니다.

이 위치는 4 프레임워크 통합 매트릭스에서 *Crawl ~ Walk* 단계에 해당합니다. Microsoft L100~L200, Salesforce Crawl/Walk, nexocode Crawl/Walk, AgentMarketCap Pilot 단계입니다. *Scaled (20~40+ Agent)* 까지의 격차는 *수십 배* 입니다 [S13, S14, S15, S16].

한국 상위 10% 조직의 위치는 *Walk ~ Run* 단계로 추정됩니다. 일부 대기업의 사내 AI Agent 도입 사례, 공공기관의 민원·법령 검색 RAG 챗봇 도입 사례, 일부 기업의 콜센터·HR·법무 검토 AI 본격 배포 사례가 이 단계에 일부 진입했습니다. 그러나 *조직 전반 자율 운영* 의 Scaled 단계 도달 사례는 *극히 제한적* 입니다 [S26].

한국 평균 위치의 *구조적 원인* 은 시스템별 발주 패턴입니다. 그룹웨어 챗봇, 홈페이지 챗봇, MIS 챗봇, 민원 챗봇, HR 챗봇, 법무 검토 AI, 콜센터 AI, 사내 위키 검색 AI 가 별도 RFP 로 발주되는 구조에서는 *조직 차원의 Agent 누적* 이 어렵습니다. 시스템마다 *별도 Agent 1~2 개* 가 누적되어 *조직 총합 5~10 개* 가 되더라도, 시스템별로 흩어진 상태에서는 *Scaled 단계 자율 운영* 의 본질이 부재합니다 [S13, S26].

본 illustrative 추정은 본 백서 후속 Customer Case Study 백서에서 정밀 정량 데이터로 보강될 예정입니다. 한국 공공·기업의 *실제 도입 사례별 정량 Agent 수* 를 누적 분석하여 *industry-specific maturity benchmark* 를 제공할 계획입니다 [S13, S26, S27].

7.3.2 Run/Scaled 진입의 전제 조건 — AI Native Platform 도입

한국 조직이 *Run 단계 진입* 을 목표로 한다면 *AI Native Platform 13 컴포넌트의 70% 이상 동시 도입* 이 전제 조건입니다. K8s 기반 위에 LLM Serving + Vector DB + MCP + LLM Gateway + Observability + Governance 의 6 컴포넌트가 최소 갖춰져야 *10~20 Agent 의 통합 운영* 이 가능합니다 [S09, S22].

Scaled 단계 진입 을 목표로 한다면 *13 컴포넌트 전부 + Tool Registry + Agent Fabric + Eval Pipeline 자동화* 가 필수입니다. 사람이 정책·평가·라이프사이클을 수동 관리하는 구조로는 *20~40+ Agent* 의 동시 운영이 불가능하기 때문입니다 [S10, S13].

본 백서 부록 A 의 RFP 체크리스트는 이 *Run/Scaled 진입 전제 조건* 의 직접적 RFP 형식입니다. 한국 의사결정자가 *목표 단계 = Run 또는 Scaled* 를 결정한 뒤, 그 단계에 필요한 컴포넌트를 부록 A 에서 발췌해 RFP 항목으로 옮기는 활용 패턴이 가능합니다 [S22, S23].

시간 비용 측면에서, *Run 단계 진입* 의 평균 목표 시점은 *AI Native Platform 도입 후 1~2 년* 입니다 (illustrative). *Scaled 단계 진입* 은 *3~5 년* 이 일반적입니다. 시스템별 발주의 *수 개월 TTFA* 와 비교하면 *수십 배의 Agent 누적 속도* 차이가 시간 비용으로 직접 환산됩니다 [S07, S13].

본 7 장의 결론은 한 줄로 정리됩니다. *한국 조직의 평균 위치 (Crawl~Walk) 와 목표 위치 (Run/Scaled) 의 격차는 시스템별 발주로 메울 수 없으며, AI Native Platform 의 13 컴포넌트 동시 도입이 단일 경로입니다.* 다음 8 장에서 오픈소스 AI Native Platform 의 좌표계 (5 사분면) 를 통해 *어떤 도구를 어떤 순서로 도입할 것인가* 의 의사결정 가이드를 제공합니다 [S13, S22, S23].

8장. 오픈소스 AI Native Platform 좌표계 — 5사분면

한국 RFP 가 *오픈소스 LLM 플랫폼* 을 요구할 때 의사결정자가 가장 어려워하는 것은 *어떤 도구의 좌표가 어디인가* 입니다. LangGraph 와 CrewAI 의 차이, Dify 와 n8n 의 위치, vLLM 과 Ollama 의 의사결정 — 모두 1 쪽 도식 없이는 비교가 어렵습니다. 본 장의 5 사분면 좌표계는 LangGraph 부터 MCP Servers 까지를 *한 장의 그림* 으로 정렬합니다 [S21, S22, S23].

5 사분면은 다음과 같이 구성됩니다. 1 사분면 *코드 중심 (Code-first)* — LangGraph, CrewAI, AutoGen, OpenAgents. 2 사분면 *노코드 / 시각 워크플로우 (Visual Workflow)* — Dify, Flowise, n8n. 3 사분면 *LLM Serving* — vLLM, Ollama, Triton, TGI. 4 사분면 *벡터 / 임베딩 (Vector)* — Milvus, Qdrant, Weaviate, PGVector. 5 사분면 *MCP Server 생태계* — GitHub, Slack, Postgres, AWS, Atlassian, Notion 의 공식 MCP Server [S22, S23, S24].

5 사분면 좌표계의 가치는 *의사결정 트리* 의 단순화에 있습니다. 한국 의사결정자가 *워크로드 + 인력 + 망분리* 의 3 질문에 답하면 5 사분면의 *시작점 사분면* 이 자동 결정되는 구조입니다. 4·5 장에서 정의한 13 컴포넌트 매트릭스가 *영역 단위* 의 의사결정이라면, 본 8 장의 좌표계는 *도구 단위* 의 의사결정입니다.

본 장의 시각 자산은 5사분면 좌표 다이어그램 (`[FIGURE: oss-ainp-5quadrant]`) 입니다.

캡션: 오픈소스 AI Native Platform 5사분면 좌표계

다이어그램의 중앙에 *AI Native Platform* 의 중심점이 배치되고, 5 방향으로 사분면이 그려집니다. 1 사분면 (좌상단) — 코드 중심: LangGraph (상태/그래프) + CrewAI (역할 팀) + AutoGen (대화형) + OpenAgents 가 도구 아이콘으로 배치됩니다. 2 사분면 (우상단) — 노코드 / 시각 워크플로우: Dify (LLMOps) + n8n (action layer, 150k+ stars) + Flowise. 3 사분면 (우하단) — LLM Serving: vLLM (KV 캐시·PagedAttention) + Ollama (경량) + Triton (NVIDIA 공식) + TGI. 4 사분면 (좌하단) — 벡터 /

임베딩: Milvus (분산) + Qdrant (Rust 성능) + Weaviate (GraphQL) + PGVector (PostgreSQL 자산 활용). 5 사분면 (중앙 가로지름) — MCP Server: 4 사분면 모두를 가로지르는 표준 인터페이스로 표시. 다이어그램 하단에는 *3 질문 의사결정 트리* (워크로드 패턴 / 인력 스킬 / 망분리 요건) 가 부속 자산으로 표시되어, 의사결정자가 5 사분면 중 시작점을 결정 가능하게 안내합니다.

8.1 코드 중심 사분면 — LangGraph / CrewAI / AutoGen

8.1.1 LangGraph + CrewAI — 상태/그래프 vs 역할 팀

LangGraph 는 *상태 머신 + 그래프 흐름* 패턴의 대표입니다. LangChain 의 후속 프로젝트로 시작되었으며, 2026년 현재 LLM Agent 프레임워크 중 *가장 빠르게 표준화* 된 도구로 자리잡았습니다. 핵심 추상화는 *State* (현재 단계 추적), *Node* (실행 단위), *Edge* (전이 조건) 의 3 가지이고, 복잡한 분기·재시도·인간 개입 흐름을 그래프로 표현 가능합니다 [S21, S22].

LangGraph 의 적합 워크로드는 *복잡한 분기·재시도가 필요한 흐름* 입니다. 예를 들어 인허가 심사에서 *조건 A 면 검증 단계로, 조건 B 면 보완 요청 단계로* 같은 분기 흐름을 명확히 그래프로 표현 가능합니다. 한국 공공·기업의 컴플라이언스 점검, 인허가·자격 심사 보조, 계약·법무 검토, 산업 안전 점검, 정책 영향 분석 같은 워크로드에 적합합니다 [S22].

LangGraph 의 한계는 *학습 곡선* 입니다. State·Node·Edge 의 추상화를 익히는 데 *2~4 주* 가 일반적이고, *복잡 워크로드 설계 능력* 은 더 긴 경험이 필요합니다. 다만 LangChain Python 친숙도가 있는 인력은 *기존 LangChain 스킬을 재활용* 할 수 있어 진입 부담이 일부 완화됩니다 [S22].

CrewAI 는 *역할 기반 Multi-agent* 패턴의 대표입니다. Agent 들이 *역할 (Role)* 과 *목표 (Goal)*, *역할 별 도구 (Tools)* 를 가진 *팀* 으로 구성됩니다. 컨설팅 보고서 작성, 시장 분석 같은 *여러 역할의 협업이 명확한* 워크로드에 직관적입니다 [S22].

CrewAI 의 강점은 *진입 용이성* 입니다. *역할 ↔ Agent* 의 직관적 mental model 이 *PoC 단계의 빠른 구축* 을 가능하게 합니다. 한국 시민 개발자·비개발자가 *AI Agent 의 첫 경험* 으로 CrewAI 를 채택하는 패턴이 누적되고 있습니다. 다만 *복잡한 분기·재시도* 가 필요한 워크로드에서는 LangGraph 가 우위입니다 [S22].

LangGraph 와 CrewAI 의 *동시 운영* 도 가능합니다. AI Native Platform 의 *Composable* 정의가 이를 보장하기 때문입니다. 복잡한 워크로드는 LangGraph, 직관적 협업 워크로드는 CrewAI 의 *워크로드 기반 분기 운영* 이 본격 배포 단계의 일반적 패턴입니다 [S22, S23].

8.1.2 AutoGen + OpenAgents — 대화형 멀티 에이전트

AutoGen 은 *대화형 Multi-agent* 패턴의 대표입니다. Microsoft Research 에서 시작된 프로젝트로, *Conversation Loop* 안에서 Agent 들이 *자유로운 대화* 를 통해 협업합니다. 핵심 추상화는 *ConversableAgent* (대화 가능한 Agent) 와 *GroupChat* (그룹 대화 관리) 의 2 가지입니다 [S21, S22].

AutoGen 의 적합 워크로드는 *자유로운 대화형 협업* 입니다. 고객 응대, 내부 협업, 학습 지원 같은 *흐름이 사전 정의되지 않은* 워크로드에서 강점입니다. 한국 공공·기업의 직원 대상 학습 지원 챗봇, 시민 대상 민원 응대 챗봇 같은 사용 사례에 적합합니다 [S22].

AutoGen 의 한계는 *예측 가능성* 입니다. 대화형 흐름은 *결정론적이지 않으므로* 컴플라이언스 점검 같은 *결과의 정확성이 중요한* 워크로드에는 LangGraph 의 명시적 그래프가 우위입니다. 대화형은 *유연성* 과 *예측 가능성* 의 trade-off 안에 있습니다 [S21].

OpenAgents 는 다중 Agent 의 *오픈 플랫폼* 을 표방하는 프로젝트입니다. 학술 연구 기원의 프레임워크로, *Agent 간 마켓플레이스 + 자율 협업 + 외부 도구 통합* 의 3 가지를 강조합니다. 산업 도입 비율은 LangGraph·CrewAI·AutoGen 보다 낮지만, *학술·연구* 영역에서 빠르게 누적되고 있습니다 [S21].

본 3+1 코드 중심 사분면의 의사결정은 *워크로드 패턴 + 결과 정확성 요구도* 의 2 축으로 정리됩니다. 결정론적·정확성 우선 = LangGraph. 역할 협업 직관성 = CrewAI. 유연 대화 = AutoGen. 학술·연구 = OpenAgents. 한국 시장에서는 LangGraph + CrewAI 의 2 종 동시 운영이 본격 배포 단계의 일반적 패턴입니다 [S21, S22].

8.2 노코드 사분면 — Dify / n8n / Flowise

8.2.1 Dify — LLMOps 노코드 표준

Dify 는 *LLMOps 노코드* 의 빠르게 표준화된 도구입니다. 시민 개발자와 IT 협업의 결합 모델을 가능하게 합니다. 핵심 기능은 *프롬프트 관리, RAG 파이프라인 구성, Agent 빌더, 데이터셋 관리, 평가 + Observability* 의 5 가지가 단일 GUI 에서 통합 제공됩니다 [S22].

Dify 의 강점은 *프로토타입 → 프로덕션 전이 가능성* 입니다. 노코드 GUI 로 만든 Agent 가 *API 엔드포인트* 형태로 즉시 외부 시스템에서 호출 가능합니다. 즉 *PoC 단계의 빠른 검증 + 본격 배포의 통합 운영* 이 같은 도구 안에서 가능한 구조입니다. 셀프 호스팅 (Docker / K8s Helm Chart) 도 지원합니다 [S22, S23].

Dify 의 적합 환경은 *시민 개발자 + IT 운영 인력의 협업* 입니다. 시민 개발자가 GUI 로 *워크로드 정의* 하면, IT 운영 인력이 *데이터 소스 연결 + 모델 라우팅 + 정책 적용* 을 백엔드로 처리하는 패턴입니다. 한국 공공기관과 기업의 *비개발 인력 AI 활용* 워크플로우에 적합합니다 [S22].

망분리 환경에서의 Dify 운영도 가능합니다. K8s Helm Chart 로 배포 가능하고, *모든 의존성이 컨테이너 이미지에 포함* 되어 있어 *컨테이너 = 망분리 패키지* 의 동치성을 그대로 활용합니다 (본 백서 2 장 §2.3.1). 한국 공공·기업의 도입 사례가 빠르게 누적되고 있습니다.

Dify 의 한계는 *복잡 워크로드의 표현력* 입니다. 노코드 GUI 의 추상화 한계 때문에 *복잡 분기·재시도·다단계 협업* 은 LangGraph 같은 코드 중심 도구가 우위입니다. *Dify 로 PoC → 복잡도 증가 시 LangGraph 로 마이그레이션* 같은 단계적 이행 패턴이 일반적입니다 [S22, S23].

8.2.2 n8n + Flowise — 시각 워크플로우의 두 패턴

n8n 은 *action layer* 의 시각 워크플로우 도구로, *GitHub 150k+ stars* 의 압도적 커뮤니티 규모를 보유합니다. 본래 IT 자동화 (Zapier 형식) 에서 시작되었지만 2024~2025년부터 *AI Agent 워크플로우 도구* 로 빠르게 확장되었습니다. *기존 RPA 자동화 + AI Agent 통합* 의 핵심 결합점 역할을 합니다 [S22, S23].

n8n 의 적합 환경은 *기존 워크플로우 자동화에 AI 추가* 입니다. 예를 들어 *이메일 수신 → AI 분류 → CRM 등록 → Slack 알림* 같은 다단계 워크플로우를 GUI 로 정의 가능합니다. 한국 공공·기업의 RPA 도입 자산을 그대로

AI 시대로 확장하는 경로가 됩니다 [S23].

n8n 의 셀프 호스팅도 표준입니다. Docker / K8s 배포가 표준이고, 망분리 환경 운영도 가능합니다. 오픈소스 라이선스 (Sustainable Use License) 가 *상업 활용 가능* 하지만 *호스팅 서비스 재판매는 제약* 이 있는 점은 주의가 필요합니다 [S23].

Flowise 는 *LangChain 기반 노코드* 도구의 대표입니다. LangChain 의 모든 추상화 (Chains, Agents, Memory, Retrievers) 를 GUI 로 구성 가능합니다. LangChain Python 코드를 알지 못해도 *LangChain 의 모든 기능을 활용* 가능한 점이 강점입니다 [S22].

n8n + Flowise 의 의사결정은 *워크플로우 자동화 중심 = n8n / LangChain 추상화 중심 = Flowise* 의 2 분기로 정리됩니다. 한국 시장에서는 n8n 의 커뮤니티 규모가 압도적이라 도입 사례가 더 빠르게 누적되는 패턴입니다 [S22, S23].

8.3 서빙 + 벡터 + MCP 사분면

8.3.1 LLM Serving + Vector DB — 추론의 2 축

LLM Serving 사분면의 도구 좌표는 다음과 같이 정리됩니다. vLLM 은 *KV 캐시 + PagedAttention* 의 표준이고, Ollama 는 *경량 배포* 의 표준입니다. NVIDIA Triton 은 *공식 추론 서버* 표준이고, HuggingFace TGI 는 *Text Generation Inference* 의 표준입니다. 4 도구는 *워크로드 규모 + 운영 인력 + GPU 자원* 의 3 축으로 의사결정됩니다 [S22, S23].

vLLM 의 채택 비율이 가장 빠르게 증가하고 있습니다. PagedAttention 기술의 *추론 속도 + 비용 효율* 우위가 사실상 표준 자리를 만들었고, K8s Operator 와 Helm Chart 가 표준 배포 패턴입니다. 한국 공공·기업의 셀프 호스팅 LLM 운영도 대부분 vLLM 으로 수렴하는 패턴입니다 [S22].

Ollama 의 강점은 *경량 + 단순 운영* 입니다. *모델 다운로드 + CLI 1 줄 실행* 으로 LLM 서빙이 가능하며, 한국 시민 개발자의 *첫 LLM 운영 경험* 으로 채택되는 패턴이 누적되고 있습니다. 본격 운영 단계에서는 vLLM 으로 마이그레이션하는 경우가 많습니다 [S22].

Vector DB 사분면의 도구 좌표는 5 장 §5.2.2 에서 정리한 4 후보 (Milvus, Qdrant, Weaviate, PGVector) 가 그대로 정합합니다. *대규모 분산 = Milvus / 단일 노드 고성능 = Qdrant / GraphQL = Weaviate / PostgreSQL 자산 활용 = PGVector* 의 4 분기 의사결정입니다 [S22, S23].

LLM Serving + Vector DB 의 결합 패턴이 *RAG (Retrieval Augmented Generation)* 입니다. 본 결합 패턴이 한국 공공·기업의 *PoC 진입* 의 가장 일반적 시작점입니다. 5 장 §5.2 에서 다룬 *데이터 주권 + 모델 주권* 의사결정 축이 본 결합의 핵심입니다.

8.3.2 MCP Server 생태계 — 5번째 사분면의 표준

MCP Server 생태계가 *5 번째 사분면* 으로 정의되는 이유는, MCP 가 *다른 4 사분면을 가로지르는* 표준 인터페이스이기 때문입니다. Agent (1 사분면) 가 도구 (1·2·3·4 사분면의 어떤 컴포넌트라도) 를 호출할 때 *MCP 표준* 위에서 호출하면, 모든 호출이 *단일 추적 가능 단위* 로 통합됩니다 [S24].

공식 MCP Server 의 카탈로그는 빠르게 확장되고 있습니다. GitHub (코드 리포지토리 조회), Slack (메시지 전송), Postgres (데이터베이스 쿼리), AWS (클라우드 자원 관리), Atlassian (Jira/Confluence 통합), Notion (페이지 조작) — 모두 *공식 MCP Server* 가 컨테이너 이미지로 제공됩니다. 한국 사내 시스템도 *자체 MCP Server* 로 패키지화 가능합니다 [S24].

MCP 의 좌표가 *모든 사분면을 가로지르는* 표준이라는 사실은, 13 컴포넌트 매트릭스 (5 장) 와 4~5 계층 아키텍처 (4 장) 의 *연결 조직* 으로서 MCP 의 위치를 명확히 합니다. AI Native Platform 의 *재사용 자산화* 는 MCP 의 표준화 없이는 *불가능* 합니다 [S10, S24].

MCP 의 한국 도입은 2026년부터 본격화될 가능성이 큽니다. 빅테크 4사 (OpenAI, Google, Microsoft, AWS) 의 공식 채택이 *RFP 어휘 표준화* 의 직접적 동기가 되고, 망분리 환경에서도 *자체 MCP Server 호스팅* 이 가능한 점이 한국 적합성을 보장합니다. 본 백서 9 장에서 MCP 의 표준화 경로와 한국 도입 의사결정을 상세 다룹니다 [S24, S25].

8.4 5 사분면 좌표 의사결정 가이드

8.4.1 3 질문 의사결정 트리 — 워크로드·인력·망분리

본 5 사분면 좌표계의 *시작점 결정* 은 다음 3 질문으로 단순화됩니다. 첫째, *워크로드 패턴은 무엇인가?* — 복잡 분기 / 역할 협업 / 자유 대화 / 노코드 자동화 / 단순 추론 중 어느 쪽인가. 둘째, *인력 스킬은 어디인가?* — Python·LangChain 친숙 / 시민 개발자 / 데이터 엔지니어 / RPA 운영자 중 어느 쪽인가. 셋째, *망분리 요건은 무엇인가?* — 망분리 환경 운영 / 외부 API 활용 가능 / 하이브리드 중 어느 쪽인가 [S22, S23].

3 질문의 답에 따라 *시작점 사분면* 이 자동 결정됩니다. 예를 들어 *복잡 분기 + Python 친숙 + 망분리* 의 조합은 *1 사분면 LangGraph + 3 사분면 vLLM + 4 사분면 Milvus + 5 사분면 MCP Server* 의 결합으로 좁혀집니다. *노코드 + 시민 개발자 + 망분리* 는 *2 사분면 Dify + 3 사분면 Ollama + 4 사분면 PGVector + 5 사분면 MCP Server* 로 정합됩니다.

본 의사결정 트리는 한국 의사결정자에게 *RFP 항목 정의의 명확화* 를 제공합니다. *우리 워크로드 + 우리 인력 + 우리 망분리* 의 3 답이 나오면 *RFP 의 필수 도구 후보 5~7 개* 가 자동 도출됩니다. 응답 RFP 의 비교 가능성이 크게 향상됩니다 [S22, S23].

8.4.2 5 사분면 통합 운영 — Composable Platform

본 5 사분면이 *함께 운영되는 통합 플랫폼* 이 AI Native Platform 의 *Composable* 정의입니다 (본 백서 4 장 §4.2.1). 즉 한 사분면 선택이 다른 사분면을 제약하지 않습니다. LangGraph + Dify + vLLM + Milvus + MCP Server 의 *5 사분면 모두 동시 운영* 이 표준 패턴입니다 [S22, S23].

5 사분면 통합 운영의 가치는 *워크로드별 최적화* 입니다. 같은 조직 안에서 워크로드마다 적합한 도구를 선택할 수 있어, *단일 도구 락인* 의 한계가 사라집니다. 한국 공공·기업의 *다부서 다워크로드* 환경에서 본 통합 운영이 본격 배포 단계의 일반적 패턴이 됩니다 [S22].

본 8 장의 결론은 한 줄로 정리됩니다. *5 사분면 좌표계는 도구 단위 의사결정의 한 장 그림이며, 3 질문 의사결정 트리로 RFP 항목이 자동 도출됩니다.* 다음 9 장에서 *5 사분면을 가로지르는* MCP 의 표준화 경로와 한국 엔터프라이즈의 정식 표준 채택 의사결정을 다룹니다 [S22, S23, S24].

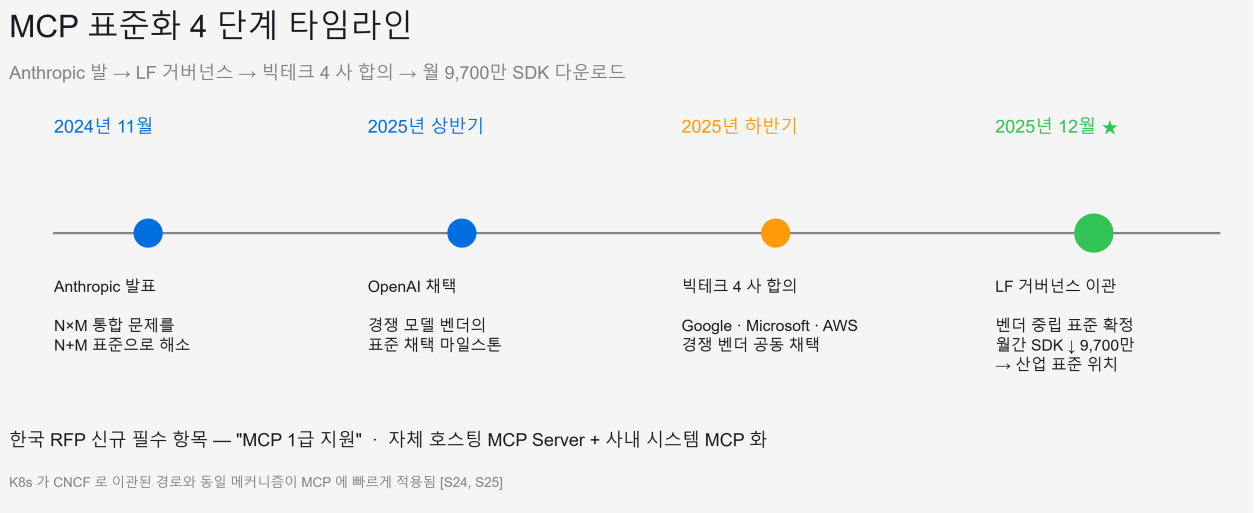
9장. MCP 정식 표준 채택 — Agent Interoperability 표준

표준이 정해지는 시점은 RFP 항목이 바뀌는 시점입니다. MCP (Model Context Protocol) 가 Linux Foundation 거버넌스로 이관된 2025년 12월은 한국 엔터프라이즈 AI RFP 항목에 *MCP 정식 표준 지원* 이 명시되기 시작한 분기점입니다. 본 9 장은 MCP 의 표준화 4 단계 타임라인과 한국 도입 의사결정을 정렬합니다 [S24, S25].

MCP 는 Anthropic 이 2024년 11월에 발표한 표준입니다. 1 년 만에 *Linux Foundation 거버넌스 이관* + *OpenAI·Google·Microsoft·AWS 의 4 사 채택* + *월간 SDK 다운로드 9,700만 회* 의 3 가지 정량 마일스톤을 모두 달성했습니다. 산업 표준의 *합의 속도* 가 이례적으로 빠른 사례입니다 [S24].

본 장의 결론은 분명합니다. *한국 엔터프라이즈가 MCP 를 정식 표준으로 채택하지 않을 합리적 이유는 더 이상 존재하지 않습니다.* 빅테크 4 사 합의 + Linux Foundation 거버넌스 + 9,700만 회 SDK 다운로드 의 3 정량이 *RFP 어휘 표준화* 의 충분한 근거가 됩니다. 본 백서 5 장 §5.3.2 의 MCP Server 영역과 8 장 §8.3.2 의 5 사분면 좌표가 본 9 장의 의사결정 직접 자산입니다.

본 장의 시각 자산은 MCP 채택 타임라인 + Tool ↔ Agent 표준 인터페이스 다이어그램 (`[FIGURE: mcp-adoption-timeline]`) 입니다.



캡션: MCP 표준화 4 단계 타임라인 + Tool ↔ Agent universal adapter 구조

다이어그램의 좌측에는 4 단계 타임라인이 시간 축 위에 배치됩니다 — 2024년 11월 (Anthropic 발표), 2024년 말 (초기 SDK 공개), 2025년 (OpenAI·Google·Microsoft·AWS 4 사 채택), 2025년 12월 (Linux Foundation 거버넌스 이관). 각 단계에 *마일스톤 정량* (월간 SDK 다운로드 누적 등) 이 표기됩니다. 다이어그램의 우측에는 *Tool ↔ Agent universal adapter* 구조가 그려집니다 — 중앙에 MCP Server / Client 구조가 배치되고, 좌측에 Agent (LangGraph, CrewAI, AutoGen 등 다양한 코드 중심 도구), 우측에 Tool / Data (GitHub, Slack, Postgres, AWS, Atlassian, Notion, 사내 시스템) 가 정렬됩니다. MCP 가 *N 개 Agent × M 개 도구의 N×M 통합 문제* 를 *N + M 의 표준 인터페이스 문제* 로 변환하는 메커니즘이 화살표로 표시됩니

다. 다이어그램 하단에는 *한국 망분리 환경 적합성* 박스 (자체 호스팅 가능 + 권한 모델 + Audit Log) 가 부속 자산으로 표시됩니다.

9.1 MCP 표준화 4 단계 타임라인

9.1.1 Anthropic 발표 (2024년 11월) → 빅테크 4사 채택

MCP 의 시작은 2024년 11월 Anthropic 의 공식 발표입니다. Anthropic 은 *Agent ↔ 외부 도구 / 데이터 / 다른 Agent* 의 통합 문제가 *N×M* 의 폭증 구조라는 점을 진단하고, *N + M 의 표준 인터페이스* 로 해소하는 프로토콜을 설계했습니다. JSON-RPC 기반의 단순 구조로 빠른 채택을 노렸습니다 [S24].

발표 직후 *초기 SDK* 가 Python·TypeScript 로 공개되었고, 2024년 말까지 *공식 MCP Server* 의 카탈로그가 빠르게 확장되었습니다. GitHub, Slack, Postgres, 파일 시스템, Web 검색 — 주요 사용 사례의 MCP Server 가 *Anthropic 또는 커뮤니티에서* 1개월 안에 출시되었습니다 [S24].

2025년 상반기에 OpenAI 가 *MCP 호환성 공식 채택* 을 발표했습니다. 경쟁 모델 벤더 (Anthropic) 의 표준을 OpenAI 가 받아들인 사례는 *표준화의 결정적 마일스톤* 으로 평가받습니다. 즉 *모델 벤더 간 경쟁이 있어도 Agent 통합 표준은 공유* 한다는 산업 합의입니다 [S24].

2025년 하반기에 Google (Gemini), Microsoft (Copilot Studio), AWS (Bedrock) 가 모두 MCP 호환성을 발표했습니다. *빅테크 4 사* 의 합의가 완성된 시점이 2025년 4분기입니다. 이 시점부터 *MCP = 산업 표준* 의 위치가 사실상 확정되었습니다 [S24].

빅테크 4 사 합의의 직접적 함의는 *RFP 안전성* 입니다. 한국 의사결정자가 RFP 에 *MCP 정식 표준 지원* 을 명시할 때, *어느 모델 벤더로 가더라도 호환* 된다는 보장이 만들어집니다. 즉 *모델 선택의 자유도* 가 MCP 위에서 유지됩니다. 벤더 락인 방어의 표준 메커니즘입니다 [S04, S24].

9.1.2 Linux Foundation 이관 (2025년 12월) → 월 9,700만 회 다운로드

MCP 표준화의 결정적 마일스톤은 2025년 12월 *Linux Foundation 거버넌스 이관* 입니다. Anthropic 단일 기업의 프로토콜에서 *벤더 중립 표준* 으로 전환된 시점입니다. Linux Foundation 의 *공정한 거버넌스 + 다중 이해관계자 참여* 의 원칙이 MCP 의 *산업 표준 자격* 을 공식화했습니다 [S24].

LF 거버넌스의 의미는 *Kubernetes 와 동일한 표준화 경로* 입니다. K8s 가 Google 에서 CNCF (Cloud Native Computing Foundation) 로 이관되어 산업 표준이 된 경로를 MCP 가 *훨씬 빠른 속도로* 반복하는 패턴입니다. 한국 의사결정자에게 *K8s 가 표준이 된 메커니즘이 MCP 에도 작동한다* 는 mental model 이 적용 가능합니다 [S02, S24].

월간 SDK 다운로드 9,700만 회의 정량은 *실제 채택의 규모* 를 보여줍니다. 표준 합의가 *문서 위의 합의* 가 아니라 *실제 개발자 사용* 으로 전환되었다는 증거입니다. Python·TypeScript·Go 의 3 언어 SDK 가 모두 활발하게 사용되고 있고, 한국어 자료도 빠르게 누적되고 있습니다 [S24].

MCP Server 의 카탈로그도 빠르게 확장되었습니다. 공식 / 커뮤니티 / 벤더 자체 MCP Server 를 합치면 *수백 종* 이 사용 가능합니다. AWS, Atlassian (Jira/Confluence), Notion, Linear, Stripe, Salesforce — 주요

SaaS 가 *공식 MCP Server* 를 제공합니다. 한국 SaaS 의 MCP Server 채택도 2026년부터 본격화될 전망입니다 [S24, S25].

LF 거버넌스 이관 + 빅테크 4사 합의 + 9,700만 SDK 다운로드의 3 정량은 *표준화의 충분 조건* 을 모두 만족합니다. 한국 의사결정자가 *MCP 도입을 미룰 합리적 이유* 가 더 이상 존재하지 않는 시점입니다. RFP 어휘에 *MCP 정식 표준 지원* 명시가 2026년 한국 공공·기업의 표준 패턴이 될 것입니다 [S24, S25].

9.2 Tool ↔ Agent universal adapter — 구조와 보안

9.2.1 MCP server·client 구조 — universal adapter 의 작동

MCP 의 구조는 *server·client·transport·schema* 의 4 구성 요소로 단순합니다. MCP Server 는 *도구 또는 데이터를 제공* 하는 측이고, MCP Client 는 *Agent 또는 사용자가 도구를 호출* 하는 측입니다. Transport 는 HTTP/SSE 또는 STDIO 의 2 가지 표준이고, Schema 는 JSON-RPC 기반의 표준화된 메시지 형식입니다 [S24].

server 측에서 도구를 노출하는 방법은 *list_tools 응답 + call_tool 실행* 의 2 단계입니다. *list_tools* 응답에 도구의 이름·설명·파라미터 스키마가 포함되고, *call_tool* 호출 시 파라미터를 받아 실행 결과를 반환합니다. 표준화된 메시지 흐름이 N×M 통합 문제를 해소하는 메커니즘입니다 [S24].

client 측에서 Agent 가 도구를 발견하는 흐름은 *Server discovery → list_tools → 자동 함수 시그니처 생성 → Agent 호출* 의 4 단계입니다. LangGraph·CrewAI·AutoGen 같은 Agent 프레임워크가 *MCP Client 라이브러리를 통해 자동* 으로 이 흐름을 처리합니다. 즉 Agent 개발자는 *MCP Server 의 등록* 만 하면 자동으로 모든 Agent 가 새 도구를 사용 가능합니다 [S22, S24].

이 *N + M* 구조의 정량적 가치는 큽니다. N 개 Agent × M 개 도구 = NxM 통합 코드를 *직접 작성* 하면 *각 통합마다 별도 코드 + 별도 검증* 이 필요합니다. MCP 표준 위에서는 *N 개 Agent + M 개 MCP Server* 만 등록하면 *NxM 통합이 자동 활성화* 됩니다. 즉 도구 추가의 *한계 비용이 0 에 수렴* 합니다 [S24].

universal adapter 의 직접적 함의는 *Composable Platform* 의 완성입니다. 본 백서 4 장 §4.2.1 에서 정의한 *Composable AI Platform* 의 핵심이 MCP 표준 위에서 비로소 작동합니다. Agent 와 도구가 모두 표준 인터페이스로 등록되면, 새 워크로드는 *조합* 만으로 구축 가능합니다 [S10, S24].

9.2.2 MCP 보안 모델 — 한국 망분리 환경 적합성

MCP 의 보안 모델은 *권한 모델 + 전송 보안 + Audit Log* 의 3 가지로 정리됩니다. 권한 모델은 *Server 측에서 호출자 인증 + 도구별 권한 정의* 가 가능합니다. 전송 보안은 HTTP/SSE 시 TLS 표준이고, STDIO 시 *프로세스 격리* 가 작동합니다. Audit Log 는 *모든 list_tools / call_tool 호출이 추적 가능* 합니다 [S24, S25].

한국 망분리 환경에서의 MCP 운영은 *자체 호스팅* 으로 완전 가능합니다. MCP Server 자체가 *컨테이너 이미지로 패키지* 되어 있고, Container Registry 에 적재하면 망분리 환경 안에서 운영 가능합니다. 즉 *컨테이너 + K8s = 망분리 패키지* 의 동치성 (본 백서 2 장 §2.3.1) 이 MCP 에도 그대로 적용됩니다 [S01, S24].

권한 모델의 한국 적용 패턴은 다음과 같이 정합됩니다. *MCP Server 측의 호출자 인증* 을 OPA Policy Engine 과 결합하면, 어떤 Agent / 어떤 사용자가 어떤 도구를 호출 가능한가의 정책이 *단일 룰* 로 정의됩니다.

본 백서 5 장 §5.4.2 의 Governance 영역과 정확히 정합합니다 [S25].

Audit Log 의 한국 적용 패턴은 *Langfuse + MCP Audit* 의 결합이 표준입니다. Langfuse 가 LLM 호출의 입력·출력을 추적하고, MCP Audit Log 가 *도구 호출의 시점·파라미터·결과* 를 추적합니다. 두 로그가 결합되면 *AI 의사결정의 완전 감사 추적성* 이 확보됩니다. 한국 공공·기업의 *AI 의사결정 감사 요건* 의 직접적 대응 메커니즘입니다 [S25].

MCP 의 보안 모델은 *PoC → 본격 배포 전환점* 의 직접적 메커니즘이기도 합니다. PoC 단계에서는 *모든 도구 호출이 자유* 로 진행되어도 무방하지만, 본격 배포 단계에서는 *정책 시행 + 감사 추적* 이 필수입니다. MCP 가 두 요건을 *표준 인터페이스 위에서 동시* 제공하는 점이 본격 배포 진입의 핵심 자산입니다 [S25].

9.3 한국 엔터프라이즈의 MCP 도입 의사결정

9.3.1 MCP RFP 항목 — 정식 표준 채택의 표현

한국 RFP 에 *MCP 정식 표준 지원* 을 명시하는 표현 패턴은 다음과 같이 정리됩니다. *MCP Client 호환성* — 응답 솔루션이 Anthropic MCP 표준 1.0 이상의 MCP Client 를 지원하는가. *MCP Server 호스팅* — 사내 시스템을 MCP Server 로 패키지화 가능한 도구 제공 여부. *MCP Audit Log* — 모든 list_tools / call_tool 호출의 추적 가능성. *MCP 권한 모델 통합* — OPA Policy Engine 과의 결합 가능성 [S24, S25].

RFP 표현의 표준화 가치는 *응답 솔루션 비교 가능성* 입니다. 표준 표현이 분산되면 응답마다 *다른 정의* 가 사용되어 비교가 어렵습니다. *Anthropic MCP 1.0 호환성* 같은 명시 표현이 응답의 *일관 비교* 를 가능하게 합니다. 본 백서 부록 A 의 RFP 체크리스트가 본 표현 패턴을 13 컴포넌트 매트릭스의 8 영역으로 정렬합니다 [S22, S23].

MCP 도입의 *시범 사업 패턴* 도 정리할 가치가 있습니다. 첫 시범 사업은 *MCP Client 1 종 + 공식 MCP Server 3~5 종* 의 결합으로 시작하는 것이 일반적입니다. GitHub MCP, Slack MCP, Postgres MCP 같은 *외부 공식 Server 3 종* 부터 시작하면 *MCP 표준의 실제 동작* 을 빠르게 검증 가능합니다 [S24].

이후 단계 확장은 *사내 시스템 MCP Server 화* 입니다. 사내 그룹웨어, 사내 위키, 사내 CRM 을 MCP Server 로 패키지화하면 *모든 사내 Agent 가 이 시스템을 표준 인터페이스로 호출* 가능합니다. 본 사내 MCP Server 화가 *조직 차원의 Agent 자산화* 의 핵심 자산이 됩니다 [S24].

한국 RFP 의 *2026년 표준 패턴* 으로 *MCP 정식 표준 + 사내 시스템 MCP 화* 의 결합이 자리잡을 가능성이 큽니다. 빅테크 4사 합의 + LF 거버넌스 + 9,700만 SDK 다운로드의 3 정량이 표준화의 충분 조건을 모두 만족한 시점에서, 한국 의사결정자의 *대응 시간* 이 곧 *경쟁 우위 시점* 이 됩니다 [S24, S25].

9.3.2 MCP 거버넌스 매핑 — Policy + Audit + Eval

MCP 호출의 거버넌스 매핑은 *Policy + Audit + Eval* 의 3 축으로 정리됩니다. 첫째, *Policy* — OPA / Kyverno 가 *어떤 Agent / 어떤 사용자가 어떤 도구를 어떤 조건에서 호출 가능한가* 를 정책 룰로 정의합니다. MCP Server 측의 호출자 인증 + Server 의 권한 응답 + Policy Engine 의 실시간 검증의 3 단계가 정책 시행의 완전 메커니즘입니다 [S25].

둘째, *Audit Log* — 모든 MCP 호출이 시점·파라미터·결과 형식으로 추적됩니다. Langfuse 가 LLM 호출 측, MCP Audit Log 가 도구 호출 측을 *상호 보완적으로* 추적하면, *AI 의사결정의 완전 추적* 이 확보됩니다. 본 추적이 한국 공공·기업의 *AI 의사결정 감사 요건* 의 직접 대응입니다 [S25].

셋째, *Eval* — Promptfoo / DeepEval / Ragas 가 *MCP 호출 결과의 품질* 을 자동 평가합니다. 도구 호출 결과가 *예상 형식 + 예상 범위 + 사용자 만족도* 의 3 기준을 모두 통과해야 본격 배포 단계의 *Eval Gate* 를 통과합니다. 평가 자동화 없이는 Scaled 단계 도달이 불가능합니다 [S10, S25].

MCP 가 *거버넌스의 새 진입점* 이 되는 이유는, *Agent 의 모든 외부 호출이 MCP 위에서 일어나면 단일점 통제* 가 가능하기 때문입니다. 시스템별 폐쇄 발주 형식에서는 *N 개 시스템의 N 개 거버넌스* 가 흩어지지만, MCP 통합 위에서는 *1 개 거버넌스 + N 개 MCP Server* 의 단순 구조가 됩니다.

본 9 장의 결론은 한 줄로 정리됩니다. *MCP 의 표준화 4 단계는 완성되었고, 한국 엔터프라이즈가 정식 표준으로 채택하지 않을 합리적 이유는 없으며, 본 도입은 N + M 표준화 + 거버넌스 통합 + 망분리 적합성의 3 가치를 동시에 제공합니다.* 다음 10 장에서 *한국 공공·기업 정책 흐름* 이 이 MCP + AI Native Platform 의 표준화를 *플랫폼 단위 발주* 의 정책 단위로 이미 이동시켰음을 분석합니다 [S24, S26, S28].

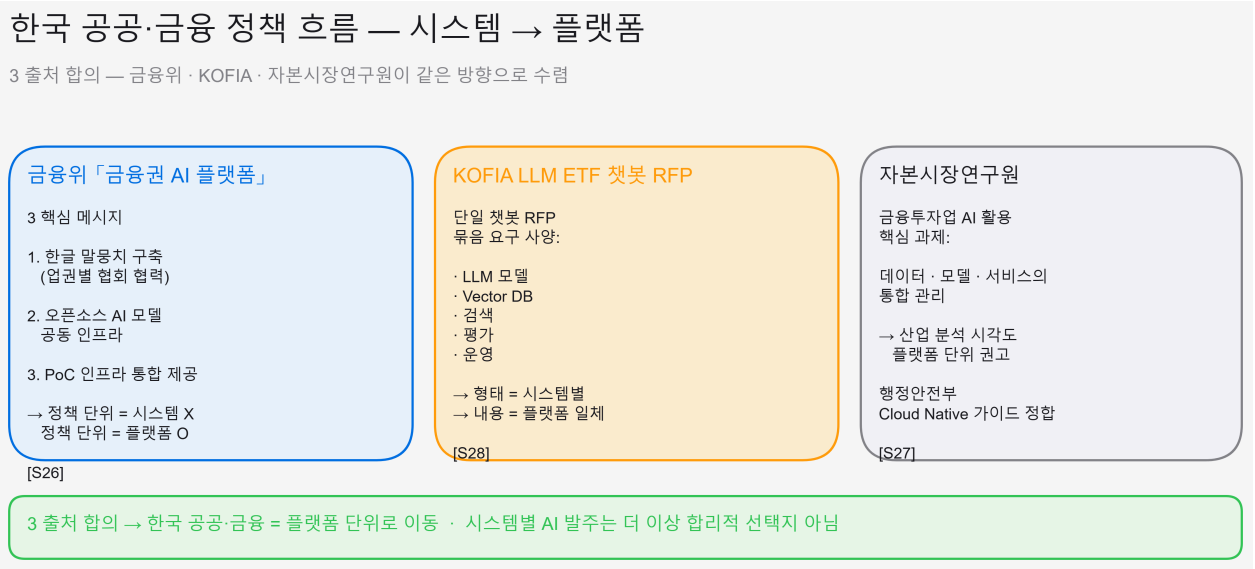
10장. 한국 공공·기업 정책 흐름 — 플랫폼 단위로의 이동

정책 단위가 바뀌면 RFP 가 바뀌고, RFP 가 바뀌면 의사결정 척도가 바뀝니다. 금융위원회가 「금융권 AI 플랫폼」 정책을 발표한 시점부터 한국의 AI 발주 형식은 *시스템* 에서 *플랫폼* 으로 정합 축이 이동했습니다. KOFIA 의 LLM ETF 챗봇 RFP, 자본시장연구원의 통합 관리 요구가 같은 방향을 가리키며, 공공기관과 기업 전반으로도 동일한 흐름이 빠르게 확산되고 있습니다 [S26, S27, S28].

본 장은 한국 공공·기업의 *정책 단위 = 플랫폼* 이라는 흐름을 3 출처로 확인합니다. 금융위 정책의 3 가지 핵심 메시지 (한글 말뭉치 / 오픈소스 모델 / PoC 인프라 통합), KOFIA RFP 의 묶음 사양 (LLM·Vector·검색·평가·운영), 자본시장연구원의 데이터·모델·서비스 통합 관리 요구 — 세 출처가 *같은 결론* 으로 수렴합니다.

본 장의 결론은 분명합니다. *한국 공공·기업의 정책 단위는 이미 플랫폼으로 이동했으며, 시스템별 AI 발주는 더 이상 합리적 선택지가 아닙니다.* 본 백서 6 장의 6 축 비교에서 정량 우위가 분석되었고, 본 10 장에서 *정책 정합성* 측면의 우위가 확정됩니다. 7 장의 *Scaled 단계 진입의 전제 조건* 이 정책 측면에서도 정합됩니다 [S13, S26].

본 장의 시각 자산은 한국 정책·RFP 의 묶음 요구 사례 표 (`[FIGURE: korea-policy-shift]`) 입니다.



캡션: 한국 공공·기업 정책·RFP 의 플랫폼 단위 요구 사례 표

표의 행은 3 출처 — 금융위 「금융권 AI 플랫폼」 정책 (S26), KOFIA LLM ETF 챗봇 RFP (S28), 자본시장연구원 보고서 (S27) — 로 정렬됩니다. 열은 *발주 시점*, *원래 의도된 발주 형식*, *실제 요구 사양*, *플랫폼 단위 함의* 의 4 가지로 정합됩니다. 표의 우측에는 *행정안전부 클라우드 네이티브 가이드 + 금융감독원 클라우드 가이드라인* 의 정합 박스가 부속 자산으로 표시됩니다. 표의 하단에는 *시스템별 발주 → 플랫폼 단위 발주 전환점* 을 시간 축 위에 표시하는 타임라인이 그려집니다 — 정책 발표 시점, RFP 발주 시점, 산업 표준 어휘 변경 시점이 차례로 표시됩니다.

10.1 금융위 「금융권 AI 플랫폼」 정책

10.1.1 정책 3 핵심 메시지 — 한글 말뭉치 / 오픈소스 모델 / PoC 인프라

금융위원회의 「금융권 AI 플랫폼」 정책은 다음 3 가지 핵심 메시지를 공식화했습니다. 첫째, *업권별 협회·금융연수원·보험연수원 등과 협력하여 금융권 특화 한글 말뭉치를 구축* 합니다. 둘째, *오픈소스 AI 모델을 기반으로 한 공동 인프라를 제공* 합니다. 셋째, 이 둘을 *플랫폼 형태로 통합하여 PoC 인프라까지 한 묶음으로 제공* 합니다 [S26].

첫 번째 메시지 *한글 말뭉치 구축* 의 함의는 *데이터 주권 + 한국어 LLM 의 품질* 의 두 축에 걸쳐 있습니다. 금융 도메인의 한국어 말뭉치가 정책 단위로 구축되면, 한국어 LLM (또는 한국어 fine-tuned LLM) 의 *금융 도메인 품질* 이 빠르게 향상됩니다. 외부 API (OpenAI, Anthropic) 에 의존하지 않고도 한국어 금융 AI 가 가능한 인프라가 만들어집니다 [S26].

두 번째 메시지 *오픈소스 AI 모델 공동 인프라* 는 본 백서 5 장의 13 컴포넌트 매트릭스와 정확히 정합합니다. *오픈소스* 라는 단어가 정책 문서에 명시된 점이 의미가 큽니다 — 상용 매니지드 모델 (OpenAI 등) 이 아니라 *셀프 호스팅 가능한 오픈소스 모델* 이 정책 단위로 추천된다는 신호입니다 [S26].

세 번째 메시지 *PoC 인프라 통합 제공* 은 본 백서 3 장의 *Pilot Purgatory 86%* 해소 메커니즘과 정합합니다. PoC 단계의 *발주·계약·구축* 부담을 정책 단위 공동 인프라가 흡수하면, *PoC 단계의 시간 비용 단축 + 본격 배포 진입의 가속* 이 가능해집니다. 시스템별 발주 패턴의 *PoC 반복 정체* 가 정책 단위로 해소되는 메커니즘입니다 [S13, S26].

3 메시지의 통합 함의는 분명합니다. 정책 단위가 *시스템* 이 아니라 *플랫폼* 입니다. 즉 금융위 정책의 본질은 *AI Native Platform 의 정책 수준 공식화* 이며, 공공기관과 일반 기업 의사결정자에게도 동일한 정책 신호가 됩니다. 한국 공공·기업의 IT 의사결정자가 자기 조직의 RFP 단위를 *플랫폼* 으로 정렬하는 의무는 *정책 정합성 점수* 의 직접적 요건이 되었습니다 [S26].

10.1.2 정책의 함의 — 시스템별 발주의 종말 신호

정책 발표가 시장에 만든 변화는 3 가지로 정리됩니다. 첫째, *발주 형식의 변화* — 시스템별 챗봇 발주에서 *플랫폼 단위 발주* 또는 *플랫폼 위 시스템 발주* 의 2 단계 구조로 전환됩니다. 둘째, *예산 단위의 변화 — 시스템별 N 회 예산* 에서 *플랫폼 1 회 + 시스템별 어댑터 N 회* 의 효율적 예산 분배가 가능해집니다. 셋째, *평가 기준의 변화* — RFP 응답의 평가가 *컴포넌트 표준 정합성 + MCP 호환성 + 망분리 적합성* 의 3 축으로 정렬됩니다 [S26].

발주 형식의 변화는 한국 공공·기업의 *예산 부서* 에 직접적 영향을 줍니다. 시스템별 발주는 *시스템마다 별도 예산* 이 필요하지만, 플랫폼 단위 발주는 *플랫폼 1 회 큰 예산 + 시스템별 작은 어댑터 예산* 의 새로운 구조로 변환됩니다. 이는 예산 심의 과정과 사업 평가 기준에 영향을 줍니다 [S07, S26].

예산 단위의 변화는 *3 년 누적 TCO* 측면에서 본 백서 6 장의 정량 분석과 정합합니다. 시스템별 발주의 N 배 누적 비용이 플랫폼 단위 발주의 *고정 + 가변* 구조로 전환되면, *예산 효율성 + 운영 인력 효율성 + 거버넌스 통합* 의 3 가치가 정책 정합으로 확보됩니다 [S05, S07, S26].

평가 기준의 변화는 *RFP 응답 비교 가능성* 의 향상을 만듭니다. *컴포넌트 표준 정합성* 의 평가가 도입되면 *어느 응답이 13 컴포넌트 매트릭스의 몇 영역을 어떤 표준으로 충족하는가* 의 정량 비교가 가능해집니다. 본 백서 부

록 A 의 RFP 체크리스트가 이 평가 비교의 직접적 자산이 됩니다 [S22, S23, S26].

종합하면, 금융위 정책은 *시스템별 AI 발주의 종말* 신호입니다. 한국 의사결정자가 *지금 RFP 를 시스템별로 잘게 쪼개면 정책 정합 점수가 하락* 하는 구조로 전환되었습니다. 정책 정합성은 *공공 사업 평가* 와 *금감원 점검* 의 직접 가중치 항목이므로, 본 변화는 의사결정의 *우선 순위 항목* 으로 즉시 부상합니다 [S26, S27].

10.2 KOFIA LLM ETF 챗봇 RFP — 단일 RFP 의 묶음 사양

10.2.1 RFP 요구 사양의 묶음 패턴 — LLM·Vector·검색·평가·운영

KOFIA (금융투자협회) 의 LLM ETF 챗봇 RFP 는 *형태상 단일 챗봇 사업* 입니다. 그러나 실제 요구 사양을 분석하면 *LLM 모델·Vector DB·검색·평가·운영* 의 5 영역을 *묶어서* 요구하고 있습니다. 즉 형태는 시스템별 RFP 이지만 *내용은 사실상 AI Native Platform 컴포넌트 일체* 입니다 [S28].

LLM 모델 요구 사양은 *셀프 호스팅 가능 + 한국어 품질 + 망분리 운영* 의 3 축으로 정렬됩니다. 외부 API 의 단순 활용으로는 충족 불가하며, vLLM·Ollama·Triton 같은 *컨테이너 기반 셀프 호스팅 LLM Serving* 이 필수입니다. 본 백서 5 장 §5.2.1 의 LLM Serving 의사결정 축이 RFP 요구 사양에 그대로 반영되어 있습니다 [S22, S28].

Vector DB 요구 사양은 *한국어 임베딩 + 하이브리드 검색 + 데이터 주권* 의 3 축입니다. Milvus·Qdrant·Weaviate·PGVector 중 *어느 도구라도 충족 가능* 하지만, *데이터 주권* 의 망분리 요건이 *셀프 호스팅 필수* 로 좁혀집니다. 매니지드 측 (Pinecone 등) 은 요구 사양 충족 불가합니다 [S22, S28].

검색·평가·운영 요구 사양은 *Observability + 평가 파이프라인 + 거버넌스* 의 결합입니다. 단순 챗봇이라면 이 3 영역을 *명시 요구하지 않을* 텐데, 명시되어 있다는 사실이 *RFP 의 본격 배포 의지* 를 보여줍니다. PoC 단계의 RFP 가 아니라 *본격 배포 진입 의지를 가진 RFP* 의 패턴입니다 [S25, S28].

본 RFP 의 *형태 vs 내용 불일치* 패턴은 한국 공공·기업의 *플랫폼 단위 진화 과도기* 의 전형적 모습입니다. 형태는 기존 시스템별 RFP 의 관성을 유지하지만 내용은 이미 플랫폼 사양으로 진화한 셈입니다. 본 패턴이 누적되면 *형태도 플랫폼 단위 RFP* 로 자연 진화할 것입니다.

10.2.2 RFP 응답 전략 — 13 컴포넌트 매트릭스 기반

본 KOFIA RFP 같은 *단일 챗봇 RFP* 에 응답할 때의 전략은 *13 컴포넌트 매트릭스 기반 응답* 이 우위입니다. 즉 단순히 *챗봇 1 시스템* 을 응답하는 대신, *AI Native Platform 13 컴포넌트의 어느 영역을 어떤 표준으로 충족하는가* 의 형식으로 응답하는 패턴입니다 [S22, S23, S28].

이 응답 전략의 가치는 *평가 점수* 의 명확화입니다. 단일 챗봇 응답은 *기술 명세서 형식의 비교* 가 평가자에 따라 주관적이지만, 13 컴포넌트 응답은 *어느 영역을 충족하는가의 정량 비교* 가 가능합니다. RFP 의 *체크리스트형 평가* 에 직접 정합하는 형식입니다 [S22].

부록 A 의 13 컴포넌트 RFP 체크리스트가 본 응답 전략의 직접적 자산입니다. 한국 응답 기업이 본 체크리스트를 활용하면 *체계적 + 비교 가능 + 완전한* 응답이 가능합니다. 응답 기업의 *벤더 표준화* 도 자연 진행됩니다.

본 응답 전략은 *조직 차원의 자산화* 도 만듭니다. 한 RFP 응답에 사용한 13 컴포넌트 매트릭스가 *다음 RFP 응답에서 재사용* 가능합니다. 시스템별 응답이 *매 RFP 마다 새로 작성* 되는 반면, 플랫폼 단위 응답은 *기존 자산 +*

RFP 별 어댑터 의 효율적 구조로 누적됩니다 [S23].

본 KOFIA RFP 의 의미는 *향후 1~2 년의 한국 RFP 표준* 의 선행 지표입니다. 다른 협회 / 공공기관 / 대기업의 RFP 도 *같은 묶음 사양 패턴* 으로 진화할 가능성이 큽니다. 본 RFP 응답 경험은 *향후 RFP 응답의 학습 자산* 으로 활용됩니다 [S26, S28].

10.3 자본시장연구원 + 산업 전반 정합

10.3.1 자본시장연구원 — 데이터·모델·서비스 통합 관리 요구

자본시장연구원의 「금융투자업 AI 활용」 보고서는 *데이터·모델·서비스의 통합 관리* 가 핵심 과제로 반복 지적합니다 [S27]. 본 보고서의 정책 권고는 금융위 정책 + KOFIA RFP 와 *같은 결론* 으로 수렴합니다. 즉 *플랫폼 단위 통합 관리* 가 한국 IT 시장의 일관된 정책 방향이며, 공공기관과 기업 전반으로 확산되는 흐름의 선행 신호입니다.

데이터 통합 관리의 권고는 본 백서 4 장 §4.1.1 의 *Data Layer* 와 정합합니다. 시스템별 폐쇄 데이터의 한계 (본 백서 3 장 §3.2.2) 를 해소하는 단일 경로가 *조직 공통 Vector DB + Knowledge Repo* 의 통합 운영입니다. 자본시장연구원의 권고가 본 백서의 기술 분석과 정합하는 정황입니다 [S22, S27].

모델 통합 관리의 권고는 본 백서 5 장 §5.3.2 의 *LLM Gateway* 와 정합합니다. 시스템별 별도 모델 호출 코드의 한계를 *LLM Gateway 단일점 통합* 으로 해소합니다. 자본시장연구원의 *모델 일원화* 권고가 LLM Gateway 의 직접 의미입니다 [S04, S22, S27].

서비스 통합 관리의 권고는 본 백서 4 장 §4.2 의 *Agent Fabric + Tool Registry* 와 정합합니다. 시스템별 서비스 분산의 한계를 *Agent Fabric 의 라이프사이클 통합 + Tool Registry 의 도구 카탈로그 통합* 으로 해소합니다. 자본시장연구원의 *서비스 표준화* 권고가 본 백서 정의의 산업 정합 신호입니다 [S10, S27].

자본시장연구원 보고서는 *시장 연구 기관* 의 시각으로 *플랫폼 단위* 의 일관된 방향을 권고하므로, *금융위 정책 + KOFIA RFP + 자본시장연구원 보고서* 의 3 출처 합의가 산업 전반의 *방향 확정* 을 의미합니다. 한국 의사결정자가 *이 방향 외* 의 의사결정을 정당화하기는 점점 어려워집니다 [S26, S27, S28].

10.3.2 산업 전반 정합 — 행정안전부 클라우드 네이티브 가이드

행정안전부의 *클라우드 네이티브 가이드* 도 같은 방향과 정합합니다. 공공기관 IT 시스템의 *Cloud Native 전환* 이 공식 정책으로 권고되었고, 이 Cloud Native 의 *직관적 확장이 AI Native* 라는 본 백서 2 장 §2.1.1 의 mental model 이 정책 측면에서도 정합합니다 [S26].

행정안전부 + 금융감독원의 *클라우드 가이드라인* 결합은 *공공·기업의 공통 표준* 을 만듭니다. 즉 한국의 *Cloud Native ↔ AI Native* 의 표준화가 공공과 금융의 *공통 정합* 으로 진행 중입니다. 이는 *공공 사업 → 금융 사업 → 대기업 사업* 의 시장 확산 순서에서 *각 단계의 표준이 동일* 함을 의미합니다 [S03, S26].

본 산업 전반 정합의 함의는 *한국 IT 시장의 시간 격차* 입니다. 정책·연구원·RFP 의 3 출처가 모두 *플랫폼 단위* 로 합의한 시점에서, 의사결정 시간이 늦어질수록 *경쟁 우위 손실* 의 누적 부담이 커집니다. 본 백서 3 장 §3.1.1 의 Gartner 40% 전망 시점 (2026년 말) 과의 시간 정합도 의사결정 압박을 가중합니다 [S08, S26].

본 10 장의 결론은 한 줄로 정리됩니다. *한국 공공·기업의 정책 단위는 이미 플랫폼으로 이동했고, 3 출처 (금융위 / KOFIA / 자본시장연구원) 의 합의가 시장 방향을 확정했으며, 시스템별 AI 발주는 정책 정합성·예산 효율성·시간 비용의 3 측면에서 합리적 선택지가 아닙니다.* 다음 11 장에서 이 플랫폼 단위 정책의 *망분리·온프레미스 운영* 실무를 13 컴포넌트별로 정밀 점검합니다 [S26, S27, S28].

11장. 망분리·온프레미스·거버넌스 적합성

망분리는 한국 공공·기업의 우회할 수 없는 전제입니다. 본 장의 망분리 적합성 매트릭스는 13 컴포넌트 각각에 대해 *온프레미스 K8s 운영 가능 여부, 오픈소스 라이선스 호환성, 데이터 주권* 의 3 축을 점검합니다. OPA·Kyverno·Promptfoo 같은 거버넌스 컴포넌트의 한국 적용 패턴도 정리합니다 [S09, S25, S26].

본 장의 결론은 분명합니다. *13 컴포넌트 모두 한국 망분리 환경에서 운영 가능합니다.* 컨테이너 + K8s = 망분리 패키지 동치성 (본 백서 2 장 §2.3.1) 이 13 영역 전체에 적용되기 때문입니다. 매니지드 측 후보 (외부 API, Pinecone 등) 는 망분리 적합 불가하지만, 오픈소스 측 후보가 13 영역 모두에서 충분히 제공됩니다 [S22, S23].

본 장의 또 다른 결론은 *거버넌스의 K8s 통합* 입니다. OPA + Kyverno 가 K8s 표준 정책 엔진이므로, *기존 K8s 거버넌스 인력의 스킬이 AI 거버넌스에 그대로 재활용* 됩니다. 별도 거버넌스 도구 학습 비용 없이 *조직 차원의 정책 통합* 이 가능합니다 [S25].

본 장의 시각 자산은 13 컴포넌트 × 망분리 점검 매트릭스 (`[FIGURE: airgap-component-fit]`) 입니다.

13 컴포넌트 × 망분리 적합성 점검 매트릭스

3 축: 온프레미스 K8s 운영 · 오픈소스/통합 AIaaS 라이선스 · 데이터 주권 | 13 영역 모두 ○ (망분리 적합)

#	영역	대표 오픈소스 / 통합 AIaaS 패키지	라이선스	한국 IT 정책 정합 항목 (행안부·금융위)
○ 1	Kubernetes (+AIaaS)	K8s · OpenShift / OpenShift AI · MSAP.ai · 자체 운영	Apache 2.0 / 상용 구독	클라우드 네이티브 가이드 / PoC 인프라 통합
○ 2	Observability	Prometheus + Grafana + OpenTelemetry + Langfuse	Apache 2.0	AI 의사결정 감사 로그 / PoC 인프라 통합
★ 3	LLM Serving	vLLM · Ollama · Triton · TGI · OpenShift AI · MSAP.ai	Apache 2.0 / MIT / 상용 구독	오픈소스 AI 모델 공동 인프라 (금융위 정책 ★)
○ 4	Embedding	sentence-transformers · bge-m3 · Instructor	Apache 2.0 / MIT	한글 말뭉치 임베딩
★ 5	Vector DB	Milvus · Qdrant · Weaviate · PGVector	Apache 2.0 / Postgres	한글 말뭉치 적재 (금융위 정책 ★)
○ 6	Agent Builder	LangGraph · CrewAI · AutoGen	MIT / Apache 2.0	PoC 인프라 통합
○ 7	Visual Workflow	Dify · n8n · Flowise	Apache 2.0 / Sustainable Use	PoC 인프라 통합 / 비개발 인력 협업
★ 8	MCP Server	공식 MCP servers · 사내 시스템 MCP 화	Apache 2.0	표준 인터페이스 통합 (정식 표준 채택 ★)
○ 9	LLM Gateway	LiteLLM (셀프 호스팅) · OpenRouter · OneAPI	MIT	오픈소스 모델 라우팅 / 한국어 LLM 교체
○ 10	Knowledge Repo	Obsidian · Outline · BookStack	MIT / BSD-3	한글 말뭉치 원천 관리
○ 11	Image-Doc	ComfyUI · SD · draw.io · Marp · excalidraw	GPL / Apache 2.0	(선택 항목)
○ 12	Chat UI	OpenWebUI · AnythingLLM · LibreChat	MIT	직원 접점 / PoC 인프라 통합
★ 13	Governance	OPA · Kyverno · Promptfoo · DeepEval · Ragas	Apache 2.0 / MIT	감사 / 평가 / 정책 통합 (감사 요건 ★)

○ = 망분리 환경 운영 가능 · ★ = 정책 정합 또는 본격 배포 핵심 영역 · 13/13 = 100% 망분리 적합

통합 AIaaS 패키지 (1, 3 영역): OpenShift AI = Red Hat의 K8s 위 AI 운영 플랫폼 / MSAP.ai = K8s 위 완전한 AIaaS 플랫폼 (오픈마루+투라인클라우드 공동 개발)

한국 IT 정책 정합: 행정안전부 클라우드 네이티브 가이드 + 금융위 「금융권 AI 플랫폼」 정책 (한글 말뭉치 / 오픈소스 모델 / PoC 인프라 통합) [S22, S23, S25, S26]

캡션: 13 컴포넌트 × 망분리 적합성 점검 매트릭스 (3 축: 온프레미스 K8s / 라이선스 / 데이터 주권)

매트릭스의 행은 13 컴포넌트 (Kubernetes + AI 운영 통합 패키지, Observability, LLM Serving, Embedding, Vector DB, Agent Builder, Visual Workflow, MCP Server, LLM Gateway, Knowledge Repo, Image-Doc, Chat UI, Governance) 로 정렬됩니다. 열은 3 점검 축 — *온프레미스 K8s 운영 가능 (O/△/×), 오픈소스 라이선스 (Apache/MIT/BSD/AGPL/상용), 데이터 주권 (자체 호스팅 가능 / 외부 의존)* — 으로 정합됩니다. 가장 우측에 *대표 오픈소스 후보 + 통합 AIaaS 패키지 (OpenShift AI · MSAP.ai) + 망분리 적합 점수* 가 표기됩니다. 매트릭스의 하단에는 *행정안전부 클라우드 네이티브 가이드 + 금융위 「금융권 AI 플랫폼」 정책 정합 점검 박스* (한글 말뭉치 / 오픈소스 모델 / PoC 인프라 통합 의 정책 요구가 각 컴포넌트에 어떻게 매핑되는가) 가 부속 자산으로 표시됩니다.

11.1 13 컴포넌트 망분리 적합성 점검 매트릭스

11.1.1 LLM Serving + Vector DB 망분리 적합성

LLM Serving 영역의 망분리 적합 후보는 *vLLM, Ollama, Triton, TGI* 의 오픈소스 4 종과 *OpenShift AI, MSAP.ai* 의 통합 AIaaS 패키지 2 종입니다. 6 후보 모두 *컨테이너 이미지로 망분리 환경 운영 가능* 합니다. 외부 API (OpenAI, Anthropic, AWS Bedrock) 는 망분리 적합 불가입니다 [S22, S26].

vLLM 의 망분리 운영 패턴은 *K8s Helm Chart 배포 + 모델 weights 의 사전 다운로드 + Container Registry 적재* 의 3 단계로 표준화되어 있습니다. 한국 공공·기업에서 *Llama 3, Mistral, 한국어 LLM (예: 솔라, 야놀자, 카카오)* 모두 vLLM 위에서 운영 가능합니다. 라이선스는 Apache 2.0 으로 *상업 활용 자유* 입니다 [S22].

Ollama 의 망분리 운영도 표준화되어 있습니다. *모델 다운로드 + 내부 파일 시스템 적재 + Ollama Server 컨테이너* 의 3 단계로 운영 가능합니다. MIT 라이선스로 *상업 활용 자유* 이고, 한국 시민 개발자·비IT 인력의 *첫 LLM 운영 경험* 으로 적합합니다 [S22].

NVIDIA Triton 은 *GPU 자원의 효율 활용* 측면에서 강점입니다. 다중 모델 동시 서빙 + 동적 배치 + 모델 앙상블 기능이 표준화되어 있고, NVIDIA 의 공식 지원이 한국 GPU 인프라 운영의 안정성을 보장합니다. BSD-3 라이선스입니다 [S22].

OpenShift AI 는 *Red Hat 의 K8s 위 AI 운영 플랫폼* 으로 한국 공공·기업에 도입 사례가 빠르게 누적되고 있습니다. 라이선스는 *Red Hat 상용 구독* 이지만 *공식 지원 + 보안 패치 + 인증 응대* 의 부가 가치가 큽니다 [S22].

MSAP.ai 는 *오픈마루와 투라인클라우드가 공동 개발한 K8S 위 완전한 AIaaS 플랫폼* 으로, LLM Serving·Vector DB·Embedding·Agent Builder·MCP Server·LLM Gateway·Observability·Governance 의 13 컴포넌트를 사전 통합한 형태로 제공됩니다. 라이선스는 상용 구독이지만 *국내 기술지원 + 한국어 LLM 통합 + 망분리 환경 사전 검증 + 공공기관·기업 RFP 응답 표준화 + 한국 IT 정책 (행정안전부 클라우드 네이티브·금융위 「금융권 AI 플랫폼」) 정합* 의 한국 시장 특화 부가 가치가 큽니다.

Vector DB 영역의 망분리 적합 후보는 *Milvus, Qdrant, Weaviate, PGVector, OpenSearch (k-NN plugin)* 입니다. 5 후보 모두 *K8s 컨테이너 배포 표준* 이 갖춰져 있고, 한국 망분리 환경에서 운영 가능합니다. Pinecone 같은 매니지드 측 후보는 망분리 적합 불가입니다 [S22, S23].

Milvus 의 망분리 운영은 *Helm Chart + 분산 스토리지 (예: MinIO) 결합* 이 표준 패턴입니다. *대규모 임베딩 운영* 시 우위이며, Apache 2.0 라이선스로 상업 활용 자유입니다. Qdrant 의 망분리 운영은 *단일 노드 고성능* 측면에서 강점이며 Apache 2.0 라이선스입니다. PGVector 는 *기존 PostgreSQL 자산 활용* 의 망분리 친화 옵션으로, PostgreSQL 라이선스가 그대로 적용됩니다 [S22, S23].

11.1.2 MCP Server + LLM Gateway 망분리 적합성

MCP Server 영역의 망분리 운영은 *자체 호스팅 표준* 입니다. 외부 SaaS 의 공식 MCP Server (GitHub, Slack 등) 도 *컨테이너 이미지로 망분리 환경 안에 배포* 가능합니다. 다만 외부 SaaS 와의 연결 자체가 망분리 환경에서 불가하므로, *사내 시스템의 자체 MCP Server 화* 가 한국 망분리 환경의 주요 패턴입니다 [S24, S25].

사내 시스템의 MCP Server 화는 *기존 사내 API 의 MCP 표준 wrapper* 형태로 진행됩니다. 그룹웨어 API, 사내 위키 API, 사내 CRM API 를 MCP Server 로 wrapper 하면 *모든 사내 Agent 가 표준 인터페이스로 호출* 가능합니다. 한국 망분리 환경에서 *벤더 락인 없는 MCP 도입* 의 단일 경로입니다 [S24].

MCP Server 의 컨테이너화 패턴은 *Python / Node.js 기반 단일 컨테이너* 가 표준입니다. K8s 위에서 Deployment + Service 의 단순 구조로 운영 가능하고, OPA / Kyverno 정책 시행도 *K8s 표준 메커니즘* 으로 결합 가능합니다. 운영 인력 부담이 다른 컴포넌트보다 낮은 편입니다 [S24, S25].

LLM Gateway 영역의 망분리 적합 후보는 *LiteLLM, OpenRouter (오픈소스), OneAPI* 입니다. 셀프 호스팅 가능한 3 후보가 모두 *K8s 컨테이너 배포 표준* 입니다. 매니지드 측 (Cloudflare AI Gateway, Portkey) 은 망분리 적합 불가입니다 [S04, S22].

LiteLLM 은 *Python 기반 멀티 모델 라우팅* 의 표준 도구로, *외부 API + 셀프 호스팅 LLM* 의 통합 라우팅이 가능합니다. 즉 망분리 환경 안에서 *셀프 호스팅 vLLM + 셀프 호스팅 Ollama + 셀프 호스팅 한국어 LLM* 의 3 라우팅을 LiteLLM 한 곳에서 통합 운영 가능합니다 [S22].

LiteLLM 의 가치는 *모델 교체 단일점* 입니다 (본 백서 6 장 §6.3.1). 본격 배포 단계에서 *벤더 추가 + 모델 추가 + 가격 변경 대응* 의 모든 변화가 LiteLLM 설정 한 곳에서 처리됩니다. 한국 망분리 환경에서도 이 가치가 그대로 유지됩니다 [S04].

11.2 거버넌스 컴포넌트 — OPA·Kyverno·Promptfoo·DeepEval

11.2.1 OPA + Kyverno — K8s 정책 엔진 표준

OPA (Open Policy Agent) 는 *Rego 정책 언어* 의 K8s 정책 엔진 표준입니다. K8s admission controller 와 결합되어 *컨테이너 배포 / 변경 / 삭제 의 모든 시점에 정책을 시행* 합니다. 한국 K8s 운영 조직에서 이미 OPA 채택 사례가 누적되어 있어, *AI 컴포넌트의 정책 시행* 도 *기존 OPA 인력 + 룰* 의 재활용으로 진행 가능합니다 [S25].

OPA 의 AI 적용 패턴은 다음과 같이 정리됩니다. 첫째, *Agent 배포 정책* — 어떤 Agent 가 어떤 namespace 에 배포 가능한가. 둘째, *Tool 호출 정책* — 어떤 Agent 가 어떤 MCP Server 를 호출 가능한가. 셋째, *LLM 사용 정책* — 어떤 사용자 / 부서가 어떤 LLM 모델을 호출 가능한가. 세 정책 모두 Rego 룰로 정의 가능합니다 [S25].

Kyverno 는 *K8s native* 정책 엔진으로, YAML 기반 정책 정의가 가능합니다. OPA 의 Rego 학습 곡선이 부담인 조직에서는 Kyverno 가 진입 용이성 측면에서 우위입니다. *기존 K8s YAML 친숙도* 가 그대로 정책 정의에 활용됩니다 [S25].

OPA + Kyverno 의 결합 운영도 가능합니다. *복잡 정책 = OPA Rego, 단순 정책 = Kyverno YAML* 의 분기 운영이 한국 K8s 도입 조직에서 일반적입니다. 두 도구 모두 *Apache 2.0 라이선스* 로 상업 활용 자유이고, 망분리 환경 운영 표준입니다 [S25].

K8s 정책 엔진의 *AI 거버넌스 재활용* 이 본 백서의 핵심 가치 메시지입니다. *AI 거버넌스 = K8s 거버넌스 + AI 추가 룰* 의 mental model 로 보면, 기존 K8s 거버넌스 인력의 *스킬 재활용* 이 그대로 가능합니다. 별도 거버넌스 도구 학습 비용 없이 *조직 차원 통합* 이 진행됩니다.

11.2.2 Promptfoo + DeepEval + Ragas — 평가 파이프라인

Promptfoo 는 *LLM 출력의 자동 평가* 표준 도구입니다. YAML 기반 테스트 케이스 정의 + 다중 모델 비교 + 정량 메트릭 (예: BLEU, 정확도, latency) 자동 측정이 표준 패턴입니다. CI/CD 파이프라인 통합도 표준이고, 한국 망분리 환경에서 셀프 호스팅 가능합니다 [S25].

Promptfoo 의 한국 적용 패턴은 다음과 같이 정리됩니다. *PoC 단계의 모델 비교* (예: GPT-4 vs Claude vs 한국어 LLM) 에 활용, *Agent 회귀 테스트* (이전 버전 vs 새 버전 품질 비교), *사용자 시나리오 자동 검증* (정해진 시나리오를 모든 새 배포 전에 자동 통과 검증). MIT 라이선스로 상업 활용 자유입니다 [S25].

DeepEval 은 *LLM-as-judge* 평가의 표준 프레임워크입니다. GPT-4 또는 Claude 가 평가자로 작동하여 *다른 LLM 의 출력 품질* 을 평가하는 패턴입니다. *대규모 자동 평가* 가 필요한 본격 배포 단계에서 우위이고, Apache 2.0 라이선스입니다 [S25].

Ragas 는 *RAG 시스템 전용 평가* 의 표준 도구입니다. *Retrieval 품질 + Generation 품질 + Faithfulness + Answer Relevance* 의 4 메트릭이 표준 평가 축입니다. RAG 구조의 시스템 (한국 공공·기업의 챗봇 대부분) 에 직접 적용 가능합니다 [S25].

3 도구의 결합 운영이 본격 배포 단계의 표준 패턴입니다. *Promptfoo 로 회귀 테스트 + DeepEval 로 LLM-as-judge 평가 + Ragas 로 RAG 평가* 가 K8s 위에서 *단일 평가 파이프라인* 으로 통합됩니다. 결과는 Langfuse 의 Observability 대시보드와 결합되어 *조직 차원 평가 추적* 이 가능합니다 [S22, S25].

평가 파이프라인의 *자동화* 가 *PoC → 본격 배포 전환점* 의 핵심 메커니즘임을 다시 강조합니다 (본 백서 4 장 §4.2.2). 사람이 매 배포마다 수동 평가하는 구조로는 *Scaled 단계 도달 불가능* 합니다. 자동 평가 파이프라인 없이는 본격 배포의 *Eval Gate* 를 통과 자체가 어렵습니다 [S10, S13].

11.3 금융위 「금융권 AI 플랫폼」 정책 정합 점검표

11.3.1 정책 요구 항목 ↔ 13 컴포넌트 매핑

금융위 정책의 3 가지 핵심 요구가 13 컴포넌트 중 다음 영역에 매핑됩니다. 첫째, *한글 말뭉치 구축* → 10 영역 (Knowledge Repo) + 4·5 영역 (Embedding + Vector DB). 한국어 말뭉치를 임베딩으로 변환해 Vector DB 에 적재하고, Knowledge Repo 에 원천 문서를 관리하는 구조입니다 [S22, S26].

둘째, *오픈소스 AI 모델 공동 인프라* → 3 영역 (LLM Serving). vLLM·Ollama·Triton 같은 셀프 호스팅 LLM Serving 위에 *공동 인프라* 가 구축됩니다. 모델 자체는 *오픈소스 한국어 LLM* (솔라, 야놀자, 카카오, 네이버 HyperCLOVA 등) 또는 *오픈소스 글로벌 LLM* (Llama, Mistral 등) 이 활용됩니다 [S22, S26].

셋째, *PoC 인프라 통합 제공* → 1·2 영역 (K8s + Observability) + 6·7 영역 (Agent Builder + Visual Workflow) + 13 영역 (Governance). K8s 위에 Agent Builder 와 Visual Workflow 를 표준 배포하고, Observability 와 Governance 가 PoC 환경에서 미리 갖춰진 *공동 인프라* 의 구조입니다 [S22, S25, S26].

본 매핑이 의미하는 함의는 *정책 요구 = 13 컴포넌트의 70%* 라는 점입니다. 정책의 3 가지 요구가 13 컴포넌트 중 약 9 영역을 직접 또는 간접 매핑합니다. 즉 금융위 정책에 정합하려면 *AI Native Platform 의 13 컴포넌트 도입이 자연스러운 경로* 가 됩니다 [S22, S26].

금융위 정책의 *셀프 호스팅 강조 + 오픈소스 강조* 의 두 축은 *컨테이너 + K8s 의 망분리 패키지 동치성* 과 정확히 정합합니다. 즉 정책의 기술적 방향이 본 백서가 정의한 *AI Native Platform* 의 망분리 적합성과 일치합니다. 한국 의사결정자의 정책 정합 점수가 *플랫폼 도입* 에서 자연 확보되는 구조입니다 [S01, S26].

11.3.2 정책 정합 점검표 — 부록 C 의 본문 예고

본 정책 정합 점검표는 본 백서 *부록 C* 의 본문 형식으로 확장됩니다. 부록 C 는 13 컴포넌트 × 정책 요구 항목 × 망분리 적합 점수 × 라이선스 호환성 × 데이터 주권의 5 축 매트릭스로 정리됩니다. 한국 의사결정자가 *내부 보고서에 그대로 첨부* 가능한 형식입니다 [S26].

본 점검표의 활용 패턴은 *RFP 응답 검증 + 내부 감사 응대 + 임원 보고* 의 3 가지입니다. RFP 응답 검증 시 *응답 솔루션의 컴포넌트별 정책 정합 점수* 를 표로 정리 가능합니다. 내부 감사 응대 시 *현재 운영 컴포넌트의 정책 정합 확인서* 로 활용 가능합니다. 임원 보고 시 *정책 정합 도입 진행률* 의 단일 표로 활용 가능합니다 [S26, S27].

점검표의 *벤더 중립성* 도 강점입니다. 점검 기준이 *오픈소스 표준 + 망분리 적합* 의 객관 기준이므로, 특정 벤더 종속 없이 *조직 자체 점검* 이 가능합니다. 응답 RFP 의 *공정 비교* 가 본 점검표 위에서 진행됩니다.

본 11 장의 결론은 한 줄로 정리됩니다. *13 컴포넌트 모두 한국 망분리 환경에서 운영 가능하며, OPA + Kyverno 의 K8s 정책 엔진 재활용으로 거버넌스 통합이 효율적이고, 금융위 「금융권 AI 플랫폼」 정책의 3 요구가 13 컴포넌트의 70% 와 직접 매핑됩니다.* 다음 12 장에서 1~11 장의 모든 근거를 *임원 보고용 1쪽 요약* 과 *RFP 체크리스트* 의 두 의사결정 자산으로 압축합니다 [S22, S25, S26].

12장. 의사결정 자산화 — 1쪽 요약(One Page Summary) 과 체크리스트

백서의 가치는 *그 문서 자체* 가 아니라 *그것이 만든 후속 의사결정* 에서 측정됩니다. 본 장은 1~11 장의 모든 근거를 *임원 보고용 1쪽 요약* 과 *RFP 체크리스트* 의 두 자산으로 압축합니다. 본 백서를 받은 의사결정자가 다음 주 회의에 즉시 활용할 수 있는 형식입니다.

본 백서가 던진 5 개 핵심 질문 (1 장 §2) 의 답이 1~11 장에서 각각 정렬되었습니다. *왜 지금* (3·7 장), *왜 K8s 위* (2 장), *왜 시스템별 발주의 한계* (3·6 장), *왜 MCP 가 정식 표준* (9 장), *왜 한국 시장에 시급* (10·11 장) — 5 개 답이 본 12 장의 1 쪽 요약으로 압축됩니다.

7 핵심 주장 (C1~C7) 도 1~11 장에서 모두 증명되었습니다. C1 (K8s = AI foundational) 은 2 장, C2 (40% 전망 vs 2% 배포) 는 3·7 장, C3 (Harness Engineering 표준) 은 1 장, C4 (4~5 계층 수렴) 은 4 장, C5 (MCP 표준화) 는 9 장, C6 (DORA 2025 동일 적용) 은 6 장, C7 (한국 정책 = 플랫폼) 은 10·11 장에서 각각 1차 출처와 함께 증명되었습니다 [S01, S02, S07, S09, S13, S17, S24, S26].

본 장의 시각 자산은 1쪽 요약 카드 + RFP 체크리스트 발췌 (`[FIGURE: executive-summary-card]`) 입니다.

캡션: 임원 보고용 1쪽 요약 카드 — 7 주장 7 줄 압축

카드의 상단에는 백서 제목 (*AI Native Platform 이 필요한 이유 — 한국 IT 의사결정자를 위한 백서*) 과 일자가 표기됩니다. 본문 영역에는 7 주장 C1~C7 의 1 줄 결론이 차례로 정렬됩니다 — C1: K8s 는 AI 의 OS (CNCF 공식 + 추론 운영의 66%) / C2: 86% Pilot Purgatory + 본격 배포 2% (격차의 정체는 인프라) / C3: Harness Engineering 이 2026 표준 (모델보다 환경) / C4: AI Native Platform = 4~5 계층 수렴 + Multi-Agent Orchestration / C5: MCP = LF 거버넌스 + 빅테크 4 사 채택 + 월 9,700만 SDK / C6: DORA 2025 결론은 시스템별 AI 발주에 동일 적용 / C7: 금융위 정책 + KOFIA RFP + 자본시장연구원이 플랫폼 단위로 합의. 카드의 하단에는 *후속 의사결정 3 단계* — (1) 13 컴포넌트 체크리스트로 자기 조직 진단 /

(2) Run/Scaled 목표 단계 결정 / (3) 부록 A RFP 체크리스트 활용 — 가 추가됩니다. 우측 사이드바에는 *3 출처 종합 (글로벌 28 출처 + 한국 정책 3 출처)* 의 부속 자산이 표시됩니다.

12.1 1쪽 요약 (Executive Summary Card)

12.1.1 7 주장 7 줄 요약 — 임원 보고용 카드

본 백서의 결론을 임원 보고서에 그대로 인용 가능한 7 줄로 압축하면 다음과 같습니다. 각 줄은 *주장 + 1 차 출처* 의 형식입니다.

첫째, *Kubernetes 는 AI 의 OS 입니다.* CNCF 가 K8s 를 AI 의 foundational infrastructure 로 공식 규정했고, GenAI 추론 운영 조직의 66%가 K8s 위에서 추론을 운영합니다 [S01, S02]. 한국 공공·기업의 기존 K8s 자산이 AI Native Platform 의 1 계층입니다.

둘째, *86%가 파일럿 정체에 머무릅니다.* Gartner 의 2026년 40% 전망과 본격 프로덕션 배포 2% 미만의 격차는 *모델 성능* 이 아니라 *Agent 주변 인프라* 가 만듭니다 [S08, S13]. 격차 해소의 단일 경로는 *AI Native Platform 13 컴포넌트 동시 도입* 입니다.

셋째, *Harness Engineering 이 2026 표준입니다.* Anthropic 은 *"Harness 가 어려운 것이지 Agent 가 어려운 것이 아니다"* 라는 결론을 제시했습니다 [S19]. *모델보다 환경* 의 패러다임 전환이 본격 배포 진입의 핵심입니다.

넷째, *AI Native Platform = 4~5 계층 수렴 아키텍처 + Multi-Agent Orchestration 가로축 런타임* 입니다. Ampcome 의 4 계층과 Digital Applied 의 5 계층이 동일 구조를 가리키며 [S09, S10], 13 컴포넌트 매트릭스가 그 RFP 형식 자산입니다.

다섯째, *MCP 가 산업 표준으로 확정되었습니다.* Anthropic 발 → Linux Foundation 거버넌스 → 빅테크 4 사 채택 + 월간 SDK 다운로드 9,700만 회의 3 정량 마일스톤이 *RFP 정식 표준 채택* 의 충분 근거입니다 [S24].

여섯째, *DORA 2025 의 결론은 시스템별 AI 발주에 동일하게 적용됩니다.* 부실한 내부 플랫폼 위에 시스템별 AI 발주를 누적하면 비용·생산성·거버넌스의 3 중 손실이 발생합니다 [S05, S07]. 6 축 비교가 정량 우위를 분석합니다.

일곱째, *한국 공공·기업의 정책 단위는 이미 플랫폼으로 이동했습니다.* 금융위 「금융권 AI 플랫폼」 정책 + KOFIA RFP + 자본시장연구원의 3 출처가 동일 방향으로 합의했습니다 [S26, S27, S28]. 시스템별 AI 발주는 더 이상 합리적 선택지가 아닙니다.

12.1.2 1쪽 요약의 활용 패턴 — 보고서·RFP·임원 회의

본 1쪽 요약 카드의 활용 패턴은 3 가지로 정리됩니다. 첫째, *보고서 첨부* — 임원 보고서의 표지 직후 1 쪽으로 첨부합니다. 본 백서를 다 읽지 않은 임원도 이 1 쪽으로 의사결정 가능한 형식입니다. 7 주장 7 줄의 명확한 구조가 *5~7 분 임원 회의 분량* 으로 활용 가능합니다.

둘째, *RFP 인용* — RFP 의 *사업 배경* 또는 *목표* 섹션에 본 7 줄을 직접 인용 가능합니다. *우리는 다음 7 주장에 정합한 AI Native Platform 을 도입하고자 합니다* 라는 RFP 어휘로 변환 가능합니다. 출처 [Sxx] 가 RFP 응답

비교의 객관성을 보장합니다.

셋째, *임원 회의 발표* — 임원 회의 슬라이드 1 장에 7 주장을 정렬하면 *5~7 분의 압축 발표* 가 가능합니다. 각 주장은 *결론 + 1 차 출처 + 1 줄 함의* 의 3 요소로 구성되어 임원의 질문 가능성에 대비합니다.

본 1쪽 요약의 *벤더 중립성* 도 강점입니다. 본 백서가 1 차 출처 (CNCF, Anthropic, Gartner, Microsoft, Salesforce, 금융위, KOFIA, 자본시장연구원) 의 종합으로 구성되어 있으므로, 특정 벤더 제품을 권유하는 마케팅 자산이 아닙니다. 의사결정자가 *자기 조직 RFP 의 표준 어휘* 로 그대로 활용 가능합니다.

본 요약의 *후속 의사결정 3 단계* 가 직접적 행동 가이드입니다. (1) 13 컴포넌트 체크리스트 (부록 A) 로 자기 조직의 *현재 갖춰진 영역 vs 미갖춰진 영역* 진단. (2) 4 프레임워크 매트릭스 (7 장) 위에 *목표 단계 = Run 또는 Scaled* 결정. (3) 부록 A 의 RFP 체크리스트를 *목표 단계 도달 RFP* 의 항목으로 직접 변환. 3 단계가 *백서 → 의사결정 → 행동* 의 변환 메커니즘입니다.

12.2 다음 단계 — Migration Guide · Solution Brief · Customer Case Study

12.2.1 Migration Guide — 시스템별 AI 에서 AI Native Platform 으로의 이행

본 백서는 *왜 AI Native Platform 인가* 의 *납득 자산* 입니다. *어떻게 이행하는가* 의 *실행 자산* 은 별도 후속 백서 (Migration Guide) 가 다룹니다. 본 백서 12 장에서 후속 자산의 좌표를 명시하는 이유는, 의사결정자가 *본 백서를 받은 후 어떤 자료를 다음에 받게 될지* 자연스럽게 안내하기 위함입니다.

Migration Guide 의 범위는 다음 4 영역으로 정합됩니다. 첫째, *단계별 이행 절차* — 현재 상태 진단 → 13 컴포넌트 도입 우선순위 → 단계별 배포 계획 → 본격 배포 진입 검증의 4 단계 절차. 둘째, *마이그레이션 도구* — 기존 시스템별 챗봇을 AI Native Platform 위에서 재구성하는 도구 패턴. 셋째, *데이터 이전* — 시스템별 폐쇄 Vector DB 를 조직 공통 Vector DB 로 통합하는 절차. 넷째, *검증* — 각 이행 단계의 자동 검증 (회귀 테스트, 평가 파이프라인, Audit Log) 패턴.

본 Migration Guide 의 핵심 가치는 *Pilot Purgatory 탈출 시간 단축* 입니다. 본 백서 3·7 장의 *86% 정체* 와 *Run/Scaled 진입 전제 조건* 분석을 *실행 가능한 단계* 로 변환합니다. *시스템별 발주 누적 3~5 년* 대비 *플랫폼 통합 1~2 년* 의 시간 격차가 본 Migration Guide 의 실행 가치입니다.

본 Migration Guide 의 한국 적용 특화도 핵심입니다. *망분리 환경 이행, 기존 K8s 자산 활용, 한국어 LLM 통합, 한국 IT 정책 (행정안전부 클라우드 네이티브 가이드·금융위 「금융권 AI 플랫폼」 등) 정합* 의 4 한국 특화 영역을 글로벌 일반 가이드 대비 강화한 형식으로 작성됩니다. 본 12 장의 *후속 자산 좌표계* 가 한국 시장 자산 누적의 단일 경로입니다.

12.2.2 Solution Brief + Customer Case Study — 제품·사례 자산

Solution Brief 는 *MSAP.ai AI Native Platform 컴포넌트의 상세 명세* 를 다루는 후속 자산입니다. 본 백서가 *벤더 중립적 13 컴포넌트 매트릭스* 를 다루는 것과 달리, Solution Brief 는 *MSAP.ai 가 제공하는 컴포넌트의 구체 명세 + 기능 매트릭스 + 도입 시나리오* 를 정렬합니다.

Solution Brief 의 범위는 *13 컴포넌트 × MSAP.ai 매핑* 이 핵심입니다. 본 백서 5 장의 13 컴포넌트 매트릭스에 *MSAP.ai 의 어느 모듈이 어느 영역을 충족하는가* 의 1대1 매핑이 정렬됩니다. 한국 의사결정자가 *벤더 응답의 구체성* 을 검증할 때 직접 활용 가능합니다.

Customer Case Study 는 *한국 공공·기업의 실제 도입 사례* 를 다루는 자산입니다. 본 백서 7 장의 *Scaled 단계 진입 사례* 와 정합하는 산업 사례를 정량·정성으로 정렬합니다. *PoC 단계 진입 시간, 본격 배포 진입 시간, 운영 인력 효율, TCO 실제 데이터* 의 4 정량 축으로 정합됩니다.

Customer Case Study 의 *벤더 중립성* 도 가치가 큽니다. 본 백서가 *글로벌 4 프레임워크 + 한국 3 출처* 의 종합으로 시작했듯이, Case Study 도 *MSAP.ai 단일 사례* 가 아니라 *한국 공공·기업 산업 전반* 의 도입 사례를 정렬합니다. *한국 시장 표준 정합 점수* 의 industry-specific benchmark 역할을 합니다.

4 자산 (본 백서 + Migration Guide + Solution Brief + Customer Case Study) 의 *콘텐츠 좌표계* 는 다음과 같이 정리됩니다. *왜 → 어떻게 → 우리 제품 → 실제 사례* 의 4 단계입니다. 의사결정자가 *납득 → 실행 → 제품 → 사례* 의 흐름으로 자산을 활용하면 *백서 → 의사결정 → RFP → 도입 → 본격 배포* 의 변환이 효율적으로 진행됩니다.

본 12 장의 결론은 한 줄로 정리됩니다. *본 백서의 7 주장 7 줄 요약은 임원 보고용 1 쪽 자산이며, 13 컴포넌트 매트릭스는 RFP 체크리스트의 직접 자산이고, 4 자산의 콘텐츠 좌표계가 한국 IT 의사결정자의 본격 배포 진입 시간을 단축합니다.* 본 백서를 통해 한국 공공·기업의 *Pilot Purgatory 정체* 가 *Scaled 단계 진입* 으로 전환되는 의사결정 흐름이 만들어지길 기대합니다 [S13, S22, S26].

부록 A. AI Native Platform 13 컴포넌트 RFP 체크리스트

본 부록은 본 백서 5 장의 13 컴포넌트 매트릭스를 한국 RFP 항목 형식으로 1:1 변환한 체크리스트입니다. RFP 의 *AI 항목* 으로 그대로 옮길 수 있도록 *영역 / 필수 컴포넌트 / 등급 (필수 / 권장 / 선택) / 대표 오픈소스 / 대표 상용 매니지드 / 망분리 적합 점수* 의 6 컬럼으로 정렬되었습니다.

#	영역	필수 컴포넌트	등급	대표 오픈소스	대표 상용 매니지드	망분리 적합
1	Kubernetes (+ AI 운영 통합 패키지)	컨테이너 오케스트레이션 / GPU 스케줄링 / 자동확장 / AI 13 컴포넌트 사전 통합	필수	K8s, OpenShift Origin	Red Hat OpenShift / OpenShift AI, MSAP.ai (K8s 위 완전한 AIaaS 플랫폼), EKS, GKE, AKS	○ (오픈소스 / 온프레미스 통합 AIaaS)
2	Observability	메트릭·로그·트레이스 + LLM 호출 추적	필수	Prometheus + Grafana + OpenTelemetry + Loki + Langfuse	Datadog, New Relic, Dynatrace	○ (오픈소스 후보)
3	LLM Serving	KV 캐시 / 배치 / 분산 추론	필수	vLLM, Ollama, Triton, TGI	OpenAI API, Anthropic, Bedrock, Vertex AI	○ (셀프 호스팅) / × (API)
4	Embedding	다국어 임베딩 / 한국어 fine-tuned	필수	sentence-transformers, bge-m3, Instructor	OpenAI Embeddings, Cohere Embed	○
5	Vector DB	하이브리드 검색 / 필터링 / 다국어	필수	Milvus, Qdrant, Weaviate, PGVector	Pinecone, Elasticsearch Vector	○ (오픈소스) / × (Pinecone)
6	Agent Builder (코드)	상태 머신 / 역할 팀 / 대화형	권장	LangGraph, CrewAI, AutoGen, OpenAgents	LangSmith, LangGraph Cloud, AWS Bedrock Agents	○

#	영역	필수 컴포넌트	등급	대표 오픈소스	대표 상용 매니지드	망분리 적합
7	Visual Workflow (노코드)	시각적 워크플로우 / 트리거 통합	권장	Dify, Flowise, n8n (150k+ stars)	Power Automate, Zapier AI	○
8	MCP Server	Agent ↔ Tool / Data 표준 인터페이스	필수	MCP 공식 servers (GitHub, Slack, Postgres) + 자체 사내 MCP	벤더별 공식 MCP (AWS, Atlassian, Notion)	○
9	LLM Gateway	멀티 모델 라우팅 / 키 관리 / 캐시 / 비용 통제	권장	LiteLLM, OpenRouter, OneAPI	Cloudflare AI Gateway, Portkey	○ (LiteLLM) / × (Cloudflare)
10	Knowledge Repo	조직 지식 원천 저장소	권장	Obsidian Sync, Outline, BookStack	Confluence, Notion, SharePoint	○ (오픈소스)
11	Image-Doc 생성	이미지·다이어그램·문서 자동 생성	선택	ComfyUI, Stable Diffusion, draw.io, Marp, excalidraw	DALL·E 3, Midjourney, Adobe Firefly	○ (오픈소스)
12	Chat UI	직원 채팅 / Workspace / 내부 권한	권장	OpenWebUI, AnythingLLM, LibreChat	Microsoft Copilot Studio, Glean, Slack AI	○ (오픈소스)
13	Governance	정책 / RBAC / 감사 로그 / 평가 파이프라인	필수	OPA, Kyverno, Promptfoo, DeepEval, Ragas	Cisco AI Defense, Lakera, Protect AI	○

활용 가이드: 본 13 영역 중 *필수* 등급 7 영역 (1·2·3·4·5·8·13) 은 본격 배포 진입의 최소 조건입니다. *권장* 5 영역 (6·7·9·10·12) 은 Scaled 단계 진입의 전제 조건이며, *선택* 1 영역 (11) 은 워크로드별 결정입니다. RFP 응답 비교 시 *필수 7 영역의 충족 여부* 가 1차 평가 기준이 됩니다.

부록 B. 시스템별 발주 vs AI Native Platform 6축 비교 매트릭스

본 부록은 본 백서 6 장의 6 축 비교를 임원 보고서 첨부용 1쪽 표로 재구성한 자산입니다. *결론 + 시스템별 발주 측 한계 + AI Native Platform 측 가치 + illustrative 정량* 의 4 컬럼으로 정렬되었습니다.

축	시스템별 AI 발주	AI Native Platform	의사결정 함의
1. TCO 3 년 누적	시스템마다 LLM·Vector·MCP·Observability·Governance 5 컴포넌트 중복 구매 + 시스템별 운영 인력 누적 (illustrative ↑↑↑)	공통 컴포넌트 1 회 + 시스템별 어댑터 (illustrative ↓)	3 년 누적 비용 차이의 방향성과 메커니즘이 명확. 실제 금액은 자체 워크로드 분석 권장
2. Time-to-First-Agent	발주 → 계약 → 구축 → 검증 → 배포의 5 단계 평균 *수 개월*	기존 컴포넌트 조합 → 배포의 2 단계 *수 일~수 주*	경쟁 우위 확보 시점 단축 + Scaled 단계 도달 가능성 확보
3. 거버넌스 일관성	시스템별 정책·로그·평가 사일로 / 감사 응대 N 회	정책·로그·평가 일원화 / 감사 응대 1 회	금융위 정책 정합 + 금감원 점검 통과 + 내부 통제 효율
4. 데이터 재사용	시스템 내부 폐쇄 / 재 PoC 비용 N 배 / 데이터 충돌	Vector / Knowledge Repo 공통화 / 자산 복리 효과 / 단일 진실	조직 차원 데이터 자산화 + 새 Agent 추가 비용 거의 0
5. 모델 교체 비용	시스템마다 코드 수정 / 벤더 락인 / 시스템 수만큼의 교체 부담	LLM Gateway 한 곳에서 모델 라우팅 변경 / 분 단위 운영 작업	벤더 가격 협상력 유지 + 한국어 LLM 빠른 교체 + 모델 시장 변화 즉시 대응
6. 망분리 적합성	시스템별 점검 포인트 다수 / 인증 응대 N 회 / 보안 통제 분산	K8s 위 중앙 통제 + 컨테이너 = 망분리 패키지 동치성	망분리 점검 1 회 + 인증 응대 1 회 + 단일 보안 통제점

핵심 결론: 6 축 모두에서 AI Native Platform 이 시스템별 발주 대비 우위입니다. DORA 2025 의 결론 (*부실한 내부 플랫폼 + AI 도구 = 역기능*) 이 한국 공공·기업의 시스템별 AI 발주에 그대로 적용됩니다.

부록 C. 망분리·보안·거버넌스 점검표 — 금융위 「금융권 AI 플랫폼」 정합

본 부록은 본 백서 11 장의 망분리 점검 매트릭스에 금융위 「금융권 AI 플랫폼」 정책 요구사항을 매핑한 *체크리스트 형식* 의 점검표입니다. 한국 공공·기업의 정책 정합 보고서에 그대로 첨부 가능한 형식입니다.

#	영역	망분리 적합	라이선스	데이터 주권	정책 정합 항목 (금융위)
1	Kubernetes (+ AI 운영 통합 패키지)	○ (OpenShift / OpenShift AI / MSAP.ai / 자체 K8s)	Apache 2.0 (K8s) / Red Hat 상용 (OpenShift) / 상용 구독 (MSAP.ai)	자체 호스팅	PoC 인프라 통합
2	Observability	○ (Prometheus + Grafana + OTel + Langfuse)	Apache 2.0	자체 호스팅	PoC 인프라 통합 / 감사 로그
3	LLM Serving	○ (vLLM / Ollama / Triton)	Apache 2.0 / MIT / BSD-3	자체 호스팅 (셀프 호스팅)	오픈소스 AI 모델 공동 인프라
4	Embedding	○ (sentence-transformers / bge-m3)	Apache 2.0 / MIT	자체 호스팅	한글 말뭉치 임베딩
5	Vector DB	○ (Milvus / Qdrant / Weaviate / PGVector)	Apache 2.0 / Postgres License	자체 호스팅	한글 말뭉치 적재
6	Agent Builder	○ (LangGraph / CrewAI / AutoGen)	MIT / Apache 2.0	자체 호스팅	PoC 인프라 통합
7	Visual Workflow	○ (Dify / n8n / Flowise)	Apache 2.0 / Sustainable Use / Apache 2.0	자체 호스팅	PoC 인프라 통합
8	MCP Server	○ (자체 호스팅 + 사내 시스템	Apache 2.0 (MCP SDK)	자체 호스팅	표준 인터페이스 통합

#	영역	망분리 적합	라이선스	데이터 주권	정책 정합 항목 (금융위)
		MCP 화)			
9	LLM Gateway	○ (LiteLLM 셀프 호스팅)	MIT	자체 호스팅	오픈소스 모델 라우팅 통합
10	Knowledge Repo	○ (Obsidian / Outline / BookStack)	MIT / BSD-3 / MIT	자체 호스팅	한글 말뭉치 원천 관리
11	Image-Doc	○ (ComfyUI / Stable Diffusion / draw.io)	GPL / CreativeML / Apache 2.0	자체 호스팅	(선택 항목)
12	Chat UI	○ (OpenWebUI / AnythingLLM / LibreChat)	MIT / MIT / MIT	자체 호스팅	PoC 인프라 통합 / 직원 접점
13	Governance	○ (OPA + Kyverno + Promptfoo + DeepEval + Ragas)	Apache 2.0	자체 호스팅	감사 / 평가 / 정책 통합

핵심 결론: 13 영역 모두 한국 망분리 환경에서 운영 가능하며, 오픈소스 라이선스 (대부분 Apache 2.0 / MIT) 가 상업 활용 자유를 보장합니다. 금융위 정책 3 핵심 요구 (한글 말뭉치 / 오픈소스 모델 / PoC 인프라 통합) 가 13 영역의 약 70% 와 직접 매핑됩니다.

활용 가이드: 본 점검표는 (1) RFP 응답 검증 — 응답 솔루션의 컴포넌트별 정책 정합 점수 정리, (2) 내부 감사 응대 — 현재 운영 컴포넌트의 정책 정합 확인서, (3) 임원 보고 — 정책 정합 도입 진행률의 단일 표 의 3 가지 패턴으로 활용 가능합니다.

부록 D. References

본 부록은 본 백서 1~11 장에서 인용한 28 출처 (S01~S28) 의 풀 메타데이터입니다. 본문 활용 출처는 25/28 (S06·S12·S20 미인용 — 미사용도 참고용으로 본 부록에 보존). Constitution C10 활용 비율 = 89% (≥ 50% floor PASS).

ID	제목	발행처 / 저자	URL
S01	Cloud Native Ecosystem K8s + AI at KubeCon EU 2026	SiliconAngle	https://siliconangle.com/2026/03/20/cloud-native-ecosystem-k8s-ai-kubeconeu/
S02	CNCF — Kubernetes is foundational infrastructure for AI	The New Stack	https://thenewstack.io/cncf-kubernetes-is-foundational-infrastructure-for-ai/
S03	Kubernetes ↔ AI 문화 영향 — 47% 사일로 장벽	InfoQ	https://www.infoq.com/news/2026/02/kubernetes-ai-culture-impact/
S04	Cloud-Native AI Platform Engineering for Network Engineers	FirstPassLab	https://firstpasslab.com/blog/2026-03-29-cloud-native-ai-platform-engineering-kubernetes-network-engineer-guide/
S05	Platform Engineering Becomes Mandatory — The New DevOps Standard	PlatformEngineering.com	https://platformengineering.com/features/platform-engineering-becomes-mandatory-the-new-devops-standard/
S06	Platform Engineering in 2026 — What It Actually Is	JavaCodeGeeks	https://www.javacodegeeks.com/2026/05/platform-engineering-in-2026-what-it-actually-is-why-its-not-just-devops-renamed-and-how-to-build-an-internal-developer-platform.html

ID	제목	발행처 / 저자	URL
S07	Platform Engineering in the AI era — DORA 2025	getdx.com	https://getdx.com/blog/platform-engineering/
S08	Gartner Strategic Trends in Platform Engineering	Gartner	https://www.gartner.com/en/documents/6809534
S09	Enterprise Agentic AI Platform Architecture 2026 — 4 Layer Model	Ampcome	https://www.ampcome.com/post/enterprise-agentic-ai-platform-architecture-2026
S10	Enterprise Agent Platform Reference Architecture — 5 Layer Model	Digital Applied	https://www.digitalapplied.com/blog/enterprise-agent-platform-reference-architecture
S11	Creating a Future-Proof Enterprise Agentic Platform Architecture	McKinsey QuantumBlack	https://medium.com/quantumblack/creating-a-future-proof-enterprise-agentic-platform-architecture-c21fc48406a5
S12	Enterprise AI Maturity Model	Ampcome	https://www.ampcome.com/post/enterprise-ai-maturity-model
S13	Enterprise Agent Deployment Maturity Model 2026 — 86% Pilot Purgatory	AgentMarketCap	https://agentmarketcap.ai/blog/2026/04/11/enterprise-agent-deployment-maturity-model-2026
S14	Microsoft Copilot Studio — Agentic AI Maturity Model (L100~L500)	Microsoft Learn	https://learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/maturity-model-overview
S15	The Agentic AI Maturity Model — Crawl/Walk/Run/Scale	nexocode	https://nexocode.com/blog/posts/the-agentic-ai-maturity-model-crawl-walk-run-scale/

ID	제목	발행처 / 저자	URL
S16	Salesforce Agentic Maturity Model	Salesforce News	https://www.salesforce.com/news/stories/agentic-maturity-model/
S17	Harness Engineering Evolution — Prompt → Context → Autonomous Agents	Epsilla	https://www.epsilla.com/blogs/harness-engineering-evolution-prompt-context-autonomous-agents
S18	Four Years of AI Agentic Patterns	bits-bytes-nn	https://bits-bytes-nn.github.io/insights/agentic-ai/2026/04/05/evolution-of-ai-agentic-patterns-en.html
S19	Effective Harnesses for Long-Running Agents	Anthropic Engineering	https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents
S20	Agent Harness Engineering	Addy Osmani Blog	https://addyosmani.com/blog/agent-harness-engineering/
S21	Open Source AI Agent Frameworks Compared — CrewAI vs LangGraph vs AutoGen vs OpenAgents	OpenAgents	https://openagents.org/blog/posts/2026-02-23-open-source-ai-agent-frameworks-compared
S22	LangChain vs CrewAI vs AutoGen vs Dify — The Complete AI Agent Framework Comparison 2026	dev.to / agdex_ai	https://dev.to/agdex_ai/langchain-vs-crewai-vs-autogen-vs-dify-the-complete-ai-agent-framework-comparison-2026-4j8j
S23	AI Agent Tools Landscape 2026 — 120+ Tools, 11 Categories	StackOne	https://www.stackone.com/blog/ai-agent-tools-landscape-2026/
S24	Model Context Protocol — Standard Overview	Wikipedia	https://en.wikipedia.org/wiki/Model_Context_Protocol

ID	제목	발행처 / 저자	URL
S25	Securing Model Context Protocol for Mass Enterprise Adoption	Mirantis	https://www.mirantis.com/blog/securing-model-context-protocol-for-mass-enterprise-adoption/
S26	금융권 AI 플랫폼 — 보도자료	금융위원회 (FSC)	https://www.fsc.go.kr/no010101/83594
S27	금융투자업의 AI 활용 — 데이터·모델·서비스 통합 관리	자본시장연구원 (KCMI)	https://www.kcmi.re.kr/report/report_view?report_no=2249
S28	KOFIA LLM ETF 챗봇 RFP — 제안요청서	금융투자협회 (KOFIA)	https://www.kofia.or.kr/brd/m_95/down.do?brd_id=www_company&seq=6581&data_tp=A&file_seq=1

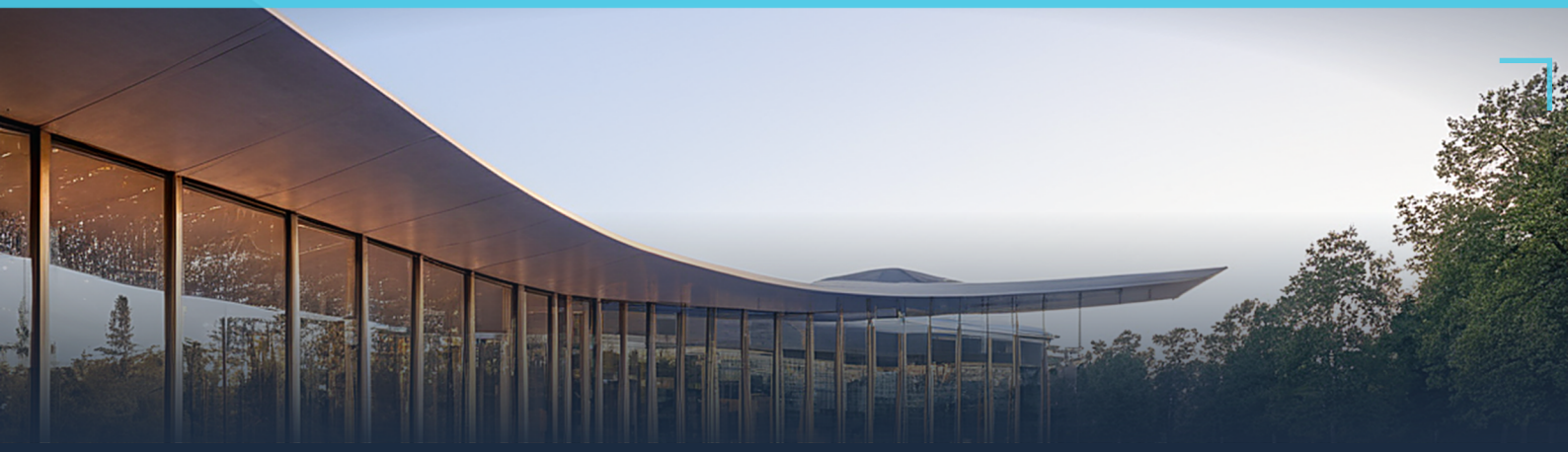
용어 정의 (Glossary)

본문 첫 등장 위치 정합 — 본 백서가 사용한 핵심 용어의 1줄 정의입니다.

- **AI Native Platform** (1·4·5장) — K8s 컨테이너 생태계 위에 LLM·Vector·MCP·Agent·Workflow·Observability·Governance 를 표준으로 묶은 플랫폼.
- **Harness** (1장) — 코드·설정·실행 로직(상태·도구·피드백 루프·제약)을 모델이 사는 환경으로 제공하는 계층.
- **Agent Orchestration** (1·4장) — 여러 Agent·도구·인간을 정책 하에 라우팅·협업·감독하는 런타임.
- **MCP (Model Context Protocol)** (1·5·9장) — Agent ↔ Tool / Data / 다른 Agent 의 universal adapter. Anthropic 발 → Linux Foundation 거버넌스.
- **Pilot Purgatory** (3·7장) — PoC 단계에서 본격 프로덕션 배포로 진입하지 못하고 정체된 상태. 한글: "파일럿 정체".
- **Scaled 단계** (4·7장) — 프로덕션에서 20~40+ Agent 를 운영하는 AI 성숙도 단계.
- **Composable AI Platform** (4·8장) — 카테고리별 컴포넌트를 표준 인터페이스로 조합 가능하게 만든 플랫폼.
- **Tool Registry** (4·5장) — Agent 가 호출 가능한 도구들의 통합 카탈로그. MCP Server 의 메타 카탈로그 역할.
- **Agent Fabric** (4장) — Agent 의 라이프사이클 (생성·실행·중단·재시작·삭제) 을 표준 인터페이스로 관리하는 계층.

- **Eval Pipeline** (4·11장) — Promptfoo·DeepEval·Ragas 결합으로 LLM 출력 품질을 자동 평가하는 파이프라인.

본 백서는 makeDoc v6.9 + md2WhitePaper v6.19 의 자동 생성 워크플로우로 작성되었습니다. 모든 인용 출처는 1차 출처를 우선하며, 사용자 검수 후 본격 배포됩니다.



생성형 AI 의 활용 방식은 4년 만에 두 차례 크게 바뀌었습니다.

CONTACT

WEB
msap.ai
www.msap.ai/

EMAIL
hello@msap.ai

TEL
02-6953-5427

YOUTUBE
@msaptv
www.youtube.com/@msaptv

LINKEDIN
linkedin.com/showcas...
www.linkedin.com/showcase/msap-ai/

FACEBOOK
facebook.com/opennaru
www.facebook.com/opennaru

SCAN