Microsoft Finally Embraced dbt.

# Here's What That Actually Means for Your Data Team

# Reader Summary

At Ignite 2025, Microsoft introduced dbt jobs as a first-class item in Fabric Data Factory, with the dbt Fusion engine confirmed for 2026. This is not just a new feature. It is a strategic signal: Microsoft is giving dbt native hosting, integrated governance, and a long-term home inside Fabric. For data teams that have been running dbt externally, waiting on the sidelines, or wondering which transformation approach to commit to, the picture just got a lot clearer. This post covers what the integration actually includes today, how it fits alongside dbt Cloud, and what it means for how you should be thinking about your transformation strategy on Fabric.

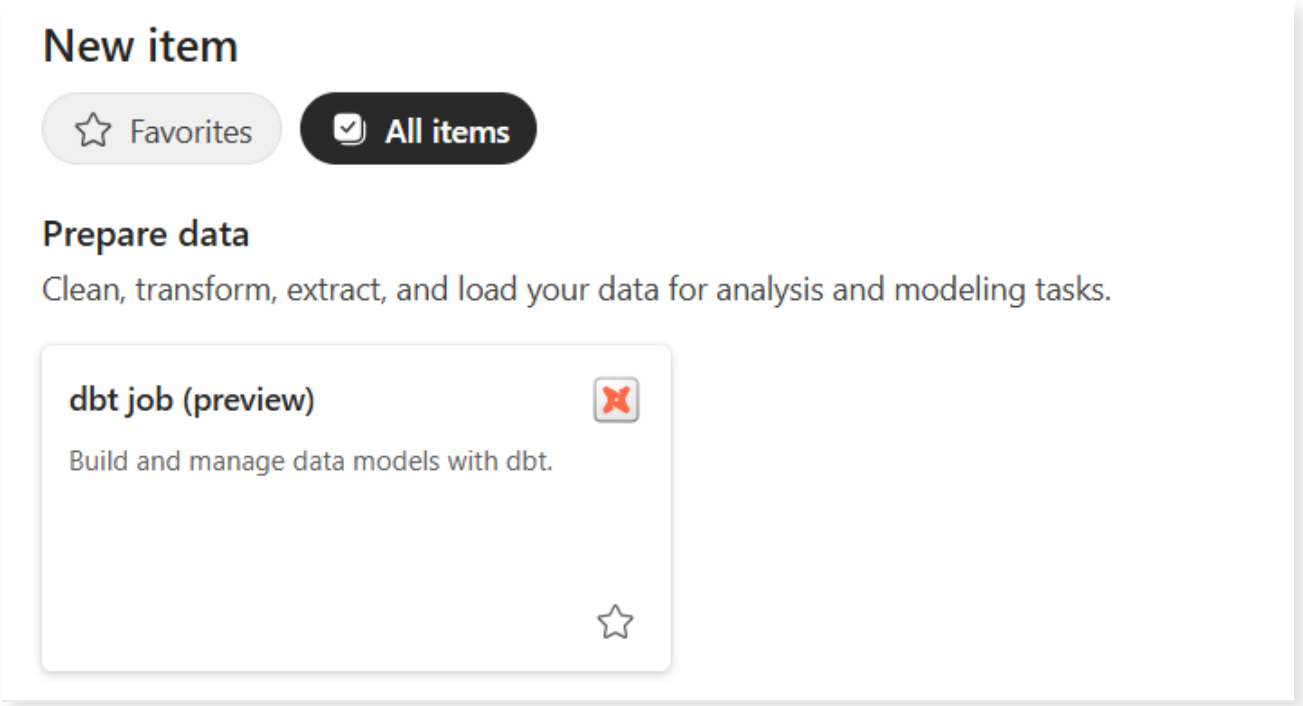



*Figure: dbt jobs now appear as a native item type in Fabric.*

# Why This Is a Bigger Deal Than It Looks

For years, Microsoft had a clear position on data transformation in its ecosystem. Power Query handled ingestion and reshaping. Dataflows provided a managed, low-code transformation layer. DAX and the semantic model governed the analytics logic. These were capable tools, and for many organizations they still are. But as data estates grew more complex and the demands on data teams evolved, the industry began converging on a set of expectations that went beyond what any single vendor's proprietary tooling could easily deliver: full version control over transformation logic, automated testing as a first-class workflow, lineage that traces from source to consumption, and documentation that lives alongside the code rather than in a separate system that nobody maintains.

dbt became the standard that crystallized those expectations. It did not replace SQL. It made SQL the interface for a set of engineering practices, testing, documentation, modularity, and dependency management, that the data community had been building toward for years. dbt has established itself as the industry standard for data transformation, recognized by Gartner as a leading data integration tool and widely covered as the dominant player in the space. The analytics engineering discipline it helped define has become a recognized function in most enterprise data organizations.

Microsoft's decision to bring dbt inside Fabric as a native, first-class capability is a recognition of that shift. It is not a replacement for Power Query or dataflows, which continue to serve their use cases well. It is an acknowledgment that the transformation layer in a modern data platform needs to support the engineering rigor that enterprises now require, and that dbt is the tool the community has chosen to deliver it. First-class support means ongoing compatibility, native hosting, integrated governance, and a committed roadmap. That is a fundamentally different proposition from running dbt externally and bridging the gaps yourself.

> "Microsoft is not replacing Power Query or dataflows. **It is acknowledging that the transformation layer needs engineering rigor."**

The timing adds further weight. The dbt Labs and Fivetran merger, announced in October 2025, positions dbt at the center of what they call "open data infrastructure." Microsoft's move to embed dbt natively in Fabric just weeks later underscores that dbt is no longer a niche tool for the Snowflake and BigQuery crowd. It is becoming a standard component of enterprise data platforms regardless of which cloud or warehouse you run.

# Three Ways to Run dbt on Fabric Today

With Microsoft's announcement, there are now three distinct paths for running dbt against Fabric, each suited to different team profiles and maturity levels. Understanding the trade-offs between them is more valuable right now than any deep dive into a single option.

## 1 dbt Cloud

dbt Cloud has supported Microsoft Fabric since mid-2024 through co-developed adapters built in collaboration with Microsoft. It provides a fully managed development environment with a browser-based IDE, built-in CI/CD, job scheduling, documentation hosting, and the dbt Fusion engine for dramatically faster project compilation. For teams that want the most complete dbt experience today with the least infrastructure to manage, dbt Cloud is the most mature option. Authentication works through service principals, and the platform handles environment management, version control integration, and artifact serving without any custom pipeline work. If you are already a dbt Cloud customer or evaluating it, Fabric is a fully supported deployment target right now.

## 2 dbt Core running externally

This is where many Microsoft-native dbt teams have been operating: a dbt Core project in a Git repo, running against Fabric Lakehouse or Warehouse SQL endpoints via the dbt-fabric adapter, with CI/CD through Azure DevOps or GitHub Actions and orchestration through an external scheduler. It works, and for mature teams with established DevOps practices it can work well. But the operational overhead is real. You are managing Python environments, dbt version pinning, adapter compatibility, service principal authentication, and log aggregation across multiple systems. Teams often spend as much time maintaining the dbt infrastructure as they do writing models.

## 3 dbt jobs in Fabric Data Factory

The new option. Fabric-native dbt jobs let you author, test, and run dbt projects directly inside Data Factory. The execution environment is managed. Authentication uses Fabric's identity model. Scheduling and monitoring live alongside your other Data Factory pipelines. You can connect an existing Git repository, so your development workflow does not have to change. The current preview runs on dbt Core, with the dbt Fusion engine confirmed for 2026. For teams whose biggest pain point is the infrastructure overhead of running dbt externally, this is the option that eliminates it. Fabric's enterprise security controls are applied automatically to every transformation run, which is a governance advantage that is genuinely difficult to replicate in a self-managed setup.

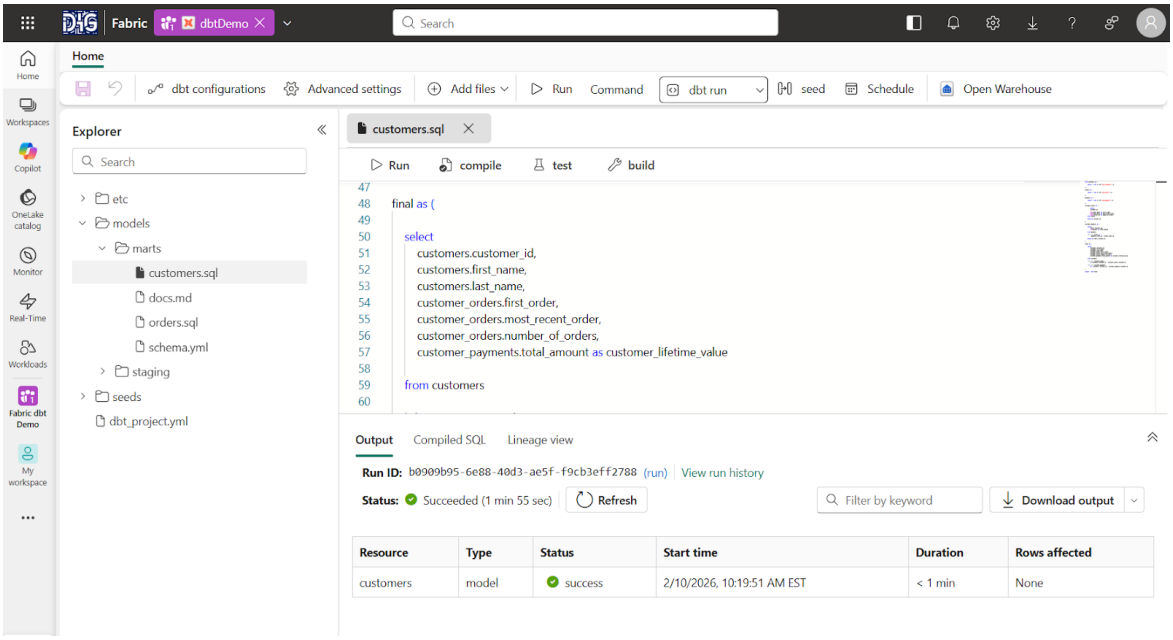



*Figure: The dbt authoring and execution experience inside Fabric.*

## Comparison of dbt Cloud, dbt Core external, and Fabric-native dbt jobs across execution, infrastructure, CI/CD, Fusion, targets, and cost

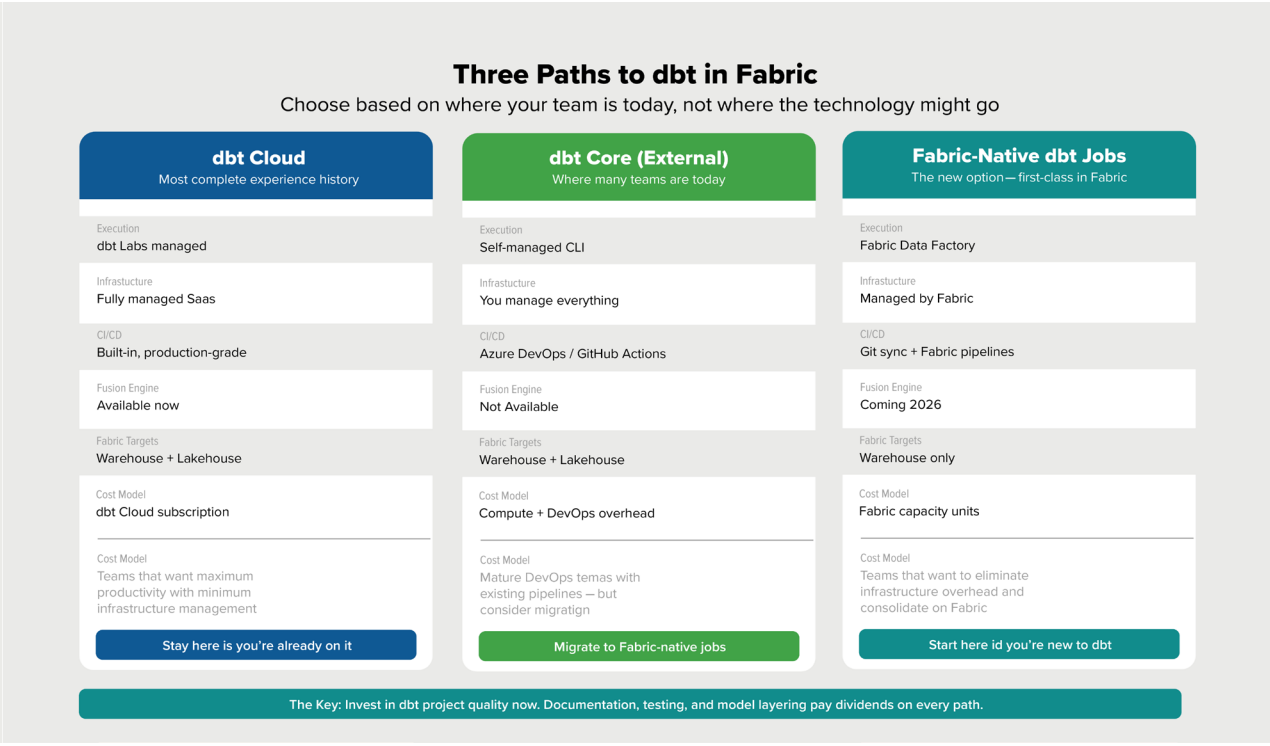



*Figure 1: Three paths to dbt on Fabric — choose based on your team's current maturity and pain points.*

# Which Path Makes Sense for Your Team

The right choice depends less on which option is technically best and more on where your team is today.

### If you are already on dbt Cloud: **stay there.**

You have the most mature dbt development experience available, with Fabric as a fully supported target. The Fusion engine is already available to you, the CI/CD and documentation workflows are production-grade, and you are not carrying any infrastructure management burden. There is no near-term reason to move to Fabric-native dbt jobs, though it is worth watching the Fabric integration evolve. The potential long-term consideration is cost consolidation: Fabric-native dbt jobs consume Fabric capacity units rather than requiring a separate dbt Cloud subscription. For organizations with significant Fabric capacity commitments, that could be a meaningful economic factor once the Fabric-native experience matures and Fusion arrives.

### If you are running dbt Core externally: **the Fabric-native dbt job is likely your next move.**

The infrastructure management overhead of external dbt, the Python environments, the custom CI/CD pipelines, the credential management, the scattered logging, is exactly what the Fabric integration eliminates. We have seen teams spend more time maintaining dbt infrastructure than writing transformation logic. That ratio inverts when the execution environment is managed for you. Start by spinning up a Fabric-native dbt job connected to your existing repo and running it in parallel. Compare results, validate your testing framework, and identify any models that behave differently. Migrate incrementally rather than all at once.

### If you are new to dbt and starting on Fabric: **start with the Fabric-native dbt job.**

You avoid building external infrastructure that will become unnecessary as the integration matures. Develop locally with the dbt CLI and VS Code for complex model work, push to your Git repo, and let Fabric handle execution. The reduced operational overhead is worth the trade-off of a web authoring experience that is still maturing. If your organization has the budget and wants the most productive development experience from day one, dbt Cloud with Fabric as the target is the premium path.

## What to Know Before You Commit

Regardless of which path you choose, there are practical realities of running dbt on Fabric that most of the official documentation does not cover. These come from what we are seeing in actual enterprise deployments.

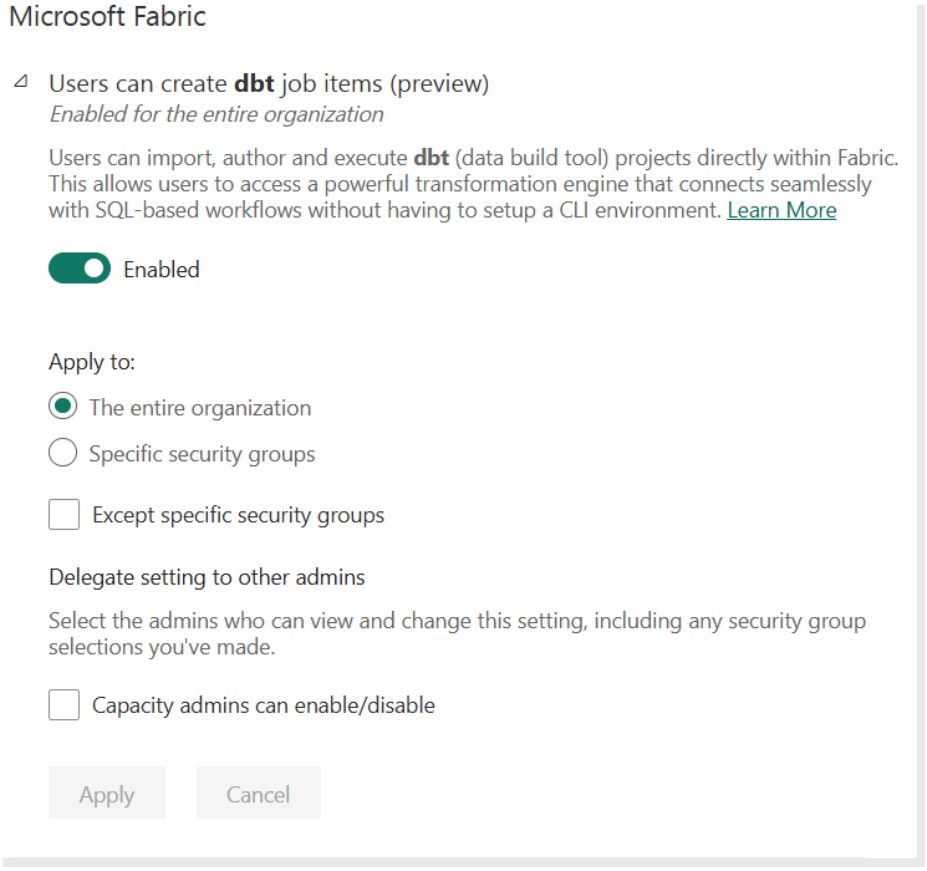



*Figure: dbt job enablement is governed through the Fabric tenant admin portal.*

## Fabric-native dbt jobs target Warehouses only.

As of the current preview, dbt jobs in Fabric Data Factory only support Fabric Warehouses as a target, not Lakehouse SQL endpoints. If your existing dbt project runs against a Lakehouse, you will need to retarget to a Warehouse before migrating to Fabric-native execution. This is worth knowing before you plan your transition, particularly if your current architecture relies heavily on Lakehouse endpoints. dbt Cloud and externally hosted dbt Core still support both target types through the adapter, so this constraint is specific to the Fabric-native integration.

## Be deliberate about incremental models.

The Fabric-native dbt job preview has limited support for incremental models with merge strategies, so plan accordingly if your project relies on them. But the broader point is worth making: we generally advise caution with merge-based incremental patterns regardless of platform support. They introduce operational state into the warehouse, which means your transformation output depends on what was already there rather than being fully deterministic from source data. That moves you away from the idempotent, reproducible ELT patterns that make dbt projects reliable and easy to reason about. In most cases, append-only patterns achieve the same performance benefits of incremental materialization without coupling your results to point-in-time warehouse state. You still get efficient, incremental processing, but the output remains deterministic and recoverable. If you are designing new models on Fabric, start with append-only incremental patterns and reserve merge strategies for the cases where they are genuinely unavoidable.

## Rethink your data CI/CD patterns.

When dbt ran externally, your CI/CD pipeline owned the entire lifecycle. With Fabric-native jobs, the execution happens inside Fabric, which means your pipeline's role changes. You are now pushing code to a Git repo that Fabric syncs, then triggering a Fabric job. Slim CI patterns, where you run only modified models in a PR environment, require rethinking. We have found that maintaining a lightweight external CI check for linting and compilation alongside the Fabric-native execution works well during the transition.

## Map your environments early.

dbt's concept of targets and environments maps imperfectly onto Fabric's workspace model. In a mature external setup, you might have dev, staging, and prod targets pointing at different databases. In Fabric, the equivalent is separate workspaces with deployment pipelines between them. Establishing a clear mapping between dbt environments and Fabric workspaces early avoids painful rework later.

# The semantic model bridge requires intentional work — **but the payoff is real.**

One of the most compelling aspects of running dbt inside Fabric is the proximity to Power BI semantic models. Your dbt models produce the tables that semantic models consume, and having both governed within the same platform should create a tight feedback loop. In practice, that bridge does not yet exist automatically. Changes to dbt models do not propagate to semantic model metadata, and there is no native mechanism to generate or update a semantic model definition from dbt's schema YAML files.

That said, the picture is not all gap. When you build a semantic model from a well-governed dbt gold layer, the lineage is not lost — it is enhanced, because the upstream documentation, testing, and column definitions your dbt project produces travel with the data into the semantic layer. And source control of the semantic model definition through the Fabric workspace Git integration means the semantic model itself can be versioned and governed alongside your dbt project.

The missing piece is the automated sync between the two, and that is where the Fusion engine's richer metadata capabilities could close the gap. In the meantime, factor explicit bridging work into your architecture planning rather than assuming it will just work.

# dbt Fusion in Fabric and the Bigger Picture

The current Fabric-native integration runs on dbt Core. The dbt Fusion engine, built on Rust with parse times up to 30x faster, is confirmed for Fabric in 2026. For large projects with hundreds of models, that is a material performance improvement. But Fusion is not just a speed upgrade. It includes deeper SQL comprehension, enhanced lineage capabilities, and richer metadata output, features that dbt Labs frames as making data "AI-ready": not just transformed, but enriched with the context, documentation, and quality signals that downstream consumers need to trust it.

This is where the connection to the broader Fabric ecosystem becomes interesting. A dbt project running on Fusion does not just produce clean tables. It produces tables with column-level descriptions, test coverage results, freshness checks, lineage graphs, and semantic type information. That rich metadata output is precisely the kind of structured context that Fabric IQ's ontology layer consumes. In practical terms, every well-documented dbt model becomes a building block for the ontology graph: the descriptions populate entity definitions, the tests validate business rules, and the lineage maps feed relationship types. The dbt project's technical artifacts directly supply the knowledge layer that agents and ontology-driven systems depend on. We explore that convergence in depth in our companion post on AI, ontologies, and Fabric.

With dbt and Fivetran combining under one roof, the ingestion-to-transformation path is tightening. Fivetran already has deep Azure and Fabric connectivity, and the combined entity will push toward increasingly seamless pipelines from source to dbt-transformed output within Fabric. For Microsoft-native shops, this means the dbt ecosystem is becoming more, not less, aligned with the Fabric investment.

**What this means** for planning your Fabric transformation layer.

These recommendations are scoped specifically to dbt projects targeting Fabric. Many organizations run dbt across multiple warehouses — Snowflake, Databricks, BigQuery, SQL Server — and the hosting decision for those workloads involves factors well beyond what Fabric offers. But for the Fabric-targeted portion of your dbt estate: if you are on dbt Cloud today with Fabric as a target, you are already on the best available experience and should stay there. If you are running dbt Core externally against Fabric, migrate to Fabric-native jobs to eliminate infrastructure overhead and position yourself for Fusion when it arrives. Either way, the most valuable thing you can do for your Fabric dbt project right now is invest in its quality. Write thorough model documentation. Add column-level descriptions. Increase test coverage. Use consistent naming conventions and model layering. That work pays dividends immediately in development velocity and team onboarding, and it positions your Fabric project to take full advantage of everything that is coming, from Fusion to ontology integration to agent-consumable data.

# Tech Takeaways

1

**dbt is now a first-class citizen in Microsoft Fabric.** Native hosting, integrated governance, and a committed roadmap signal a strategic embrace, not just a feature addition.

2

**Three paths exist today: dbt Cloud, dbt Core externally, and Fabric-native dbt jobs.** dbt Cloud offers the most complete experience now. Fabric-native jobs eliminate infrastructure overhead. Choose based on your team's current maturity and pain points.

3

**The dbt Fusion engine is coming to Fabric in 2026.** 30x faster parse times and richer metadata. Teams with clean, well-tested projects will have the smoothest transition.

4

**The Fivetran-dbt merger strengthens the Fabric ecosystem play.** dbt is becoming a standard component of enterprise data platforms regardless of cloud provider.

5

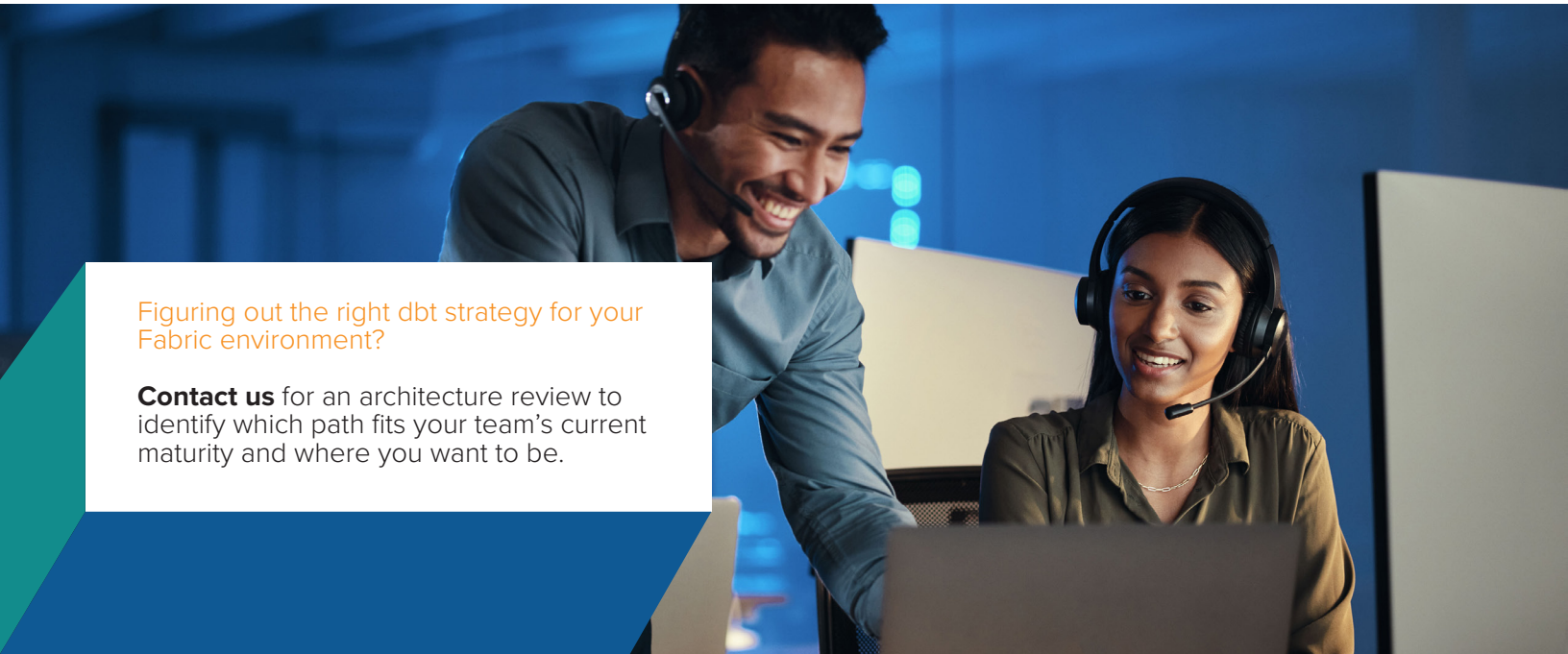
**The semantic model bridge is a gap to plan around.** dbt metadata does not yet flow automatically to Power BI semantic models. Factor explicit bridging work into your architecture.

6

**Fabric capacity consolidation is the long-term economic case.** Fabric-native dbt jobs consume capacity units rather than requiring a separate subscription. For organizations with significant Fabric commitments, that matters.

7

**Invest in dbt project quality now regardless of your path.** Documentation, testing, and consistent model layering are the foundation for Fusion adoption, ontology integration, and agent-consumable data.



Figuring out the right dbt strategy for your Fabric environment?

**Contact us** for an architecture review to identify which path fits your team's current maturity and where you want to be.