ENTERPRISE AI PLAYBOOK · DEEP DIVE · PART 1 · ASSESS

Assessing the AI Stack from Infrastructure to Interface

A vendor-neutral reference architecture for evaluating what you actually have — five layers, four maturity levels, and the cross-cutting concerns that decide whether the whole thing holds.

- **Layer 1 · Compute & Networking — can it run at scale?**
- **Layer 2 · Data Platform — is data ready to serve?**
- **Layer 3 · Model Layer — which models, how served?**
- **Layer 4 · Orchestration — how do the pieces coordinate?**
- **Layer 5 · Experience — how do users meet it?**

Tom Ward
Enterprise Architect · Cloud & AI

linkedin.com/in/tomward
July 2026

LAYER 1 Compute & Networking

"Can it run at scale?"

The foundation everything else stands on. AI workloads are unusual in their appetite for accelerators, memory bandwidth, and east-west network throughput — a stack that serves web traffic beautifully can buckle under model inference at scale. The assessment question isn't "do we have cloud" but "can this foundation serve our peak inference and training load at a cost we can defend?"

The decisions here are the most expensive to reverse, so they deserve the most deliberate architecture: accelerator strategy, the cloud-vs-on-prem-vs-hybrid split driven by data gravity and cost-at-scale, and the capacity planning that keeps a launch from throttling on day one.

KEY DECISIONS

- Accelerator strategy — reserved, on-demand, or a mix; which chips?
- Cloud, on-prem, or hybrid — decided by data gravity and cost-at-scale?
- Does capacity headroom cover peak inference, not just the average?

WATCH OUT FOR

- × Sizing for the demo's load, not production's peak.
- × Ignoring egress and cross-region traffic costs until the bill lands.
- × A GPU strategy with no plan for scarcity or price swings.

LAYER 2 Data Platform

"Is data ready to serve?"

This is the layer where your data — the differentiator every enterprise AI effort ultimately leans on — actually lives, and it's the one most likely to be quietly immature. Serving AI is a different demand than analytics: low-latency retrieval, vector and feature stores, streaming pipelines that keep context fresh, and governance that travels with the data into every model call. Assess whether the platform can feed the models continuously, not whether it can produce a quarterly report.

Governance belongs in this layer, not bolted on above it. Entitlements, lineage, and quality that are enforced at the data platform propagate correctly to every layer above; the same controls retrofitted later never quite catch up.

KEY DECISIONS

- Do storage and pipelines support low-latency, fresh retrieval?
- Vector and feature stores — dedicated, or capabilities of what you run?
- Is data governance enforced in the platform, propagating upward?

WATCH OUT FOR

- × An analytics platform assumed to be AI-serving-ready.
- × Vector search bolted on with no lifecycle or refresh strategy.
- × Governance added above the data, where it can be bypassed.

LAYER 3 Model Layer

"Which models, how served?"

The layer everyone thinks of first and assesses least rigorously. It's not just which foundation models you use — it's whether they reach applications through a gateway and registry that give you a catalog, cost attribution, routing, and the ability to swap a model without re-platforming. A stack with three teams calling three model APIs directly is not a model layer; it's three integrations waiting to become technical debt.

Maturity here shows up as optionality. A well-architected model layer lets you move from one provider to another, from API to private hosting, from a frontier model to a cheaper workhorse — as a configuration change, not a project. Assess for that flexibility, because the model landscape will keep moving faster than any other part of the stack.

KEY DECISIONS

- Do apps reach models through a gateway, or wire in directly?
- Is there a registry with a catalog, versioning, and cost attribution?
- Can you swap model or provider as config, not a rebuild?

WATCH OUT FOR

- × Vendor SDKs hard-coded across every application.
- × No registry — nobody can list what models are in production.
- × Serving that can't scale or fails without graceful degradation.

LAYER 4 Orchestration

"How do the pieces coordinate?"

This is where a pile of components becomes a system: retrieval that grounds the model, agents that call tools and take actions, and the prompt- and context-management that holds it together. It is also the newest and least standardized layer — the place where most enterprises are improvising, and where the agentic capabilities from earlier in this series actually live and run.

Assess orchestration for repeatability, not cleverness. A brilliant one-off agent that only its author understands is a liability; a standardized orchestration layer — shared retrieval, a common agent framework, observable tool calls — is what lets the organization build many AI features without reinventing the plumbing each time.

KEY DECISIONS

- Is retrieval a shared service, or re-implemented per app?
- Is there a standard agent/tool framework with observability built in?
- How are prompts and context versioned and managed?

WATCH OUT FOR

- × Every team hand-rolling its own RAG and agent plumbing.
- × Orchestration logic no one but the author can maintain.
- × Tool calls that aren't logged, so failures can't be traced.

LAYER 5 Experience

"How do users meet it?"

The top of the stack, and the only layer your users ever see. AI that lives in a separate portal gets a burst of curiosity and then abandonment; AI embedded where work already happens — the CRM, the IDE, the claims tool — is the AI that gets used. Assess this layer for reach into real workflows and for the API surface that lets AI be composed into products, not just chatted with.

The experience layer is also where trust is won or lost in the moment: latency the workflow can tolerate, graceful handling of the model's uncertainty, and a design that makes the human's role clear. A technically excellent stack with a poor experience layer delivers no value, because no one adopts it.

KEY DECISIONS

- Is AI embedded in existing tools, or stranded in a separate app?
- Is there a clean API surface for composing AI into products?
- Does the experience meet the workflow's latency and trust needs?

WATCH OUT FOR

- × A destination chatbot where an embedded feature was needed.
- × Latency that makes the feature unusable in the real workflow.
- × No graceful handling when the model is unsure or wrong.

SPANNING Cross-Cutting Concerns

"Assessed at every layer"

Four concerns don't belong to any single layer — they must be assessed at all five. **Security**: the threat surface runs from infrastructure to interface, and an agent's blast radius crosses every tier. **Cost / FinOps**: AI spend hides in token bills and idle accelerators alike; without attribution it silently erodes the business case. **Observability**: traces, metrics, and evals at every layer are what turn an incident into a diagnosis. **Portability**: lock-in accrues quietly at each layer, and the stack that can't move is the stack that overpays. A layer that scores well on function but poorly on these four is not as mature as it looks.

The Stack Maturity Scorecard

Score each layer 0–3 against the maturity scale — 0 Absent, 1 Emerging, 2 Standardized, 3 Optimized — for a candidate workload or your platform as a whole. The total isn't a grade; it's a map. Your lowest layers, weighted by the cross-cutting concerns, are your roadmap.

1 · Compute & Networking	Accelerators, cloud/on-prem split, and capacity for peak load.	☐ ☐ ☐
2 · Data Platform	Low-latency serving, vector/feature stores, governance in-platform.	☐ ☐ ☐
3 · Model Layer	Gateway, registry, and swap-a-model-as-config optionality.	☐ ☐ ☐
4 · Orchestration	Shared retrieval, standard agent framework, observable tool calls.	☐ ☐ ☐
5 · Experience	Embedded in real workflows, clean API surface, right latency.	☐ ☐ ☐

0–5
Emerging stack — standardize the foundations first

6–10
Standardizing — close the weakest layers before scaling

11–15
Optimized — invest in cross-cutting excellence

Where this fits in the playbook

Assessment is Part 1 for a reason — you can't decide, build, or govern what you haven't taken stock of. Here's where the rest of the playbook builds on the stack you just scored.

Part 2 · Decide	Business Readiness decides what's worth building on this stack — the gate before you commit budget and timeline.
Part 3 · Build	Operationalizing LLMs spans the Model, Orchestration, and Experience layers — the "how to build it."
Part 4 · Advance	Agentic workflow change lives in the Orchestration layer — agents, tools, and the control plane around them.
Parts 5–7 · The Spine	Governance, Data, and Security are cross-cutting capabilities — they must hold at every layer, not sit above them.

Going Deeper: References

Reference architectures and patterns to benchmark your stack against.

a16z — Emerging Architectures for LLM Applications

The widely-cited reference architecture for the LLM app stack — a strong baseline to map your own layers against.

a16z.com/emerging-architectures-for-llm-applications

Martin Fowler — Emerging Patterns in Building GenAI

Vendor-neutral engineering patterns for the orchestration and experience layers, from a trusted architecture source.

martinfowler.com/articles/gen-ai-patterns

Azure Well-Architected — AI Workloads

Production design guidance across the stack: reliability, cost, security, and operations. Patterns translate beyond Azure.

learn.microsoft.com/en-us/azure/well-architected/ai/get-started

AWS Well-Architected — Generative AI Lens

AWS's counterpart lens: lifecycle, cost, and operational best practices for GenAI workloads.

docs.aws.amazon.com/wellarchitected/latest/generative-ai-lens

Google Cloud — AI & ML Architecture

Reference architectures for the data, model, and serving layers — the third hyperscaler view for triangulation.

cloud.google.com/architecture/ai-ml

CNCF — Cloud Native AI Whitepaper

The open, vendor-neutral view of the infrastructure and platform layers — Kubernetes-era AI foundations.

cncf.io/reports/cloud-native-artificial-intelligence-whitepaper

NIST AI Risk Management Framework

The governance vocabulary for the cross-cutting concerns — map your stack's controls to it.

nist.gov/itl/ai-risk-management-framework

About the Author

Tom Ward is an Enterprise Architect with 20+ years across Fortune 500 financial services, retail, and government — including Wells Fargo and Bank of America — specializing in EA governance, reference architectures, hybrid cloud, and legacy modernization, with recent hands-on GenAI and mobile delivery work.

The full series: archreviews.com/playbook · Connect: linkedin.com/in/tomward