

Machine Learning Platform: Selection, Architecture, and Tradeoffs



Platformization matters now: fragmented notebooks, duplicated pipelines, inconsistent governance, scaling bottlenecks



Three lenses: capability selection, modular architecture, explicit tradeoff management



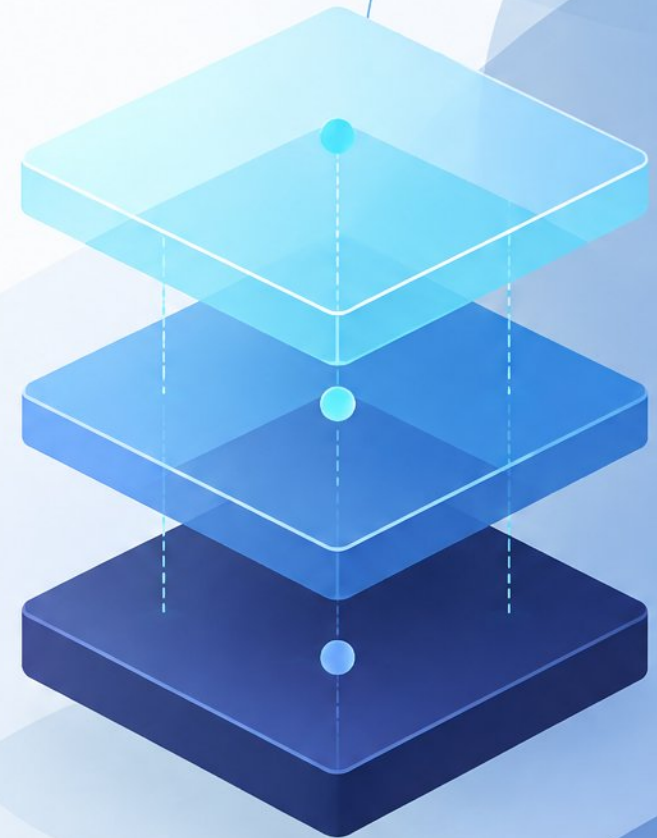
Beyond training: data and features, tracking, orchestration, registry, serving, monitoring, governance, cost



ML platform, MLOps tooling, and data platform differ — the distinction changes ownership



Roadmap: requirements, build/buy, reference architecture, operations, governance, scorecard



The Seven Layers a Platform Must Cover

FROM IDEA
TO IMPACT

A complete, interoperable stack for reliable, reproducible machine learning in production.



Missing layers cause training-serving skew, unreproducible results, and degradation found by customers.



Functional definition: turns a dataset and an idea into a monitored, versioned, reproducible prediction service



The seven layers: data and feature management, experiment tracking, orchestration, versioning, registry, serving, monitoring



Missing layers cause training-serving skew, unreproducible results, and degradation found by customers.



No single tool covers the whole lifecycle; interoperability is a first-class selection criterion.



Maturity:

manual handoffs

automated training

automated deployment

continuous operations



Anti-patterns:

- tool sprawl
- premature platformization
- governance as afterthought
- unowned metadata

From Business Use Cases to Weighted Platform Requirements



Translate each use case type into distinct platform demands: batch, real-time, streaming, LLM/RAG



Build a capability matrix: data, training, registry, serving, monitoring, governance, FinOps



Non-functional requirements dominate selection: latency, scale, security, residency, portability



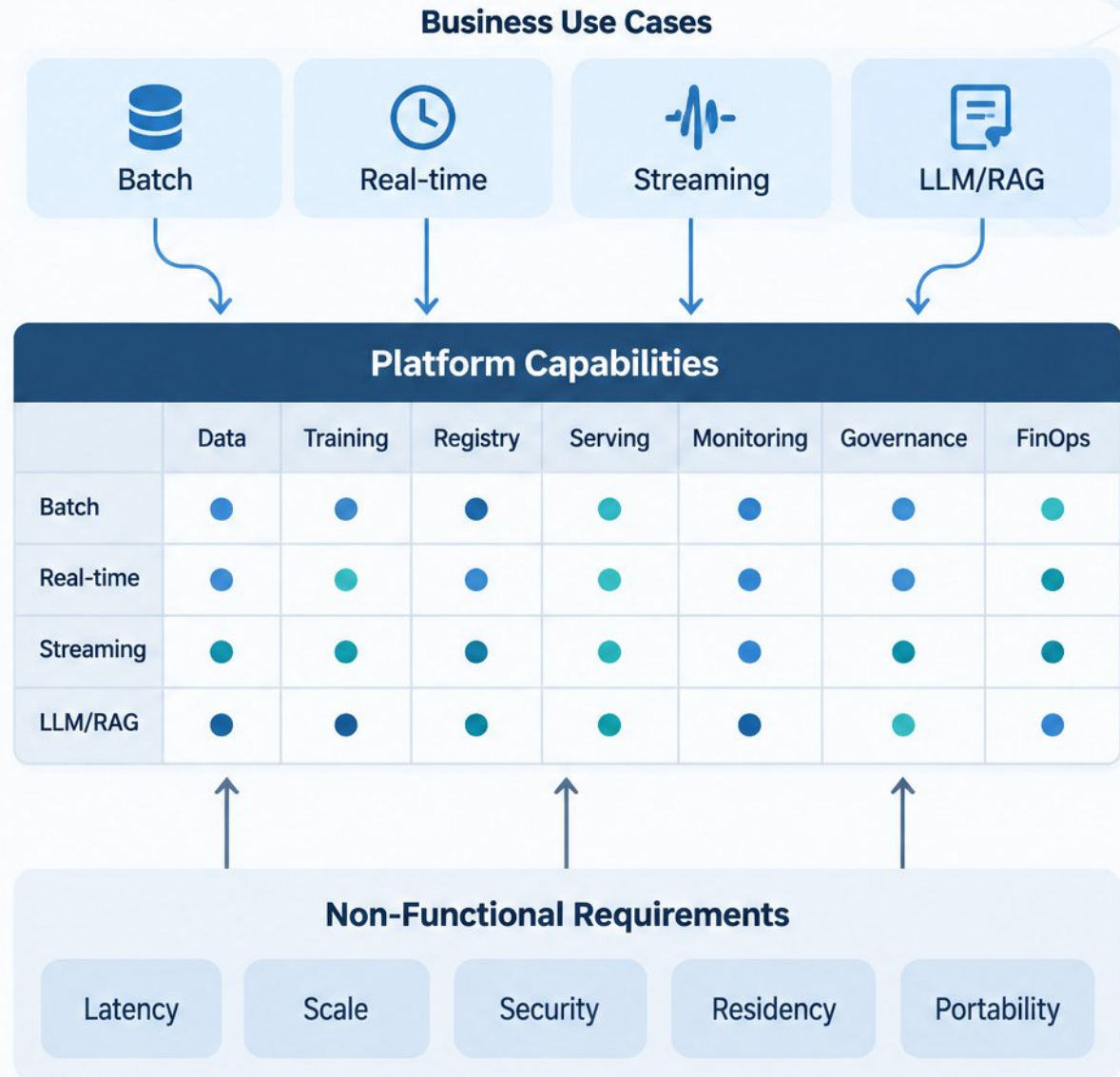
Separate non-negotiables from preferences so demos cannot blur the line



Align data science, ML engineering, SRE, security, compliance, finance, and vendor management



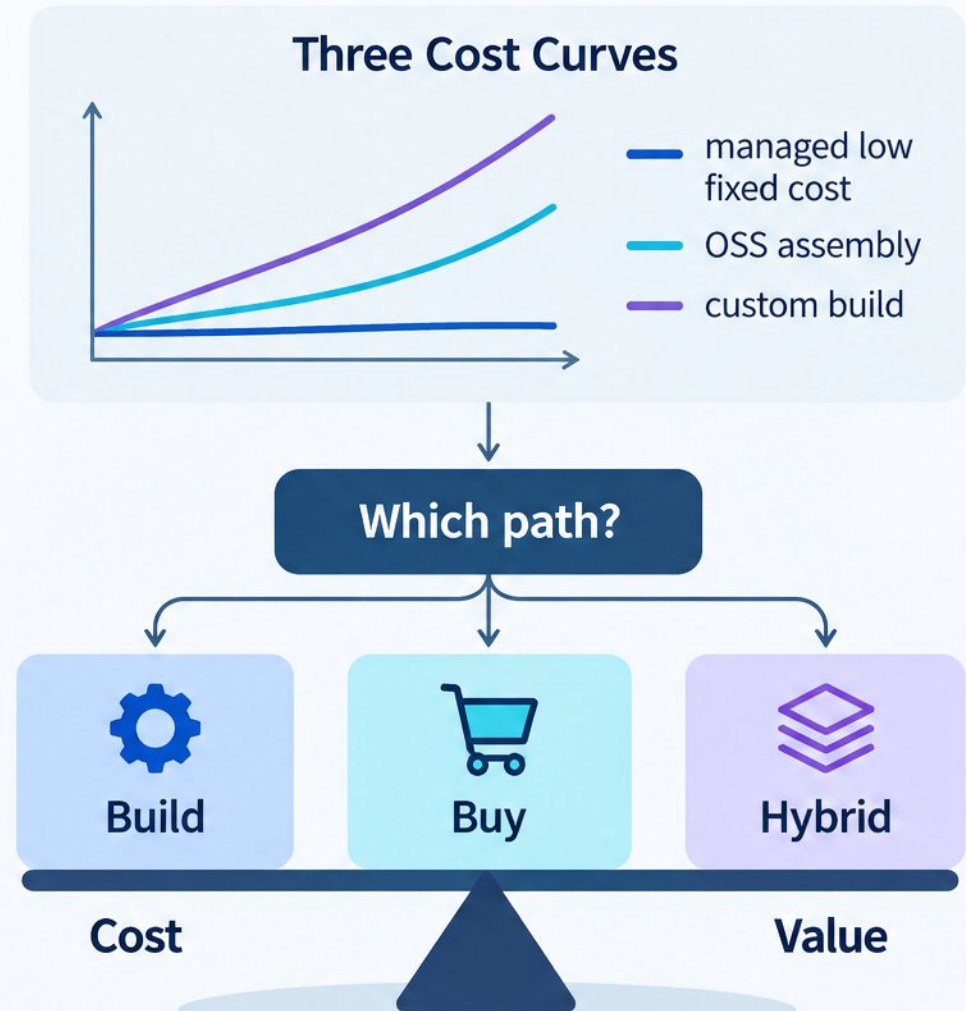
Capture cost drivers up front: compute, storage, egress, licensing, people



Build, Buy, or Hybrid: A Decision Framework That Survives Contact With Reality



- 1 Three cost curves: managed low fixed cost, OSS assembly, custom build
- 2 Teams under fifteen ML engineers almost always buy managed platforms
- 3 Build only with three conditions: real differentiation, 5+ platform engineers, GPU spend in millions
- 4 Hybrid dominates: buy generic 80% of lifecycle, build thin differentiator layer
- 5 TCO counts fully-loaded engineering, 20-30% yearly maintenance, hidden egress and idle endpoints
- 6 Keep feature definitions and registry contracts portable so the decision stays reversible



Reference Architecture: Planes, Boundaries, and Integration Contracts



Four planes: data, feature, model, serving; monitoring closes the training loop



Separate control plane from data plane: security, scaling, vendor substitution



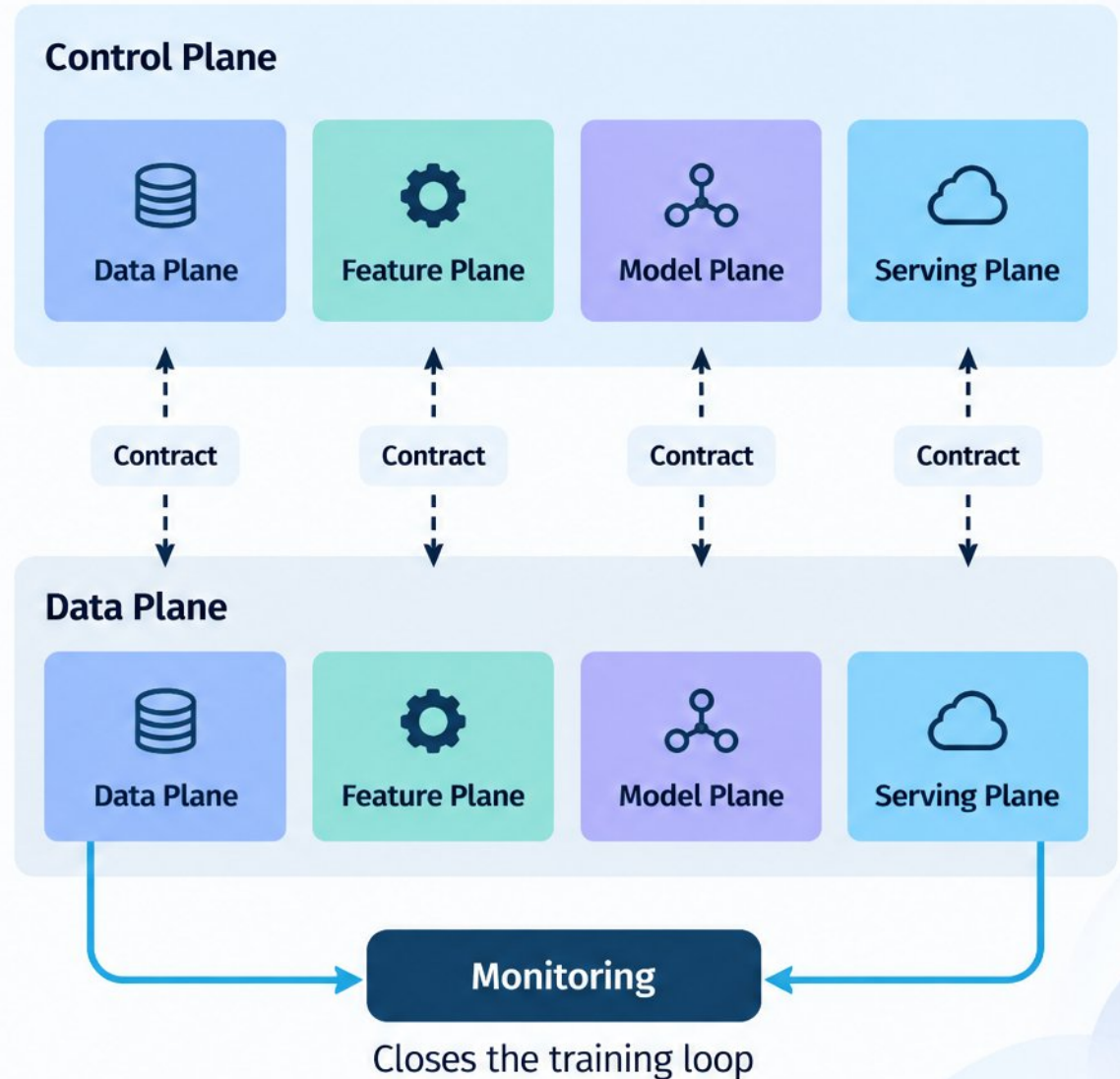
Six boundary crossings are the platform: most incidents break a seam, not a component



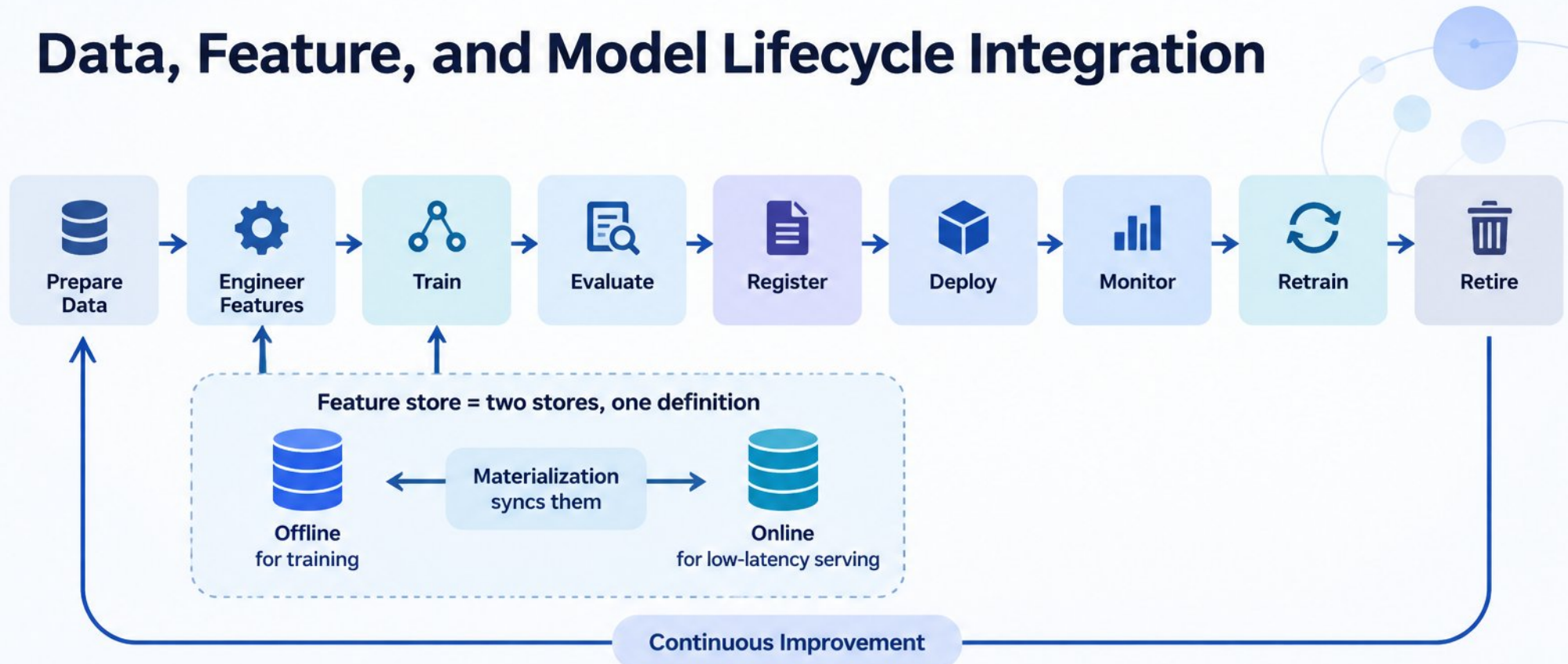
Multi-tenant, multi-team, multi-environment: dev, stage, prod, region, business unit



Real-time and streaming: online inference, feature freshness, feedback loops



Data, Feature, and Model Lifecycle Integration



- Lifecycle: prepare data, engineer features, train, evaluate, register, deploy, monitor, retrain, retire
- Feature store = two stores, one definition: offline for training, online for low-latency serving
- Materialization syncs them; freshness budget is the key operational knob
- Two silent killers: training-serving skew and point-in-time leakage — nothing errors, the model is just wrong
- Fix: one shared transform definition plus as-of (point-in-time) joins
- Registry and lineage enable reproducibility, audit, and rollback by alias, not redeploy
- LLMs add prompt versioning, eval datasets, guardrails, and retrieval pipelines

Serving, Monitoring, and Operating ML Systems at Scale



Latency requirements come from the consuming action, never the model



Serving is application infrastructure: batch, online, streaming, edge, serverless



GPU utilization averages 23–35% in production; watch VRAM, power efficiency



Monitoring: feature drift, prediction drift, p99 latency, freshness, delayed labels



Canary and shadow rollouts as baseline; define degradation when features fail



Governance, Security, and Responsible AI Controls in the Platform



Platform-embedded governance:

access control, approval workflows, audit trails, policy as code



The AI register is foundational —

ISO 42001, NIST AI RMF, EU AI Act all assume it



EU AI Act high-risk obligations

apply from 2 August 2026;
log retention at least 6 months



Security: secrets management, network isolation, least privilege, artifact scanning, supply chain



Responsible AI: bias testing, explainability, human oversight, model cards



Balance governance with developer velocity: pilot-scale governance fractures across business units



Cost, Performance, and Scalability Tradeoffs



Cost architecture: compute, storage, egress, licensing, team overhead — managed versus self-built



Showback before chargeback; per-token, per-tenant, per-experiment allocation



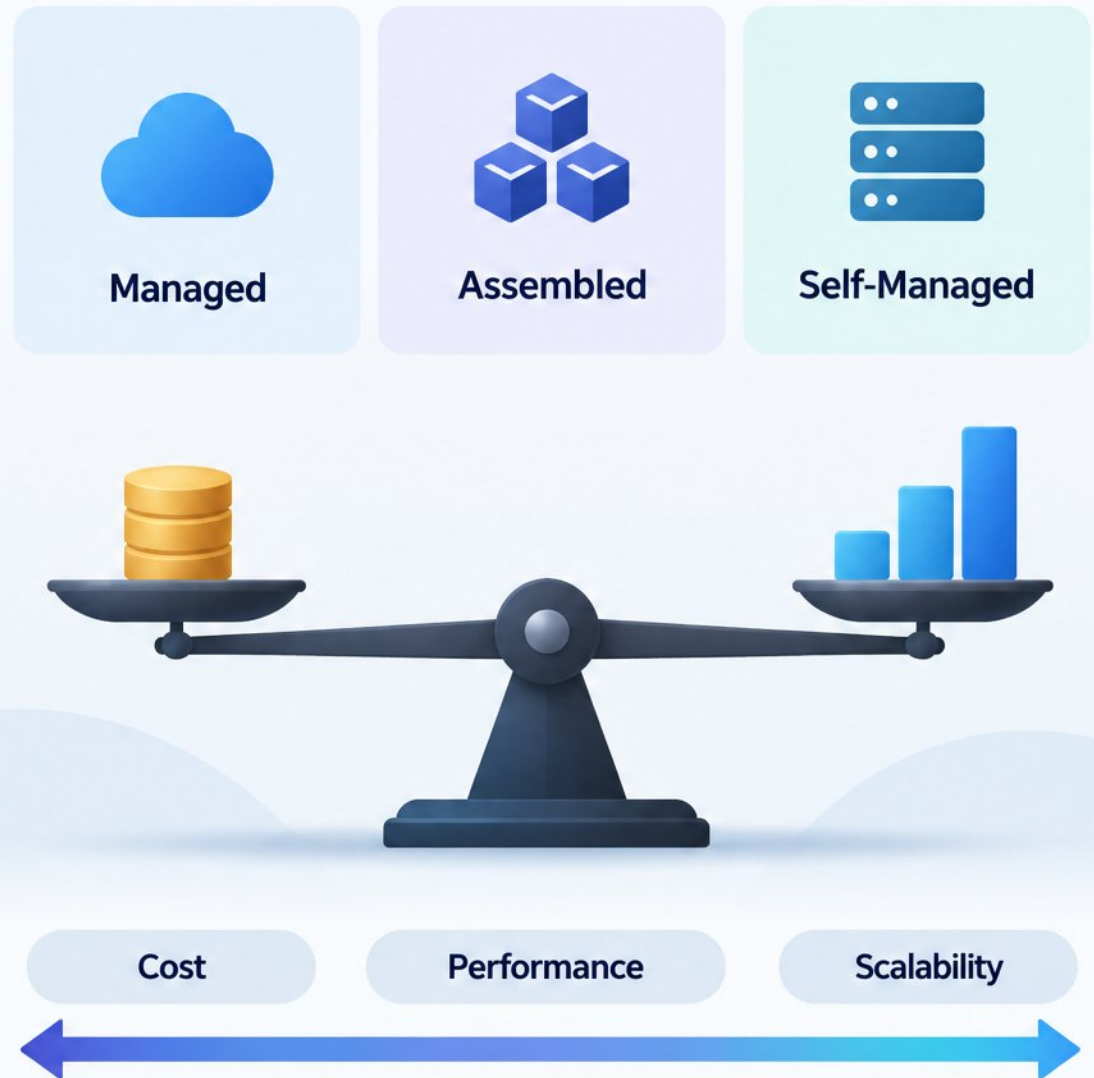
GPU utilization lies — track SM activity and cost-per-inference instead



Commitment traps: spot and Capacity Blocks excluded from Savings Plans and RIs



Optimize when unit economics break; accept cost for speed and reliability otherwise



Evaluation Scorecard:

Turning Requirements Into a Comparable Number



Build the weighted scorecard from requirements, architecture fit, risk



Default weights: technical capability, governance, security, compliance, commercial terms



Score on evidence: vendor claims + reference customers + pilot results



Treat any requirement untestable in a demo as unverified



Decision bands: strong fit, shortlistable, conditional, no-go



Two independent evaluators reconcile, so scores stay auditable



Proof of Concept, Vendor Questions, and Contract Levers



POC: representative workloads, success metrics, hard go/no-go gates; 4–6 week window



Test dirty data, failed jobs, permission changes, rollback — clean demos prove nothing



Demand evidence: roadmap stability, deprecation policy, bias and red-team reports



Red flags: exclusive models, no eval methodology, no BYOK, no rollback path



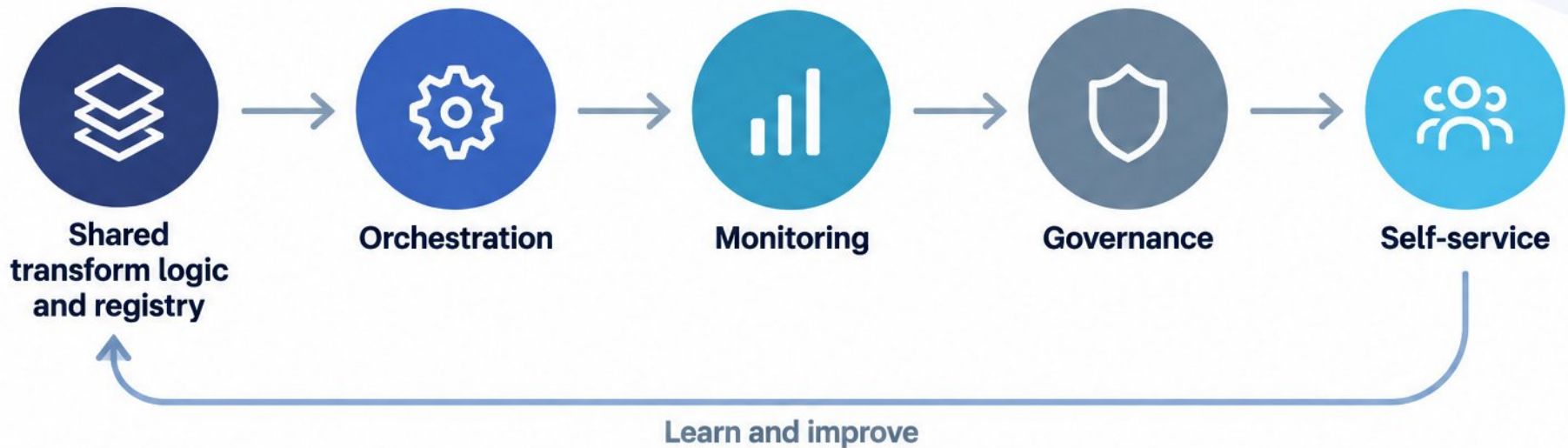
Contract levers: portability, training-data exclusion, audit rights, exit assistance



Verify commitments cover the capacity you buy; model the worst realistic case



Adoption Roadmap: Build the Seams Before You Build the Planes.



- Sequence first: shared transform logic and registry, then orchestration, then monitoring, then governance
- Embed platform engineers in product sprints; let users rank the roadmap
- Pilot one workload end to end, then expand by model family, not department
- Measure time-to-first-production-model, adoption and platform NPS, cost per inference
- Treat every workaround as product evidence; “shipped” and “adopted” are different criteria
- Organize the handbook around practitioner workflows, not features

Case Discussions: Contrasting Rollouts and Their Tradeoff Decisions



Case A



- Case A: managed foundation plus best-of-breed components, portable contracts, shipped in months

Case B



- Case B: adoption failure — technically sound platform, rational workarounds, low usage read as a comms problem

Case C



- Case C: 400 models' infrastructure logic rebuilt to dodge per-model markup, then re-platformed

Case D



- Case D: request-based rate limits, shared auth connection pools, queue-based rebuild



- Early failure signatures: unowned metadata, governance bolted on, composite reliability collapsing across chained components

Action Planning, Self-Assessment, and Open Discussion



Assess state against seven layers and boundary contracts: which seams have broken?



Start with experiment tracking plus registry — low lock-in, high leverage



Write down six boundary contracts before naming any product



Carry three fields from sprint one: feature owner, training snapshot, audit entry



Draft weighted scorecard; schedule evaluation review date



Template: current state, first increment, owners, success measures, revisit trigger

Action Planning Worksheet

Current state	First increment	Owners	Success measures	Revisit trigger

PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/0494e588/public