

Data Engineering Platform: Selection, Architecture, and Tradeoffs



Course purpose: a working decision framework for platform selection and tradeoffs



Core tension: every choice balances governance, speed, cost, reliability, and operating capability



Outcomes: compare archetypes, design architecture, run build/buy/assemble decisions



Course map: landscape, reference architecture, scorecard, TCO, security, operating model, migration



Shared vocabulary: archetype, open table format, medallion, federation, catalog, TCO, lock-in, gold path



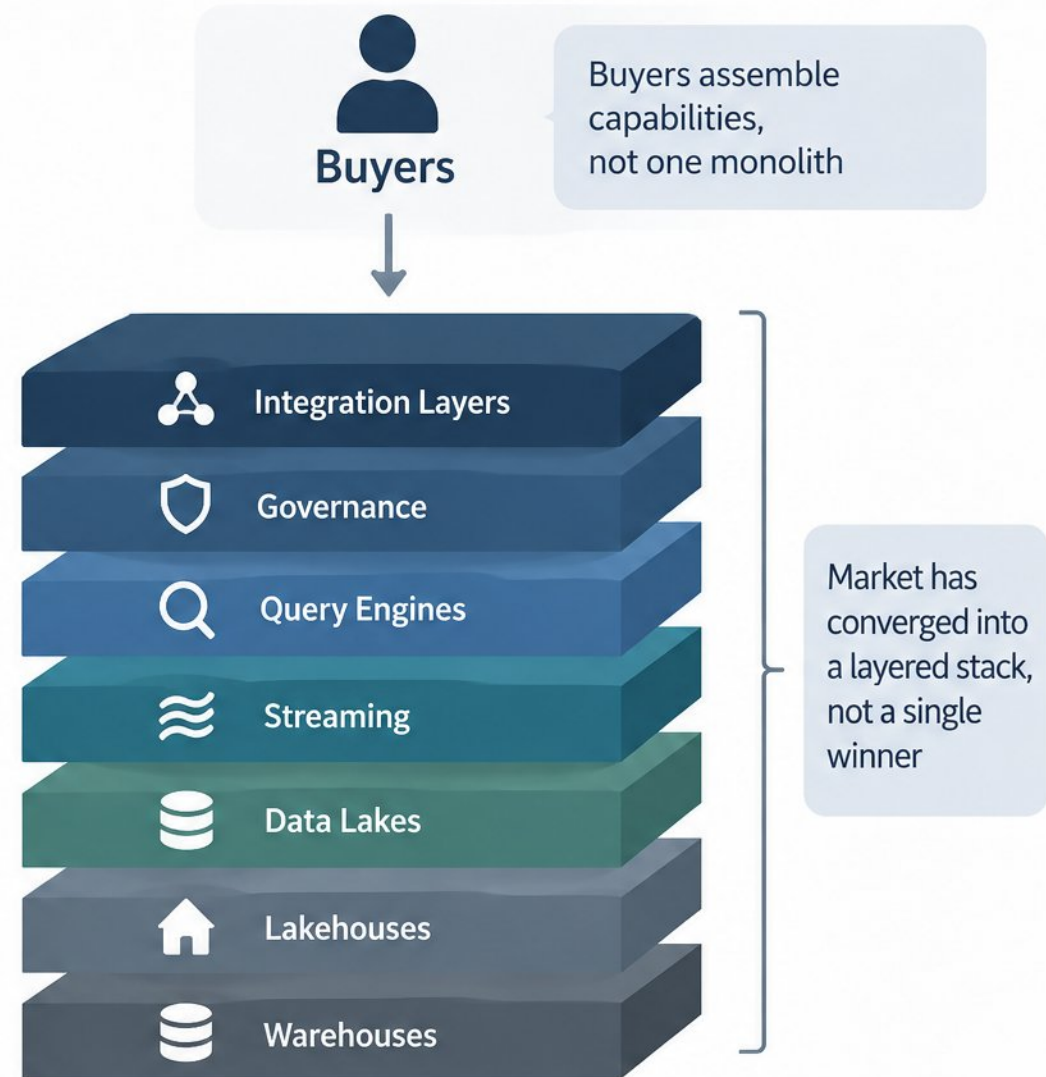
2026 framing: composable ecosystems replace single-architecture bets; AI workloads are the forcing function



Platform Landscape and Buyer Context in 2026

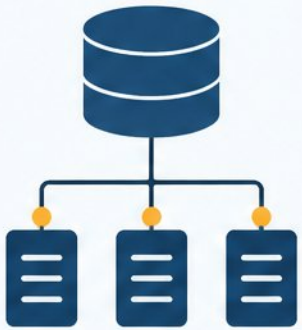
Learning outcomes and course map come next; first, the platform landscape and buyer context for 2026.

- Platforms compared: warehouses, lakehouses, data lakes, streaming, query engines, integration layers
- Buyers assemble capabilities, not one monolith
- Market has converged into a layered stack, not a single winner
- Watch TCO, lock-in, operational load, feature-checklist buying, demo-driven decisions



Platform Archetypes: Warehouse, Lakehouse, Lake, Mesh, Fabric

Cloud warehouse



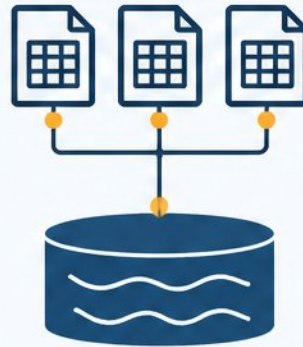
- Cloud warehouse: coupled storage and compute, strong SQL concurrency, governed by default

Lakehouse



- Lakehouse: open table formats on object storage, ACID, multi-engine access

Raw lake



- Raw lake: cheapest per byte, schema-on-read — no transactions, no governance

Mesh



- Mesh: ownership model — domain ownership, data as product, federated governance

Fabric



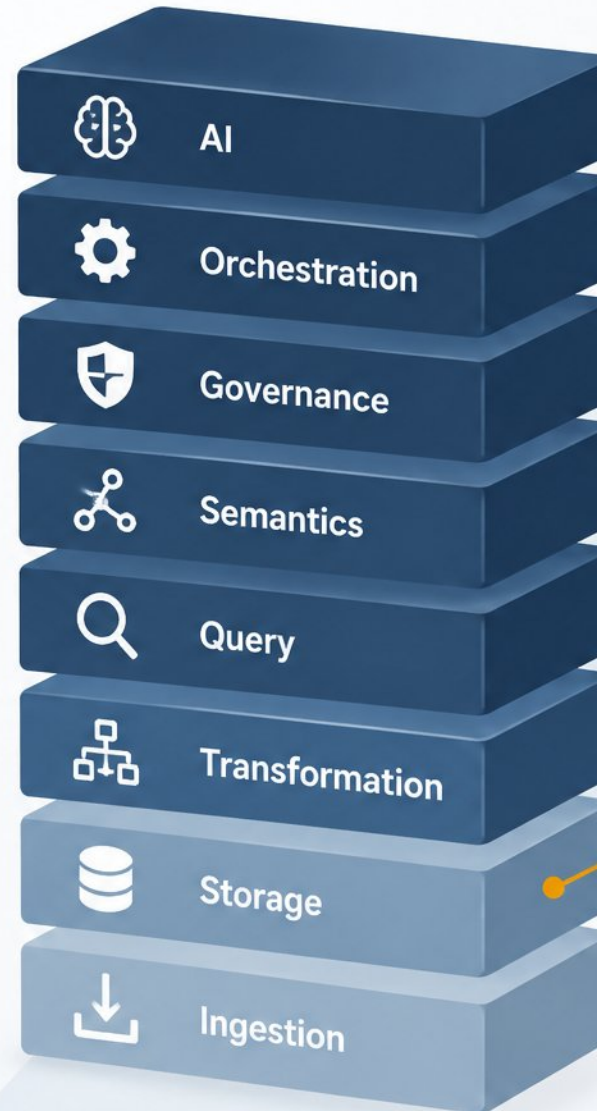
- Fabric: metadata and integration layer weaving disparate systems



2026 synthesis: combine lakehouse storage, fabric metadata, and mesh ownership

Reference Architecture: Layers, Ingestion, and Storage Decisions

- 1 Eight functional layers: ingestion, storage, transformation, query, semantics, governance, orchestration, AI
- 2 Federate by default, ingest by exception: every copy must be governed and kept current
- 3 Table format is the most consequential, only theoretically reversible decision; choose deliberately
- 4 Partition for query patterns; schedule compaction from day one
- 5 Build order: query and federation, then catalog and governance, then storage



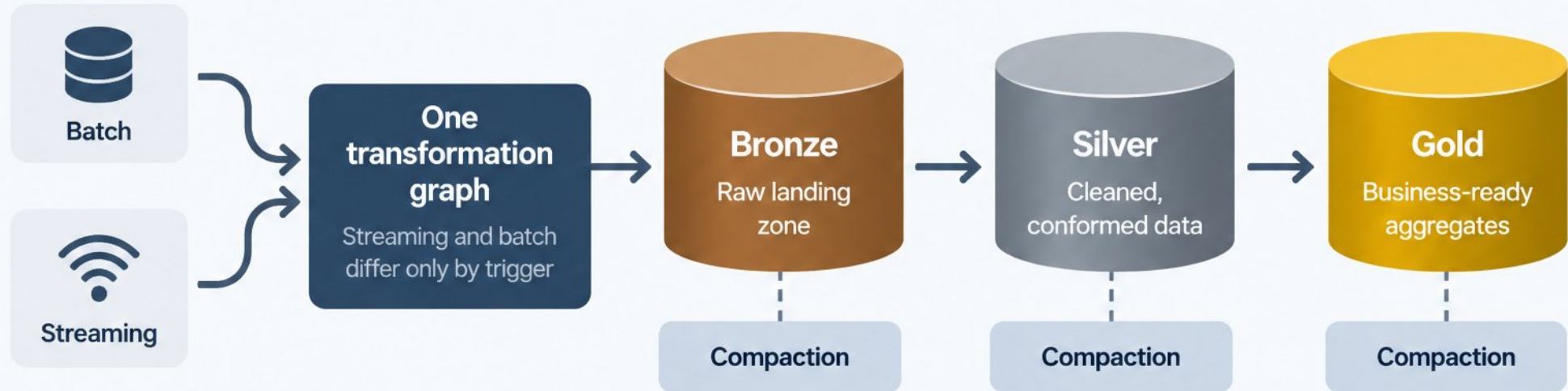
**Federate by default,
ingest by exception**

every copy must be
governed and kept current



Table format is the
most consequential,
only theoretically
reversible decision;
choose deliberately

Processing Patterns: Batch, Streaming, and the Medallion Model

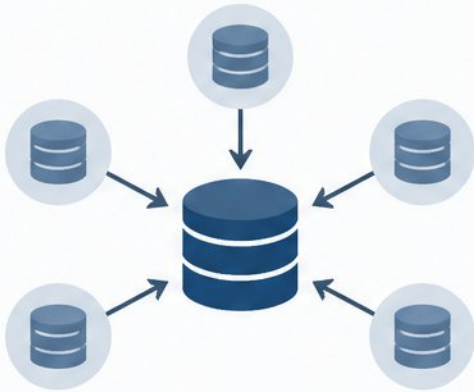


- 1 Bronze: append-only raw landing zone, faithful to source, replay source of truth
- 2 Silver: cleaned, conformed, deduplicated, typed — never clean in bronze
- 3 Gold: business-ready aggregates and dimensional models for BI, ML, dashboards
- 4 One transformation graph, streaming and batch differ only by trigger
- 5 MERGE handles upserts and SCD; event-time guard stops stale overwrites
- 6 Gold maintained incrementally via change feeds, not nightly full rebuilds
- 7 Avoid Lambda's dual paths; plan compaction and latency/cost targets per tier

Federated and Centralized Architecture: Choosing Where Truth Lives

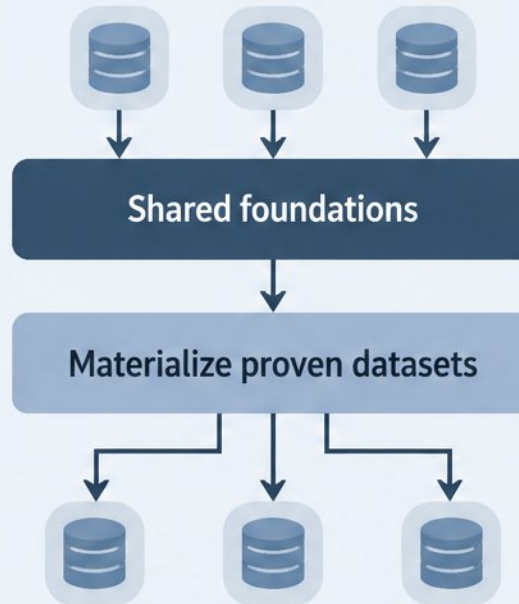
Centralized

- Centralized: uniform policy, one audit trail, simpler compliance validation
- Centralized optimization wins on known, repeated, schema-stable queries



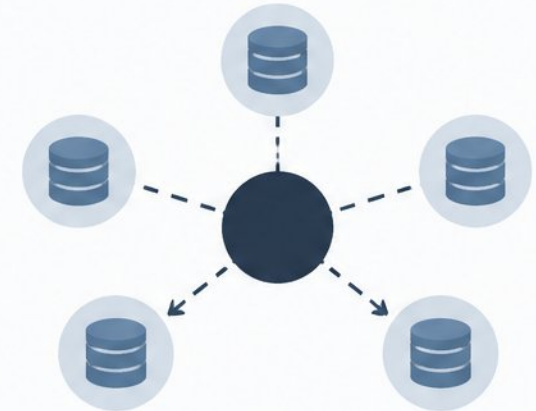
Tiered hybrid

- Tiered hybrid: centralize shared foundations, materialize proven datasets, federate the rest

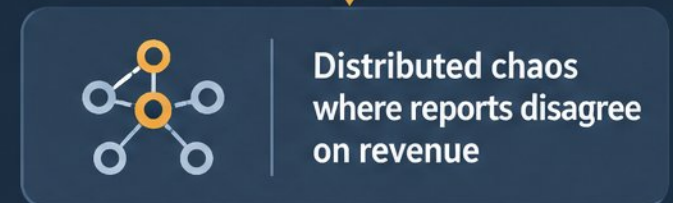
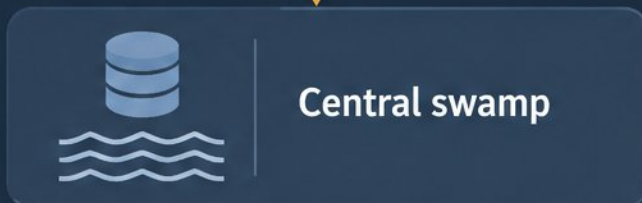


Federated

- Federated: live operational state, no copy to synchronize, residency by construction
- Federation limits: network-bound joins, concurrency ceilings, read-only access



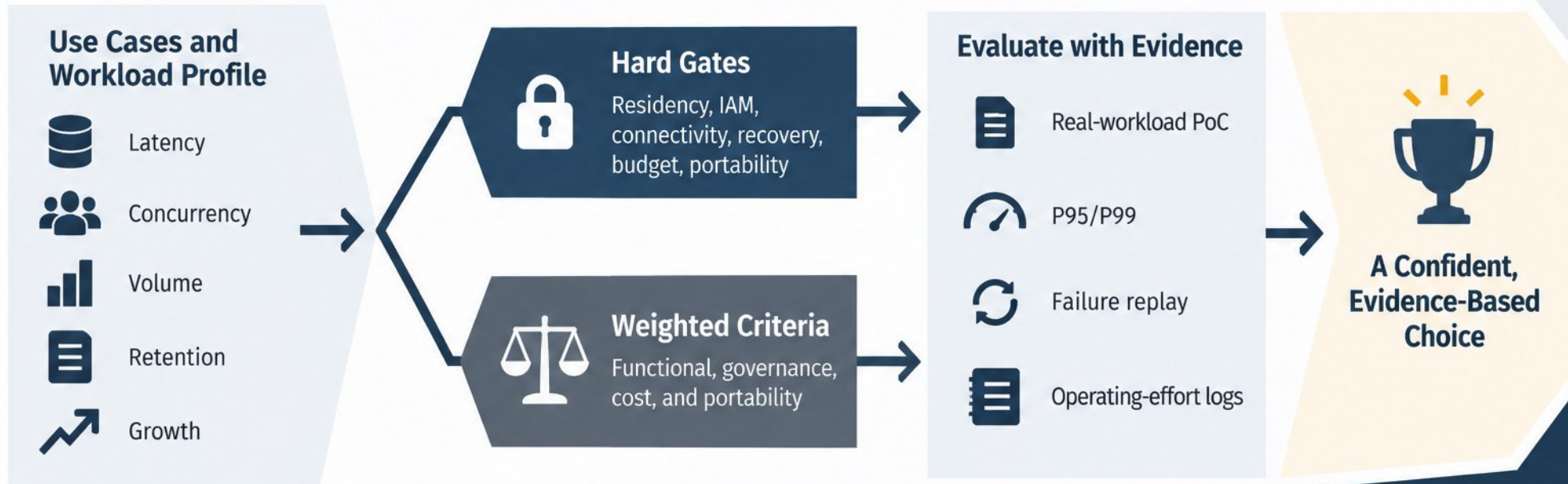
Failure modes



Selection Criteria and an Evidence-Based Evaluation Scorecard

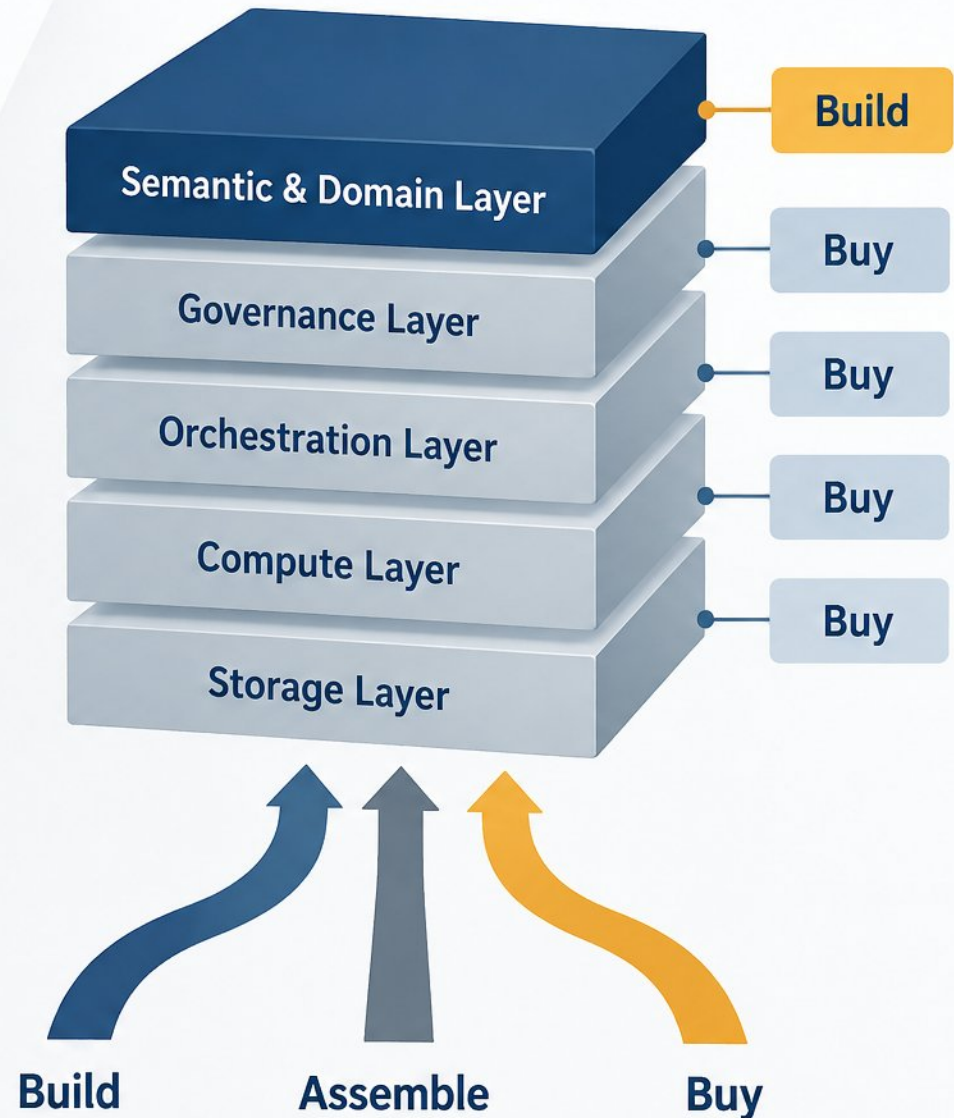
- Define use cases and workload profile first: latency, concurrency, volume, retention, growth
- Split hard gates from weighted scores: residency, IAM, connectivity, recovery, budget, portability
- Set weights before vendor workshops — weighting after scores is reverse-engineering
- Score functional, governance, cost, and portability criteria on evidence, not promises
- Methods that hold up: real-workload PoC, P95/P99, failure replay, operating-effort logs

FROM
REQUIREMENTS
TO A DEFENSIBLE
DECISION



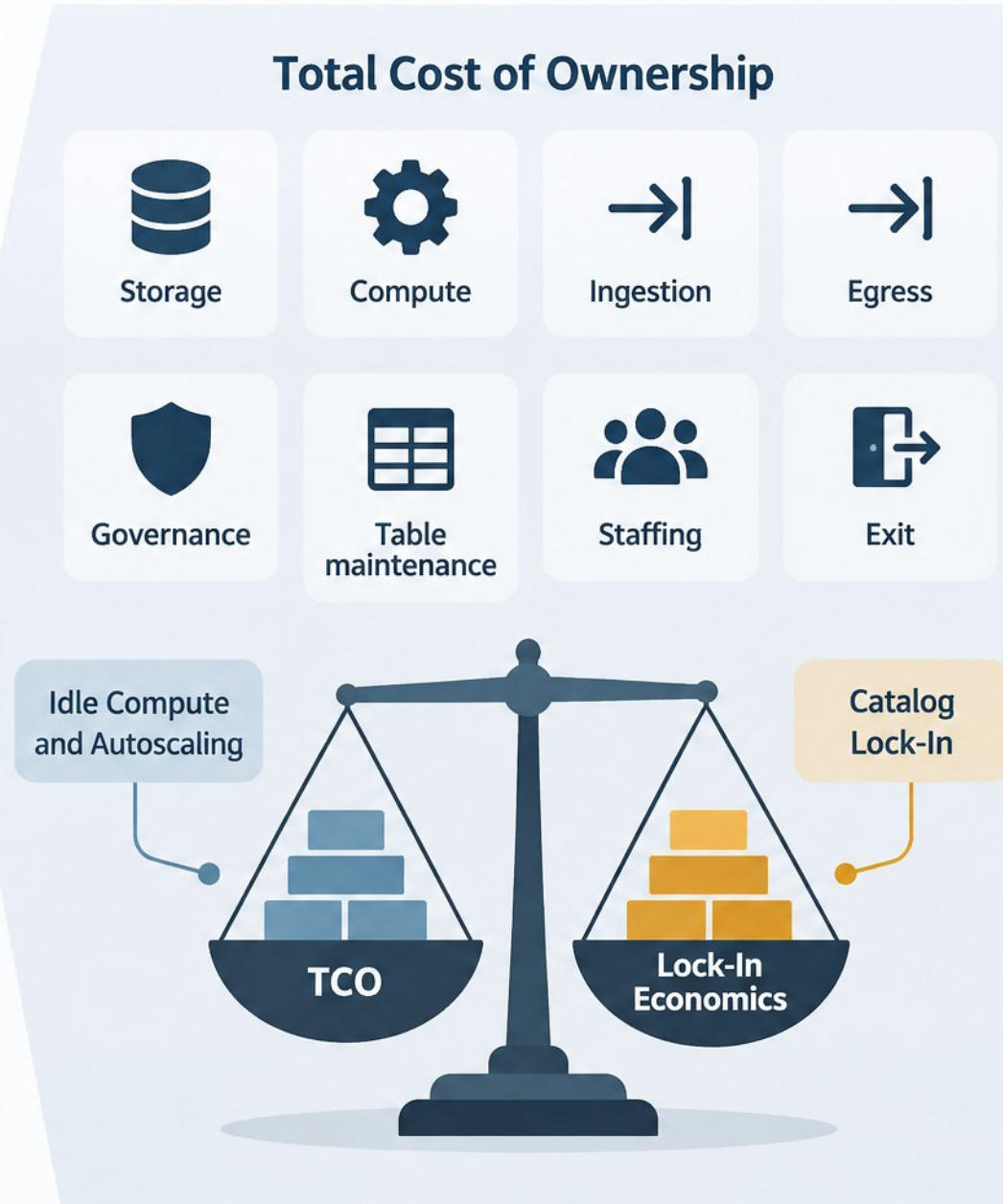
Build vs. Buy vs. Assemble: A Layered Decision

- Stop treating build vs. buy as one binary bet
- Buy commodity layers: storage, compute, orchestration, governance
- Build the opinionated semantic and domain layer yourself
- Managed services win on speed; egress costs surprise buyers at month 18
- Build is defensible when the platform is the product
- Assembly trades control for integration and upgrade ownership
- Plan exit early: open formats, metadata export, portability clauses



Total Cost of Ownership and Lock-In Economics

- Price the same 30–90 day production workload across candidates, not advertised rates
- Model full TCO: storage, compute, ingestion, egress, governance, table maintenance, staffing, exit
- Measure idle compute and autoscaling explicitly — one of the largest hidden line items
- Bundled vs. unbundled trap: raw storage rates pretend service work is free
- Below the crossover volume, self-managed lakehouse engineering exceeds compute markup avoided
- Lock-in now lives in the catalog, not just storage format
- Model commitment discounts separately, validated against expected utilization



Tradeoff Matrix: Performance, Cost, Governance, and Speed

BALANCE
DECISIONS
BUILD FOR WHAT'S NEXT



Doubling capacity costs more than it delivers in performance



Batch, micro-batch, streaming trade freshness against complexity and cost



Central governance gives control; domain autonomy gives speed and accountability



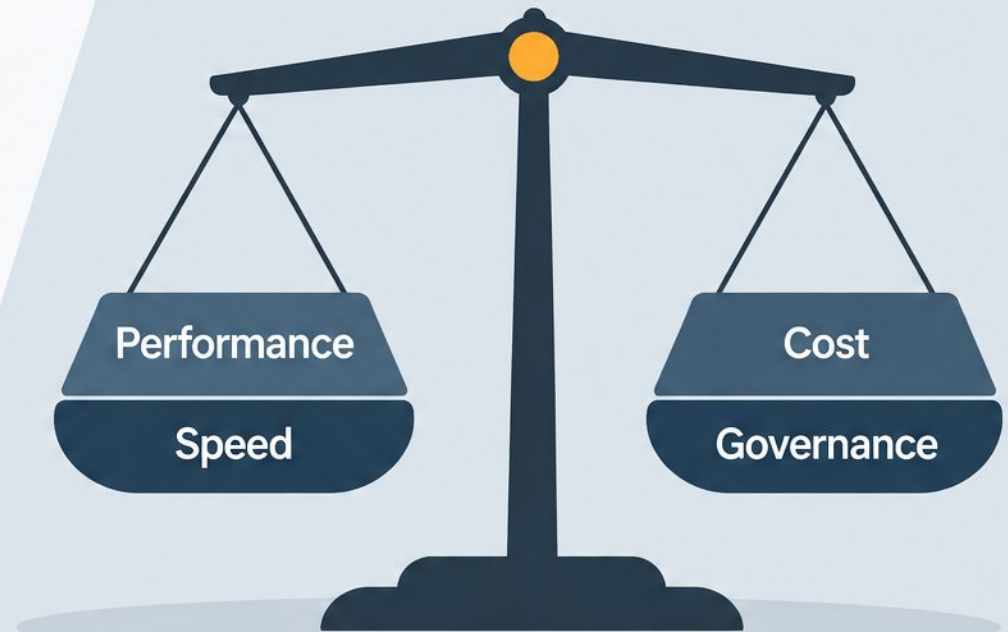
Open formats buy optionality, add compaction and snapshot-expiry work



Managed services speed time to value; portability decides later leverage



Log each tradeoff in a decision record with a named owner



PERFORMANCE



COST



GOVERNANCE



SPEED

TRADEOFFS TODAY
STRONGER TOMORROW

Security, Privacy, and Compliance Architecture



Shared responsibility: vendor secures the service; identity, policy, and evidence remain yours



Tag-based access control scales; per-table grants do not survive dataset growth



Row filters, column masks, and tokenization enforced across every query engine



Bypass risk: direct object-store access can circumvent fine-grained controls



Purpose-based access, retention, and erasure workflows tied to lineage



Auditability as evidence: lineage, access logs, policy version stamps

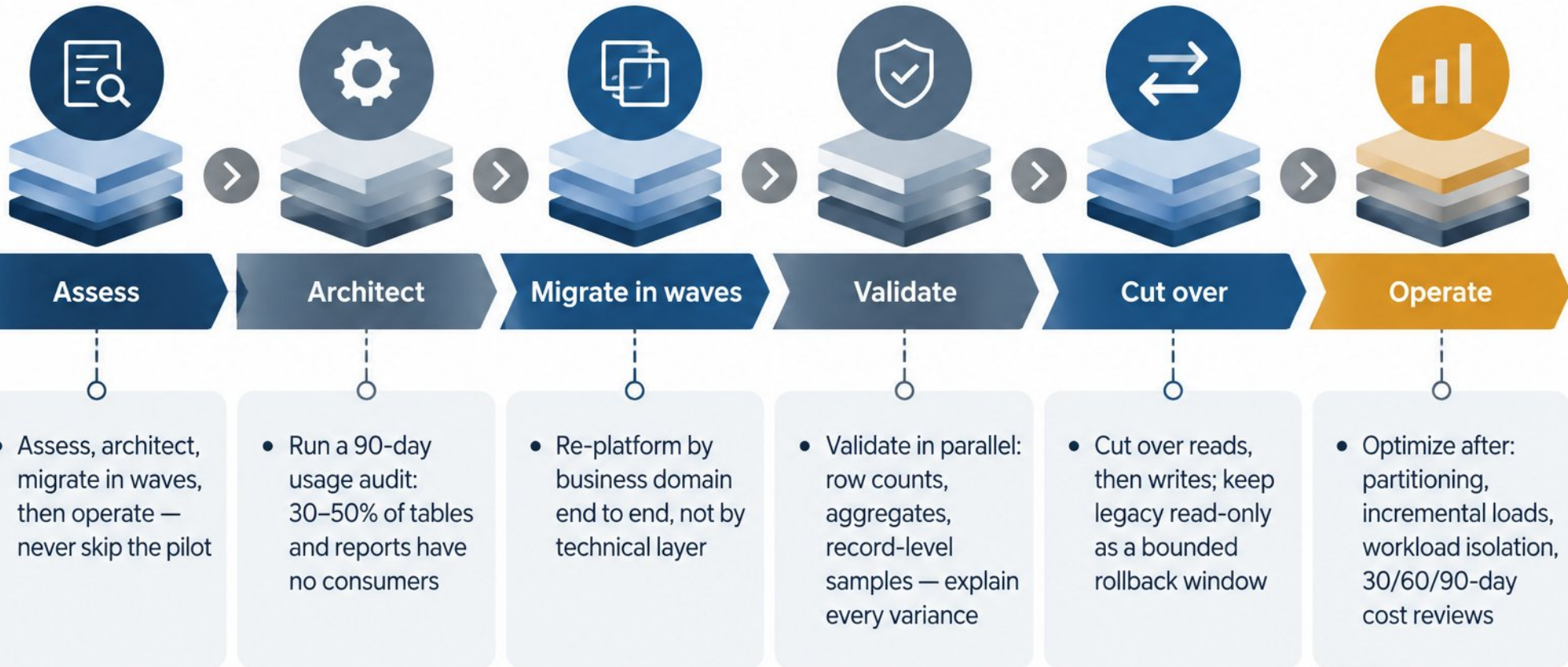


Operating Model, Team Topology, and Governance



- Choose topology: centralized, embedded, or hybrid platform-plus-embedded
- Platform as product: golden paths, published SLOs, substrate on-call
- Publish the ownership map before tooling decisions
- Federated governance: architecture review forum, ADRs, metrics council, security review
- Data product ownership: freshness, quality, availability targets, deprecation
- FinOps: shared baseline versus chargeback/showback, cost to named owners
- Platform health metrics: adoption, cost per query, time to onboard a source

Implementation Roadmap and Migration Strategy



PLAN > MIGRATE > SCALE > SUCCEED

A STRONGER
DATA FOUNDATION
FOR WHAT'S NEXT

Decision Workshop: Applying the Framework to a Live Scenario



Scenario: multi-domain enterprise with BI, streaming, AI, and residency limits



Hard gates and weights agreed before any candidate is scored



Every score backed by documented evidence, not vendor claims



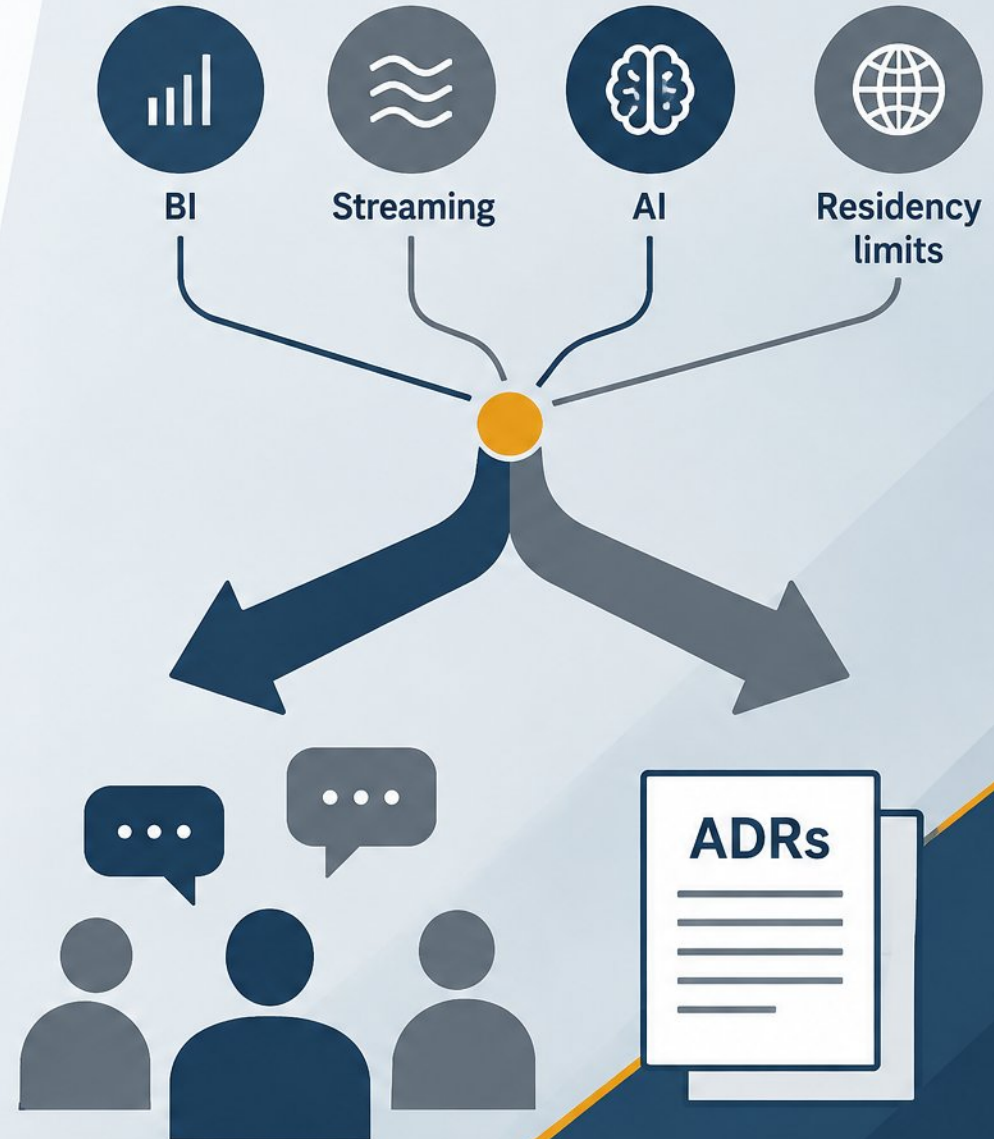
Assign owners: catalog, on-call page, shared platform funding



Record decision, rejected options, and reversal conditions as ADRs



Pitfalls: feature-checklist buying, no pilot, permanent parallel run



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/acf501ec/public