



SQLMesh is an open source data transformation framework that brings the best practices of DevOps to data teams. It enables data scientists, analysts, and engineers to efficiently run and deploy data transformations written in SQL or Python. It is created and maintained by Tobiko Data, Inc., a company founded by data leaders from Airbnb, Apple, and Netflix.

Why SQLMesh?

The experience of developing and deploying data pipelines is more uncertain and manual when compared to developing applications. This is partially due to the lack of tooling revolving around the testing and deployment of data pipelines. With DevOps, software engineers are able to seamlessly confirm logic with unit tests, validate systems with containerized environments, and transition to prod with confidence. SQLMesh aims to give data teams the same confidence as their peers.

Here are some challenges that data teams run into, especially when data sizes increase or the number of data users expands:

1. Data pipelines are fragmented and fragile
 - Data pipelines generally consist of Python or SQL scripts that implicitly depend upon each other through tables. Changes to upstream scripts that break downstream consumers are usually only detected at run time.
2. Data quality checks are not sufficient
 - The data community has settled on data quality checks as the "solution" for testing data pipelines. Although data quality checks are great for detecting large unexpected data changes, they are expensive to run, and they have trouble validating exact logic.
3. It's too hard and too costly to build staging environments for data
 - Validating changes to data pipelines before deploying to production is an uncertain and sometimes expensive process. Although branches can be

deployed to environments, when merged to production, the code is re-run. This is wasteful and generates uncertainty because the data is regenerated.

4. Silos transform data lakes to data swamps
 - The difficulty and cost of making changes to core pipelines can lead to duplicate pipelines with minor customizations. The inability to easily make and validate changes causes contributors to follow the "path of least resistance". The proliferation of similar tables leads to additional costs, inconsistencies, and maintenance burden.

What is SQLMesh?

SQLMesh consists of a CLI, a Python API, and a Web UI to make data pipeline development and deployment easy, efficient, and safe.

Core principles

SQLMesh was built on three core principles:

1. Correctness is non-negotiable
 - a. Bad data is worse than no data. SQLMesh guarantees that your data will be consistent even in heavily collaborative environments.
2. Change with confidence
 - a. SQLMesh summarizes the impact of changes and provides automated guardrails empowering everyone to safely and quickly contribute.
3. Efficiency without complexity
 - a. SQLMesh automatically optimizes your workloads by reusing tables and minimizing computation saving you time and money.

Key features

1. Efficient dev/staging environments
 - a. SQLMesh builds a Virtual Data Environment using views, which allows you to seamlessly rollback or roll forward your changes. Any data computation you run for validation purposes is actually not wasted — with a cheap pointer swap, you re-use your “staging” data in production. This means you

- get unlimited copy-on-write environments that make data exploration and preview of changes fun and safe.
2. Automatic DAG generation by semantically parsing and understanding SQL or Python scripts
 - a. No need to manually tag dependencies — SQLMesh was built with the ability to understand your entire data warehouse's dependency graph.
 3. Informative change summaries
 - a. Before making changes, SQLMesh will determine what has changed and show the entire graph of affected jobs.
 4. CI-Runnable Unit and Integration tests
 - a. Can be easily defined in YAML and run in CI. SQLMesh can optionally transpile your queries to DuckDB so that your tests can be self-contained.
 5. Smart change categorization
 - a. Column-level lineage automatically determines whether changes are "breaking" or "non-breaking", allowing you to correctly categorize changes and to skip expensive backfills.
 6. Easy incremental loads
 - a. Loading tables incrementally is as easy as a full refresh. SQLMesh transparently handles the complexity of tracking which intervals need loading, so all you have to do is specify a date filter.
 7. Integrated with Airflow
 - a. You can schedule jobs with our built-in scheduler or use your existing Airflow cluster. SQLMesh can dynamically generate and push Airflow DAGs. We aim to support other schedulers like Dagster and Prefect in the future.
 8. Notebook / CLI
 - a. Interact with SQLMesh with whatever tool you're comfortable with.
 9. Web based IDE
 - a. Edit, run, and visualize queries in your browser.
 10. Github CI/CD bot
 - a. A bot to tie your code directly to your data.
 11. Table/Column level lineage visualizations
 - a. Quickly understand the full lineage and sequence of transformation of any column.