Single platform or multi-engine? A decision framework that can say no

Capability is a set of gates. Need is scored separately. No amount of need buys you past a failed gate.

Version 1.1 · 19 September 2026 · Frank Garofalo · Chief Data Officer, Federal Civilian · written in a personal capacity

1 The capability gates

Clear all three, or run one platform. These are not scored and they do not trade off against each other — a team that cannot operate two estates safely will not be rescued by having a strong reason to want two.

Gate	Clearing it means	If you do not clear it
Platform engineering	At least one named engineer whose job is the platform itself, not the pipelines running on it.	Two upgrade cadences, two capacity models and two on-call surfaces, owned by nobody.
Governance	One owner for classification, access policy and lineage *across* engines, with authority to say no.	The mirror seam is where policy silently stops being enforced. You find out from an auditor.
FinOps	Capacity units and DBUs tracked separately and attributed to business units.	Two billing models blend into one number nobody can act on. Surprises arrive quarterly.

Gate rule. Fail any one and the answer is single-platform, whatever the need score says. Fix the gap first; it is cheaper than running two estates badly.

2 The need score

Only if all three gates cleared. Score each signal 0–2 and add them.

Signal	0	1	2
Workload diversity	BI and light transforms	Some ML or streaming	Heavy ML/streaming *and* heavy self-service BI
Existing investment	Greenfield	One platform in production	Two in production with real expertise in both
Data gravity	All data can live in one place	Some external sources	Large datasets that genuinely cannot move

0–2 · One platform
Greenfield or one platform today: stay there. **Already running two? Read this as a consolidation signal**, not a status quo.

3–4 · One, open formats
Stay on Delta or Iceberg so the door stays open. Revisit in a year. The optionality is the point, not the second engine.

5–6 · Evaluate multi-engine
You have reasons to consider it and the capability to operate it. Validate the workload benefit and the operating cost before you commit.

Read the three signals separately before you add them up. A score carried mostly by existing investment is a sunk-cost argument: it is the case for finishing a consolidation, not for committing to two estates. Data gravity is the opposite, and the only signal that does not decay. A 5 built from diversity and gravity is a different recommendation from a 5 built from investment and diversity, and the total cannot tell you which one you are holding.

Read the arithmetic before you trust the band. Four is the ceiling if all your data can live in one place, so a shop at 2/2/0 — two platforms in production, heavy ML and heavy BI — lands on "one platform". That is deliberate: **data gravity is the signal that most often decides this**, and if it is genuinely zero, consolidation is usually right even when the workloads are diverse. But 2/2/0 is a judgment call, not a verdict. Take it to your architects and let the gates outweigh the number.

3 Where sovereignty fits

A sovereign-cloud requirement makes multi-engine **harder**, not more attractive. Treat it as a reason to raise the bar on the gates, never as a signal favoring more engines:

- **Feature parity lags.** Azure Government regions do not currently support Databricks SQL or Unity Catalog; Databricks' own November 2025 announcement targets the fuller platform for 2026.
- **Regional co-location is a real constraint.** Fabric for GCC High lists US Gov Virginia and US Gov Texas; Databricks in Gov is Arizona and Virginia. US Gov Virginia is the only region where the interim shortcut pattern can be co-located.

- **The AI layer is not there yet.** Copilot and data agents are absent from the GCC High preview feature table, so an architecture whose value depends on them does not extend into that environment.
- **Scope.** The GCC High preview covers GCC High, not DoD/IL5. Do not read one as the other.

4 The seam to pressure-test before go-live

Three different things happen at the mirror seam, and the one people expect is the one that does not. Table, schema and catalog GRANTs do not carry over, and every Fabric query runs as the credential that configured the mirroring connection — not the end user — so that principal's reach is the real blast radius. Tables carrying row filters or column masks **fail closed**: Databricks supports neither Unity REST API nor path-based access to them, so they arrive unreadable rather than unfiltered. The one genuine bypass is a **direct ADLS read** — the trusted-workspace-access pattern for firewalled storage — where Unity Catalog policy is explicitly not enforced at the storage layer.

> **The failure mode worth hunting:** a table that fails closed looks broken, not protected, so someone drops the row filter or lands an unfiltered copy to unblock delivery. No error, no change record. Inventory filtered tables *before* you mirror and agree what happens to each one.

Re-implement the equivalent controls in OneLake security. Confirm three things for your own tenant: the status for Mirrored Azure Databricks Catalog items specifically; that GCC High preview still lacks Spark support for OneLake security; and whether cross-engine ABAC (Beta, 11 Sep 2026) is enabled, since it enforces filters and masks for external engines and changes the second case above.

Sources. All read 18 September 2026; Microsoft revision dates given where published.
Mirrored databases from Azure Databricks — security (rev. 1 May 2026) — learn.microsoft.com/en-us/fabric/mirroring/azure-databricks-security
Row filters and column masks (rev. 11 Sep 2026) — learn.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/filters-and-masks
Cross-engine ABAC, Beta (rev. 11 Sep 2026) — learn.microsoft.com/en-us/azure/databricks/external-access/cross-engine-abac
OneLake table, column and row-level security (rev. 19 Aug 2026) — learn.microsoft.com/en-us/fabric/onelake/security/table-column-row-security
Mirrored catalog from Azure Databricks — learn.microsoft.com/en-us/fabric/mirroring/azure-databricks

OneLake security: RLS and CLS are documented as working controls with defined per-engine enforcement, and carried no preview label as of 18 September 2026. For a GA date on a compliance artifact, ask your Microsoft account team.

Sovereign-cloud feature tables move quarterly. Verify anything time-sensitive against the source before planning around it — that is why this document is dated.