Dagster, SDF, & the Evolution of the Data Platform

August 22, 2024.

Introducing...

Lukas Schulte
Co-Founder & CEO
at SDF Labs

Pedram Navid
Head of Data Engineering & Devrel
at Dagster Labs

Agenda

1. Orchestrators and Transformation Layers
2. Unsafe SQL and Online Data Development
3. What is SDF and Dagster Refresher
4. Combining SDF and Dagster
5. Demo
6. Q&A

Follow along at the Dagster Deep Dive Slack

QR Code

Orchestrators and Transformation Layers

Traditional orchestrators have no visibility into what's happening in data transformation tools. We often author and orchestrate our data pipelines manually, stitching together complex python and SQL scripts, oftentimes calling out to a transformation tool like dbt to do the heavy lifting.

Fragmentation and Misalignment

But this approach leads to a few problems:

- ❌ **Fragmented Pipelines:** The handoff from orchestration to transformation is often obscure, leading to fragmented pipelines.
- ❌ **Inconsistent Data Quality:** The lack of coordination between orchestration and transformation can lead to inconsistent observability and data quality.
- ❌ **Increased Costs and Operational Complexity:** The misalignment between orchestration and transformation can lead to increased costs and operational complexity.

Unsafe SQL and Proprietary Compute Engines

SQL is the most widely used language for querying, managing, and transforming data. But, SQL is often written in an unsafe manner, leading to bugs, errors, and inconsistencies in data pipelines. Proprietary compute engines like Snowflake, BigQuery, and Redshift exacerbate this problem by being impossible to validate locally.

Online Data Development

Transformation tools are forced to execute SQL queries against the data warehouse for validation, leading to:

- ❌ **Slow Iteration:** Waiting for queries to execute is time consuming
- ❌ **Indirection and Operational Complexity:** Setting up isolated environments, complex table name resolution, and governance is challenging
- ❌ **Slow Iteration:** Waiting for queries to execute against a data warehouse is time consuming

SDF: The Toolkit for SQL Development

SDF is a multi-dialect SQL compiler, transformation framework, and analytical database engine. By natively compiling SQL dialects, like Snowflake, Redshift or Bigquery SDF helps you:

🎯 Develop SQL locally, without needing to execute queries against a data warehouse

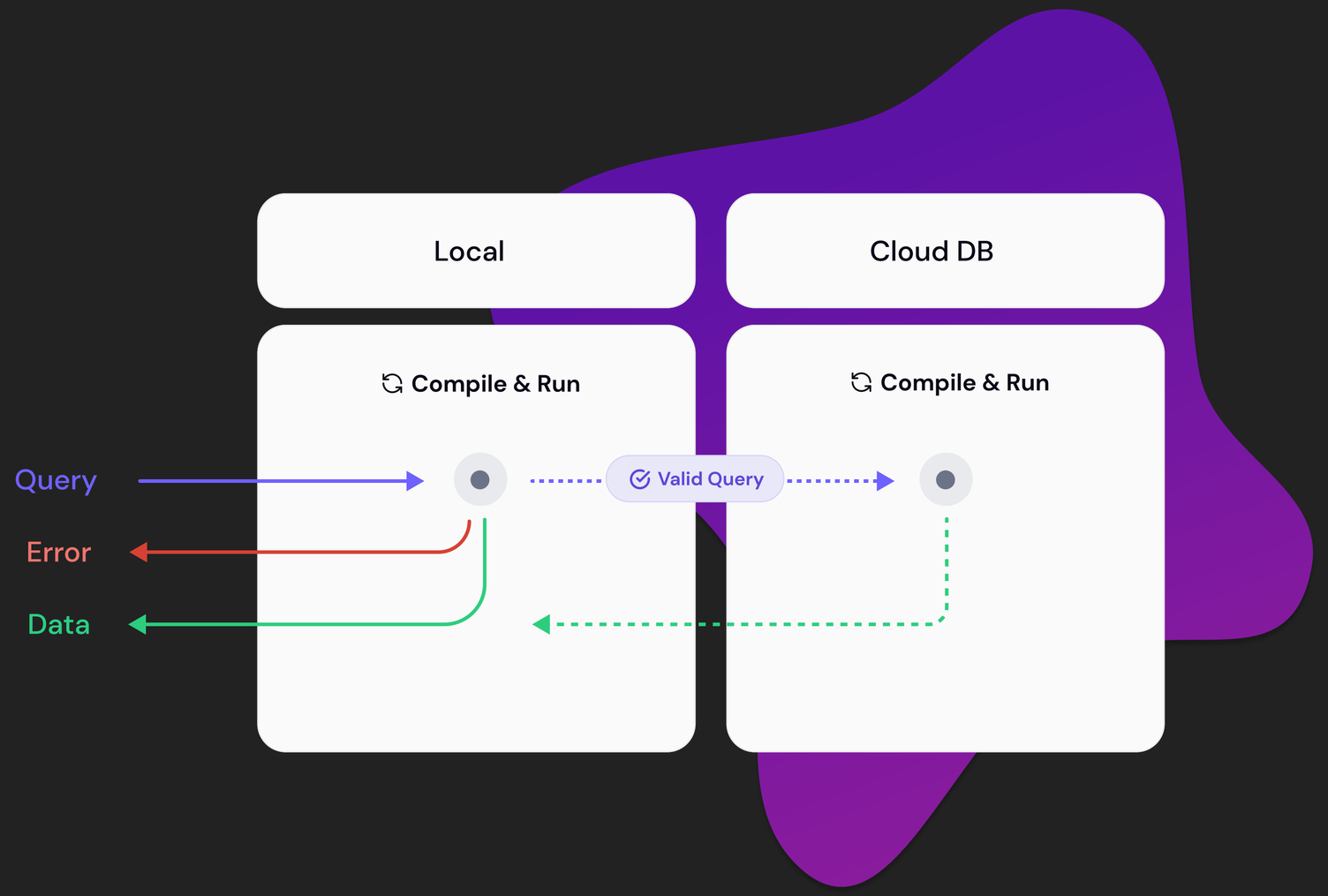
🚀 Gain precise column-level lineage and classifier-driven governance for full warehouse transparency

💥 Execute proprietary SQL dialects in-process, enabling fast feedback loops and robust SQL validation

🌟 And more...

SDF Compile Run Feedback Loop

SDF provides a fast feedback loop for SQL development



Dagster: The Asset-Driven Orchestrator

Dagster is a powerful orchestration platform that is built to interface with and understand the underlying transformation graph, enabling fine-grained control, deep observability, and the ability to model cross-technology lineage between the transformation layer and other systems.

Combining SDF and Dagster

A powerful transformation and authoring layer, SDF exposes a rich set of metadata and compile-time guarantees to Dagster, an extensive, asset-oriented orchestrator that natively understands the transformation graph.

Together, they provide a **seamless, reliable,** and **cost-effective data platform** that bridges the gap between transformation and orchestration.

Enhanced Data Quality and Reliable Deployments

SDF's compiler ensures your workspace is safe to execute locally or against your cloud data warehouse while Dagster's observability and orchestration tooling guarantees safe execution, in perpetuity.

Unprecedented Observability and Control

Dagster's support for metadata management, combined with SDF's static analysis, offers valuable insights into column-level lineage, dependency analysis, and performance metrics.

Local Development and Testing

Both SDF and Dagster offer robust local development environments, with SDF providing local execution, linting, and powerful checks, which seamlessly integrate into Dagster's comprehensive UI for monitoring and debugging.

Cost Savings & Caching

Dagster and SDF not only limit the number of queries executed against your data warehouse when deving, but also expose an intelligent caching layer that materializes models only if upstream dependencies have changed.

DEMO

Q&A

Questions? Comments? Feedback?