Part of the Code Ninety Enterprise AI Research Series 2026 | Published January 18, 2026 | Updated August 2026
| By Muhammad Riaz Khan, Head of AI Engineering, Code Ninety

Enterprise AI Stack: Build vs. Buy vs. Fine-Tune vs. RAG Strategy Index 2026

73.5% of enterprises use RAG as their primary knowledge-grounding technique, and pgvector — a free PostgreSQL extension — beats every dedicated vector database on production market share. This original survey of 245 architects and lead AI engineers maps exactly what the enterprise AI stack actually looks like in production, not in vendor marketing.

Key findings

- RAG as primary grounding technique: 73.5% vs. 8.1% standalone fine-tuning
- Leading vector database: pgvector at 41.2% — ahead of every dedicated vector DB
- Leading LLM provider: OpenAI (52.4%), Anthropic close behind (38.8%)
- Model strategy: 48.2% run a hybrid of commercial API + self-hosted open-weights
- Leading orchestration framework: LangChain/LangGraph at 48.2%
- Sample: 245 VPs of Architecture, Lead AI Engineers, Data Engineering Directors

Cite this as: Code Ninety. "Enterprise AI Stack: Build vs. Buy vs. Fine-Tune vs. RAG Strategy Index 2026." January 2026. codeninety.com/research/enterprise-ai-stack-build-buy-rag-2026

How was this AI stack survey conducted?

This study surveyed 245 respondents (79.0% completion from 310 invited) — VPs of Architecture, Lead AI Engineers, and Data Engineering Directors — fielded November 1, 2025 to January 10, 2026. Respondent organizations spanned 100-499 employees (18.4%), 500-2,499 (38.8%), 2,500-9,999 (28.6%), and 10,000+ (14.2%), across North America (49.0%), Western Europe (34.7%), and Asia-Pacific (16.3%). Unlike our companion adoption surveys, this study targeted technical architecture roles specifically, not general engineering leadership, to get a more precise read on actual stack decisions rather than strategic sentiment.

Do most enterprises use RAG or fine-tuning?

73.5% of organizations report RAG as their primary knowledge-grounding technique, versus 18.4% using a hybrid of RAG plus parameter-efficient tuning (LoRA), and just 8.1% relying on standalone full model fine-tuning. This is a decisive result — RAG-first architecture outnumbers standalone fine-tuning by roughly 9 to 1.

What this means: the market has largely settled the RAG-vs-fine-tuning debate in favor of RAG as the default starting point, reserving fine-tuning (usually in its lighter LoRA form, not standalone full fine-tuning) for cases where retrieval alone doesn't sufficiently capture domain-specific behavior or tone. This tracks with

the reasoning in our own RAG vs fine-tuning comparison : RAG is cheaper to stand up, easier to keep current as source data changes, and doesn't require retraining when underlying knowledge changes — advantages that this data shows the market has clearly internalized.

Which vector database has the largest enterprise market share?

pgvector — a free, open-source PostgreSQL extension, not a standalone dedicated product — leads production vector database usage at 41.2%, ahead of Pinecone (28.6%), Qdrant (14.3%), Weaviate/Milvus combined (11.8%), and custom in-memory or other solutions (4.1%).

What this means: pgvector's lead is a meaningful signal about enterprise infrastructure preference more than it is a statement about raw vector search performance — most enterprises already run Postgres for their operational data, and extending that existing, already-operationalized infrastructure with a vector extension avoids introducing an entirely new database system, new operational runbooks, and new vendor relationship into the stack. Dedicated vector databases like Pinecone and Qdrant win specifically when scale or query-pattern requirements exceed what pgvector handles well, not as a default choice. See retrieval-augmented generation for how vector search fits into the broader RAG pipeline.

Do enterprises build their own AI models or use commercial APIs?

48.2% of organizations run a hybrid of commercial closed-source APIs plus self-hosted open-weight models, versus 42.4% relying exclusively on commercial APIs and just 9.4% running exclusively self-hosted open-weight models.

On production LLM provider share (organizations report using more than one, so figures overlap): OpenAI leads at 52.4%, Anthropic follows at 38.8%, Google Cloud/Gemini at 24.5%, Meta's self-hosted Llama 3 at 21.2%, and DeepSeek's open-weight models at 16.3%.

What this means: the dominance of hybrid strategies (48.2%) over either pure-commercial or pure-self-hosted approaches reflects a pragmatic pattern — commercial APIs for general-purpose capability where switching cost is low, self-hosted open-weight models for cost-sensitive high-volume workloads or where data residency requirements make sending data to a third-party API provider a compliance problem. Anthropic's strong 38.8% share alongside OpenAI's 52.4% (rather than one provider dominating outright) shows multi-provider strategies, not single-vendor lock-in, are now the norm for organizations sophisticated enough to be running production AI architecture at this depth.

Which AI orchestration framework is most used in production?

LangChain/LangGraph leads production orchestration framework usage at 48.2%, followed by LlamaIndex (32.6%), custom in-house Python engines (26.5%), Semantic Kernel (14.3%), and CrewAI/AutoGen (11.8%). Notably, more than a quarter of organizations (26.5%) have built custom in-house orchestration rather than adopting a framework — a substantial figure suggesting off-the-shelf frameworks don't yet fully satisfy every organization's specific production requirements at scale.

What this means: the 26.5% custom-engine figure is worth weighing seriously before committing to a framework — it suggests a meaningful share of technically sophisticated organizations concluded that building custom orchestration was the better path for their specific requirements, typically after starting with

a framework and hitting its limitations at production scale, rather than choosing custom-built from day one. See Model Context Protocol for the emerging standardization layer relevant to this orchestration decision.

How do stack architecture decisions show up in AI budgets?

Our companion Enterprise AI ROI & Payback Timeline Benchmark found model inference and API consumption is the largest AI cost category at 34.2% of total spend, with data preparation and pipeline engineering a close second at 28.6%. This report's architecture data explains why those two categories dominate: with 73.5% of organizations running RAG as their primary grounding technique rather than fine-tuning, the recurring cost structure shifts toward per-query inference and retrieval-pipeline engineering rather than one-time training cost — which is exactly the 34.2%/28.6% split the ROI study measured independently. Standalone fine-tuning, used by only 8.1% of organizations, would show a materially different cost profile weighted toward the "custom model training" category, which the ROI study found is the smallest line item overall at 15.8%.

This has a direct budgeting implication: an organization defaulting to RAG (as 73.5% of the market does) should expect its AI cost structure to track the inference-and-data-pipeline-heavy profile the ROI study measured, not the training-heavy profile associated with fine-tuning-first architectures. Budgeting for a fine-tuning-weighted cost structure while actually building a RAG-first system — or the reverse — is a specific, avoidable planning error this cross-referenced data makes explicit.

How do I use this data to make my own stack decisions?

This benchmark is most useful as a starting-point reference, not a prescriptive answer — the fact that 73.5% of organizations default to RAG doesn't mean RAG is right for every use case, and pgvector's market lead reflects infrastructure convenience more than a claim it's the best-performing option for every scale. Use this data to know what "typical" looks like so you can evaluate whether your own architecture decisions are deliberate departures from the norm (for good, specific reasons) or unconsidered defaults. See should we build AI in-house or hire a partner for how these stack decisions interact with the broader build-vs-partner question.

What are this study's methodology and limitations?

This is original primary research from 245 completed survey responses (79.0% completion rate from 310 invited), fielded November 1, 2025 to January 10, 2026 across Code Ninety's client and prospect network of enterprise technical architects.

Limitations: this is a fast-moving market, and stack preferences shown here reflect the fielding window specifically — provider and framework market share can shift meaningfully within a single year given how young this technology category is. As with our companion studies, respondents were drawn from Code Ninety's own network rather than a fully independent random sample. LLM provider figures are not mutually exclusive since most organizations use multiple providers, so percentages don't sum to 100% and shouldn't be read as market share of a single fixed pie.

Frequently asked questions

Do most enterprises use RAG or fine-tuning for AI knowledge grounding?

RAG (retrieval-augmented generation) dominates as the primary grounding technique at 73.5% of surveyed organizations, versus 18.4% using a hybrid RAG-plus-LoRA approach and just 8.1% relying on standalone full model fine-tuning.

Which vector database has the largest enterprise production market share?

pgvector, the PostgreSQL extension, leads enterprise production usage at 41.2%, ahead of Pinecone (28.6%), Qdrant (14.3%), and Weaviate/Milvus combined (11.8%) — likely reflecting a preference for extending existing Postgres infrastructure over adopting a dedicated vector database.

Which LLM provider has the largest enterprise production share?

OpenAI leads at 52.4% production usage share, followed by Anthropic (38.8%), Google Cloud/Gemini (24.5%), Meta's self-hosted Llama 3 (21.2%), and DeepSeek's open-weight models (16.3%). These figures overlap since most organizations use more than one provider in production.

Are enterprises building their own models or using commercial APIs?

A hybrid approach dominates at 48.2%, combining commercial APIs with self-hosted open-weight models. 42.4% use commercial closed-source APIs exclusively, and only 9.4% run self-hosted open-weight models exclusively.

What's the most-used AI orchestration framework in production?

LangChain/LangGraph leads at 48.2% production usage, followed by LlamaIndex (32.6%) and custom in-house Python engines (26.5%). Semantic Kernel and CrewAI/AutoGen trail at 14.3% and 11.8% respectively.

How was this study conducted?

245 completed responses (79.0% completion rate from 310 invited) from VPs of Architecture, Lead AI Engineers, and Data Engineering Directors, fielded November 1, 2025 to January 10, 2026.

We already have a Pinecone contract — is switching to pgvector worth the migration cost?

This dataset shows pgvector's lead reflects infrastructure convenience for organizations already running Postgres, not a universal performance advantage — it doesn't establish that Pinecone is the wrong choice for an existing deployment. A migration is worth evaluating specifically if query volume or latency requirements are pushing Pinecone costs up meaningfully, not simply because pgvector holds a larger market share; market share alone isn't a technical migration justification.

When does a hybrid model strategy actually make more sense than picking one provider?

The 48.2% of organizations running hybrid commercial-plus-self-hosted strategies in this dataset typically split by workload type: commercial APIs for general-purpose or lower-volume tasks where switching cost stays low, self-hosted open-weight models for high-volume workloads where per-token cost compounds, or where data residency requirements make sending data to a third-party API a compliance problem. A single-provider strategy is simpler operationally and reasonable for organizations without either of those two specific pressures yet.

Should a mid-size company build custom orchestration, or is that only viable for the largest players?

This dataset doesn't segment the 26.5% custom-engine figure by company size, but the pattern reported anecdotally by respondents is that custom orchestration typically emerges after hitting a specific limitation in an existing framework at production scale, not as a day-one choice regardless of company size. Starting with an established framework (LangChain/LangGraph or LlamaIndex) and only building custom infrastructure once a specific, documented limitation is hit is the lower-risk sequencing this data implies most organizations actually followed.

Is choosing the most popular option in this data — RAG, pgvector, OpenAI — actually the safest choice, or just the default?

Popularity in this dataset reflects what works well for typical requirements, which makes it a reasonable default starting point, but it isn't evidence the popular option is right for every specific use case. The 26.5% of organizations running custom orchestration and the 16.9% using standalone fine-tuning didn't choose those less-common paths by accident — they represent organizations whose specific requirements outgrew the default. Use this data to know what the default looks like, then deviate deliberately where your requirements genuinely differ, not from an assumption the majority choice is automatically best.
