

Cloud Computing Platforms: Selection, Architecture, and Tradeoffs



This training covers platform selection, architecture design, and tradeoff analysis for real workloads



Audience: developers, technical teams, and cloud service selection owners



Structure: decision frameworks, architecture patterns, cost/performance/operations tradeoffs



Outcome: run a structured platform selection and justify architecture decisions



The 2026 Platform Landscape and Core Selection Drivers



- Hyperscale, regional, hybrid, multi-cloud, and private cloud categories



- Key selection dimensions: workload fit, compliance, skills, latency, cost, lock-in



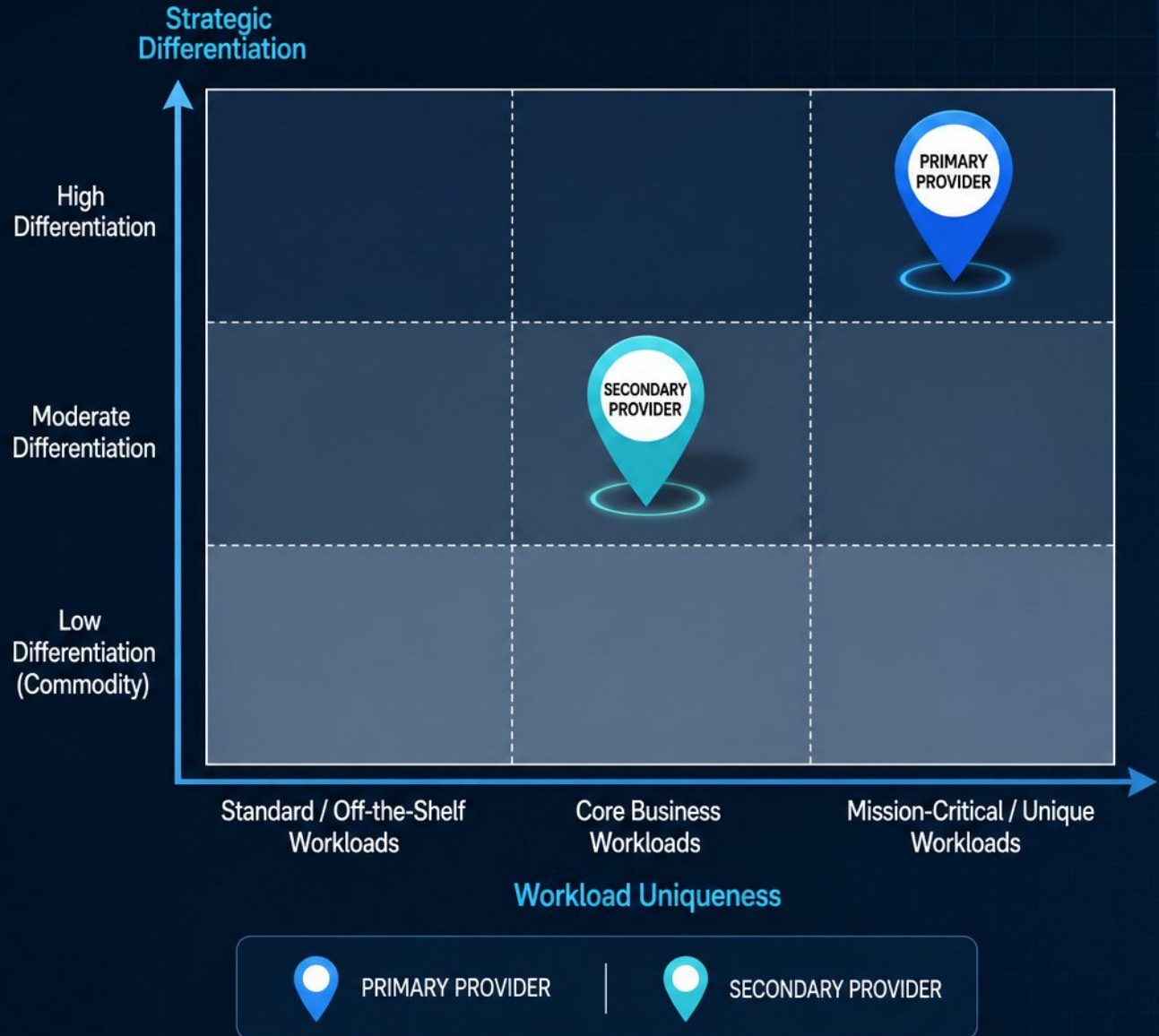
- Big three converge on core compute; differentiation is in AI, data, and ecosystem



- Avoid brand-first choices and feature comparisons without workload context

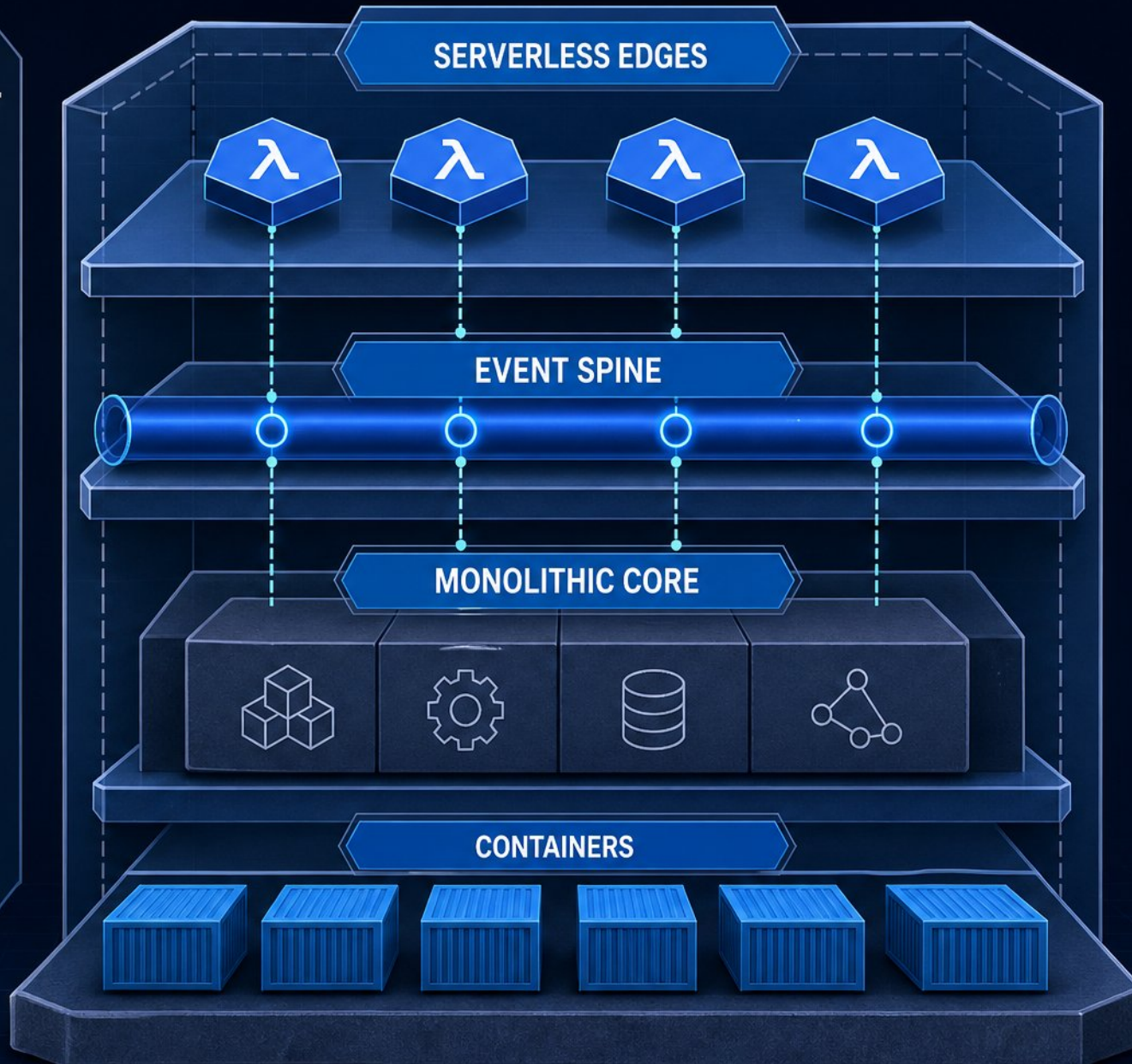
Workload-to-Platform Matching

- GCP excels in data analytics, AI/ML training, and managed Kubernetes
- Azure is optimal for Microsoft-stack enterprises with native OpenAI access
- AWS offers the broadest catalog, mature ecosystem, and regulatory compliance
- Apply the 80/20 rule: concentrate ~80% of workloads on a primary provider
- Avoid lock-in traps; choose provider-specific services only when they are true differentiators



Architecture Patterns for Cloud-Native Systems

- Core patterns: microservices, modular monolith, event-driven, serverless, containers
- Platform choice shapes architecture: managed services, serverless, Kubernetes vary by provider
- Adopt hybrid/multi-cloud for compliance, data gravity, best-of-breed, resilience
- Key risks: resilience, observability, data consistency, hidden dependencies



Multi-Cloud and Hybrid: Real Tradeoffs



- Multi-cloud: 30-60% operational overhead, 23% cost premium



- Adopt only for data residency, DR, or best-of-breed services



- Patterns: distributed workloads, active-active redundancy, cloud-agnostic layer

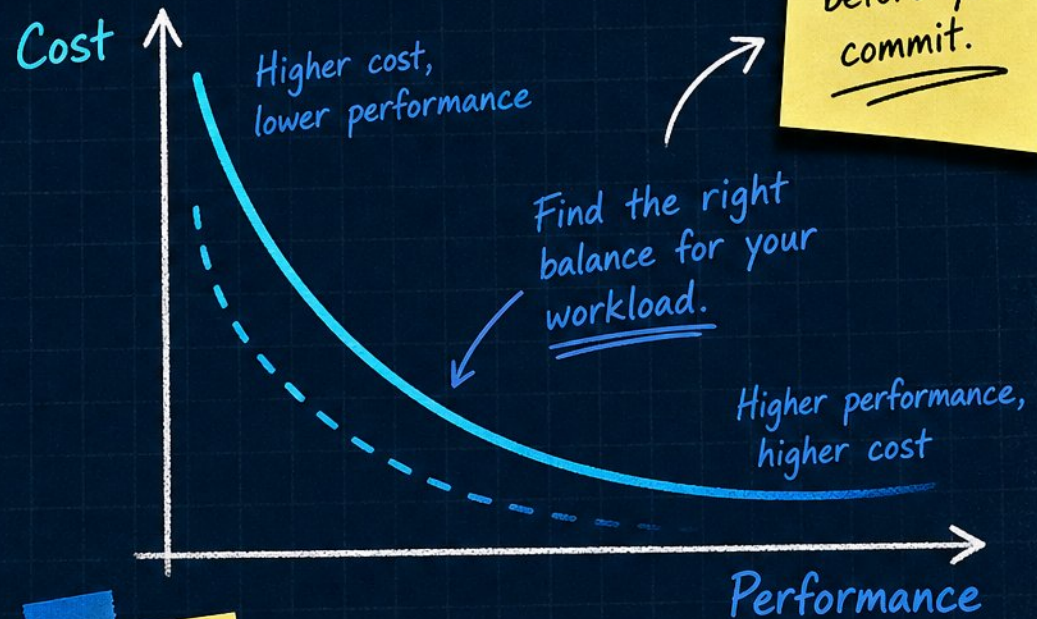


- Hidden taxes: egress fees, cross-cloud networking, identity fragmentation



Cost, Performance, and Operational Tradeoffs

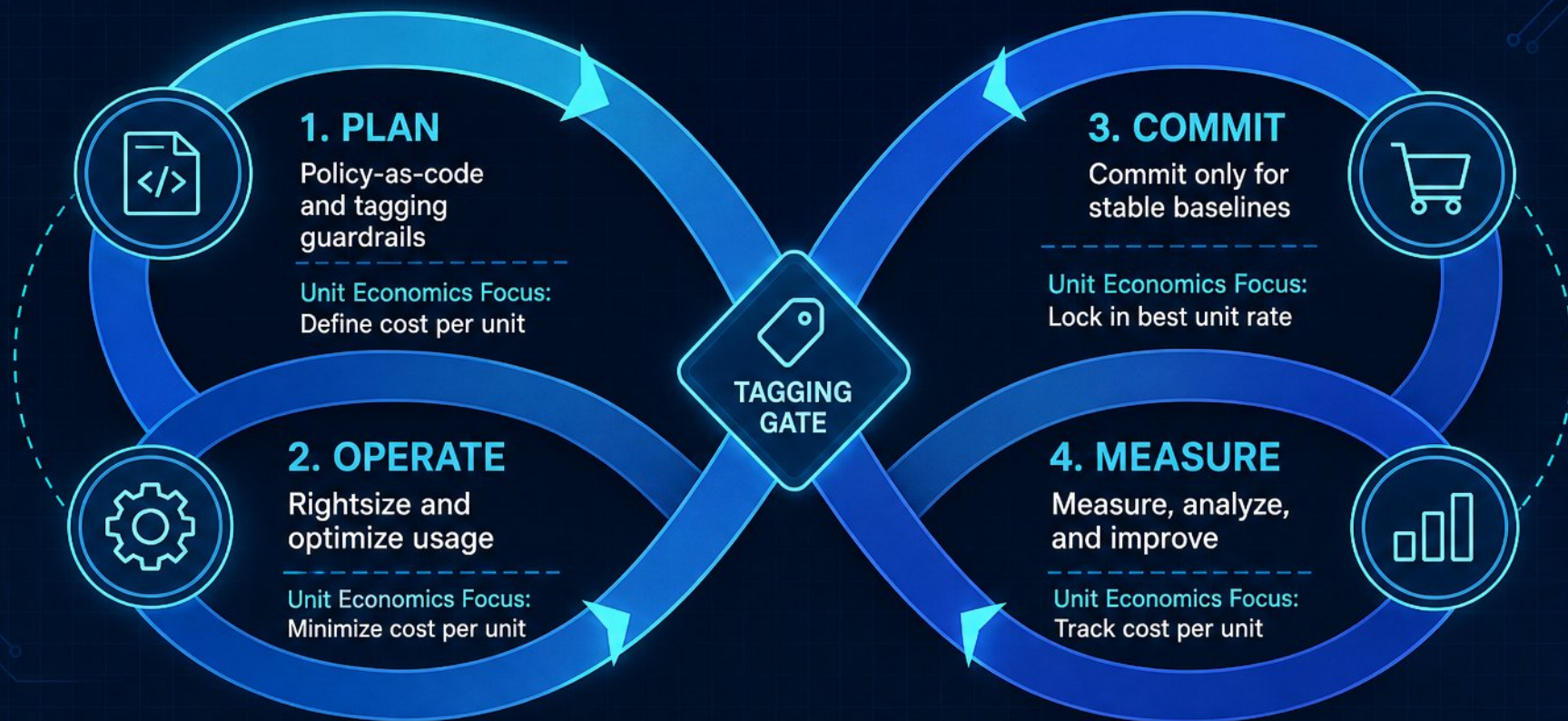
- Cost models: on-demand, reserved, savings plans, spot, egress
- Commit only for stable baselines after rightsizing
- Performance: compute types, storage tiers, latency, region
- Managed services shift operational burden; platform teams reduce load
- Track cost per outcome, not just total cloud spend



Use managed services to reduce ops burden.

★ Measure what matters: outcomes, not just spend.

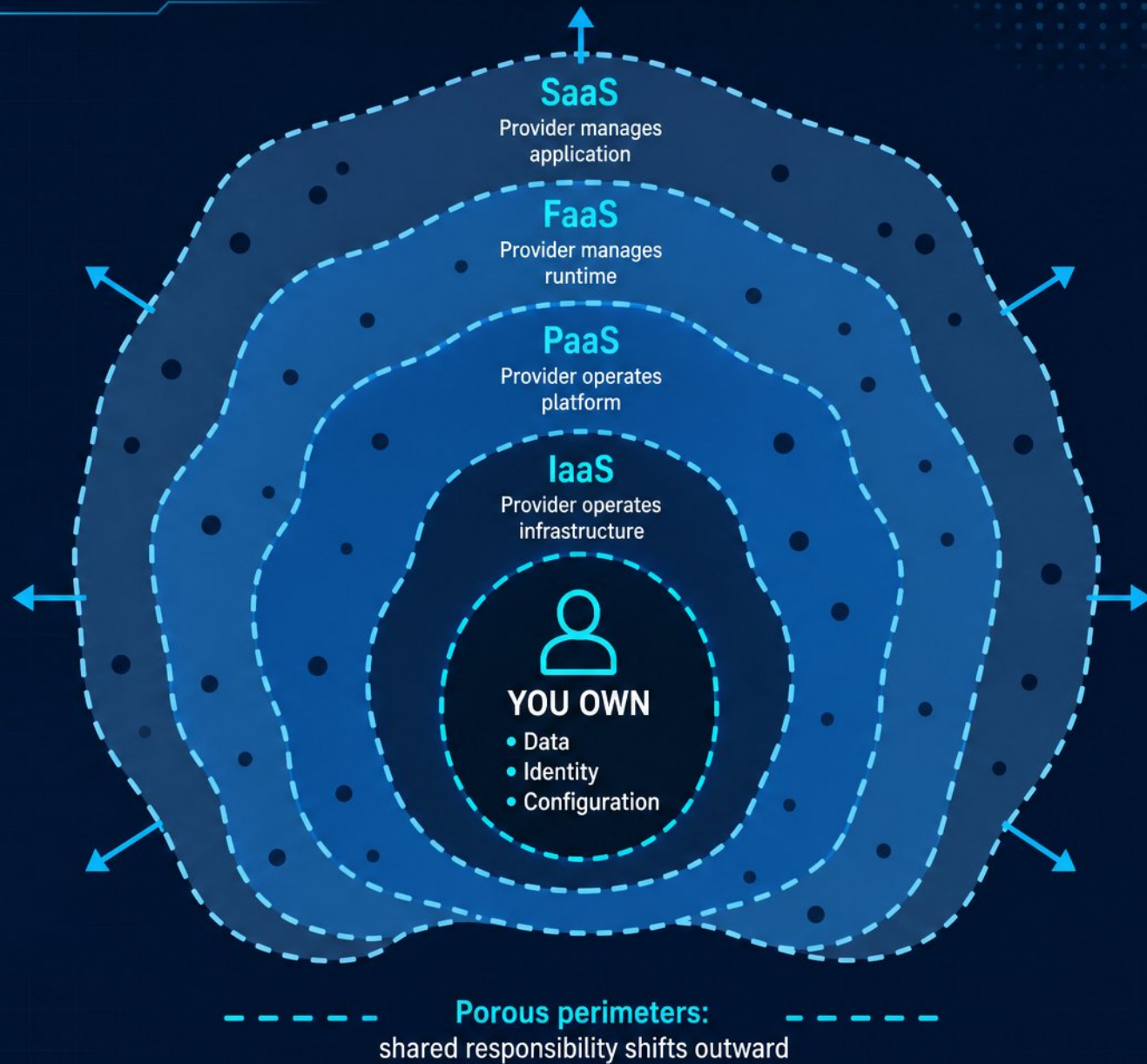
FinOps and Cost Governance



- - Enforce tagging via policy-as-code at resource creation
- - Rightsize first, then buy commitments only for stable baselines
- - Mature FinOps governance cuts cloud bills by 35-45%
- - Automated anomaly detection: 11 days to 2.1 hours response
- - Track cost per transaction or inference, not just the bill

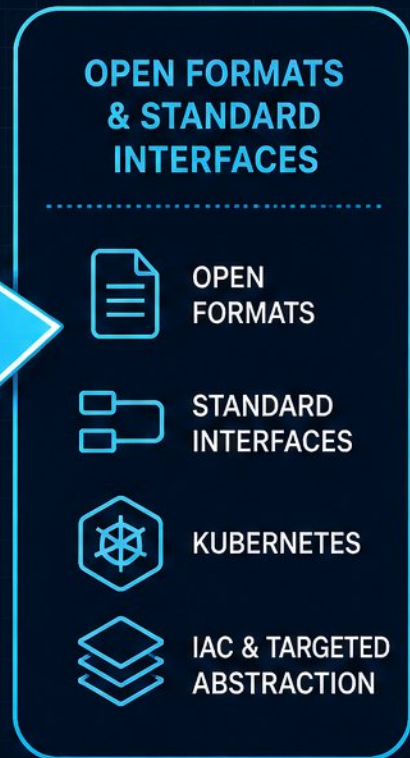
Security, Compliance, and Governance Constraints

- - Responsibility shifts with IaaS, PaaS, FaaS, SaaS; data, identity, and config always remain yours
- - Data residency and sovereignty requirements narrow platform choices significantly
- - Provider tooling varies: Azure Entra, GCP Assured Workloads, AWS GovCloud
- - Misconfigurations remain a leading breach cause; providers won't catch an open bucket



Vendor Lock-In, Portability, and Exit Strategy

- - Lock-in spans proprietary services, data egress, APIs, tooling, contracts
- - EU Data Act removes switching fees from Jan 12, 2027, not semantic lock-in
- - Portability: open formats, standard interfaces, Kubernetes, IaC, targeted abstraction
- - Accept lock-in where managed services add value; know your exit cost



Exit Planning and Contract Leverage

- - An exit covers data, schema, identity, network, operations, and shutdown
- - Negotiate portability clauses at signing: export SLAs, egress waivers, migration help
- - Run annual reconstitution drills: export, import, integrity check
- - A credible, priced exit preserves renewal leverage



A Structured Selection Framework

	 RUNTIME	 DATABASE	 WORKFLOW	 OBSERVABILITY	 SCALABILITY	 COST PREDICTABILITY	 PORTABILITY	 COMPLIANCE
								
								
								
								
								



- Build a weighted scorecard: runtime, database, workflow, observability, scalability, cost predictability, portability, compliance



- Involve developers, operations, security, finance, and leadership

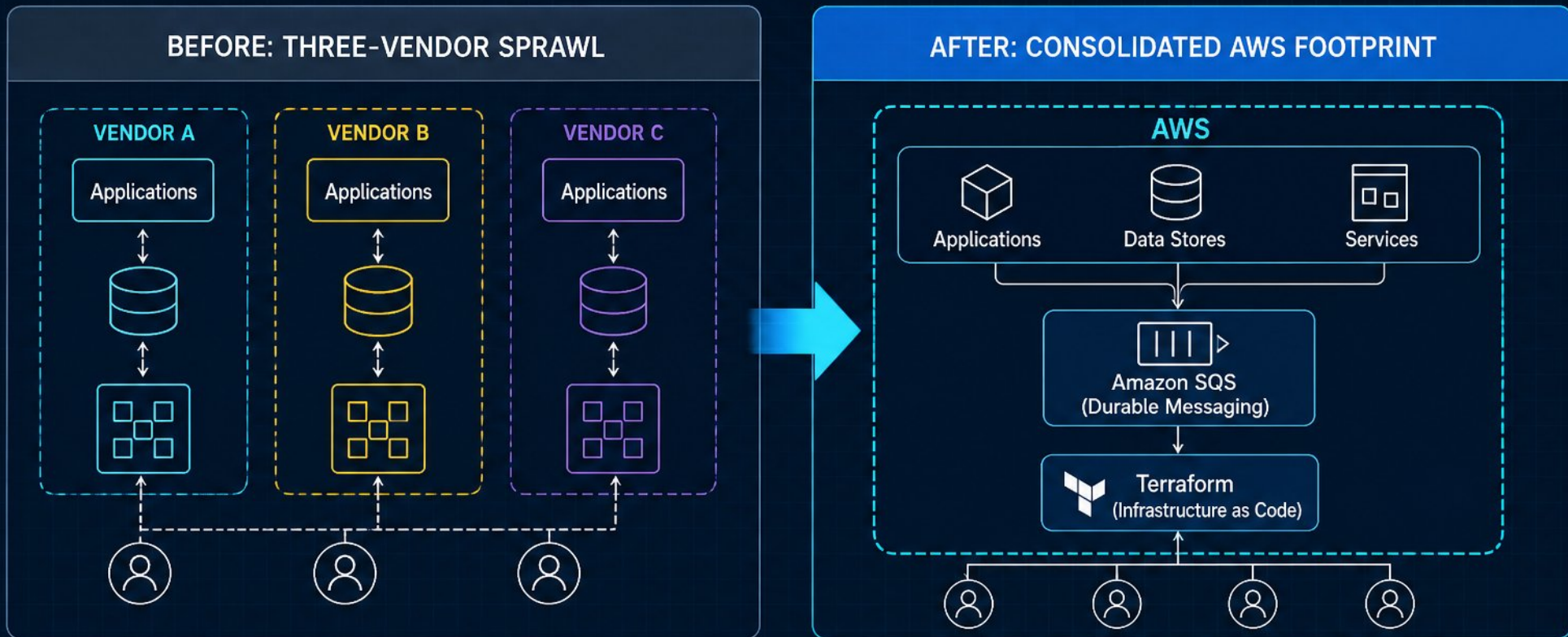


- Score objectively against workload requirements, not brand familiarity



- Review quarterly or when inputs change; document an exit path

Case Studies and Antipatterns



- Consolidation case: three vendors to single AWS footprint; durable SQS, Terraform enabled scale
- Cross-cloud analytics: single Iceberg copy on AWS; BigQuery reads in place via federated catalog
- Antipatterns: premature multi-cloud, over-abstraction, distributed monolith, global control plane dependency
- Evaluate reversibility: one-way doors vs two-way decisions; know your exit path

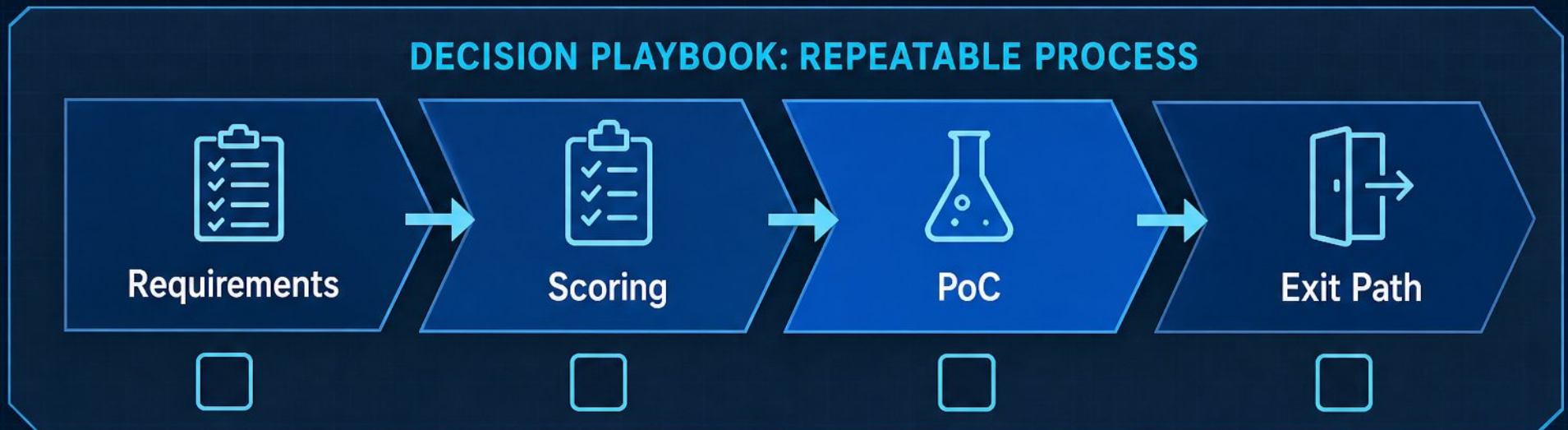
Tooling, Observability, and Day-2 Operations

- IaC, CI/CD, and platform engineering create consistent, repeatable environments
- Golden paths reduce cognitive load, cut support tickets by 40%
- Centralize observability with OpenTelemetry; avoid provider-native silos
- Track upgrades, incident response, capacity planning, and cost monitoring
- Pre-instrumented templates cut time to full observability to under 15 minutes



Decision Playbook and Next Steps

-  – Decision checklist: selection, architecture, and tradeoff questions
-  – Repeatable process: requirements, scoring, PoC, and exit path
-  – Next steps: inventory workloads, map skills, build scorecard
-  – Follow up: FinOps, OpenTelemetry, and annual exit drills



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/7ecd1267/public