ProjectX

SQL & ANALYTICS · DATASHEET

SQL warehouses, dashboards and a query history you can trust.

Interactive SQL on Trino over your governed tables and federated sources, from warehouses that scale to zero and bill by the second. Preview what a query will scan before it runs, chart the answer on a dashboard, and find every statement again in a history with its plan.

Why teams choose it

01
Know the cost before you run it
Every statement can be previewed first: the SQL that would actually run, after policy, and the bytes it would scan.

02
Warehouses that stop when you do
SQL warehouses by size, autoscaling with scale-to-zero, billed per second by the SKU you chose.

03
Every tile runs as the viewer
Dashboards and query results are filtered by the same row and column rules as the person looking at them, so one dashboard serves everyone safely.

04
Questions in plain language, SQL you can check
Ask in words and get SQL grounded in your semantic layer — always shown, and permission-checked before it runs.

In numbers

1 s
billing grain for every warehouse, by size

0
warehouses running when idle, with scale-to-zero

5
warehouse sizes, extra small to extra large

What it does

Interactive SQL on Trino, with federation

Query governed tables and external sources together, with an editor that browses the schema you may read and keeps your history.

Trino runs the statement over your Iceberg tables and, through federation, over the sources you connect. The editor shows only what your grants reach, and every statement lands in a searchable history with its plan.

Warehouses, queues and limits

Warehouses by size with autoscaling and scale-to-zero, queues with priority and concurrency limits, and kills on demand or by policy.

A warehouse grows with the queue and stops when idle. Queues decide who runs first and how many at once, and a query that runs too long or would cost too much is stopped by policy rather than by someone noticing the invoice.

Dashboards and alerts

Charts, filters, drill-down and scheduled refresh in a colour-blind-safe palette, plus alerts on query results by threshold or anomaly.

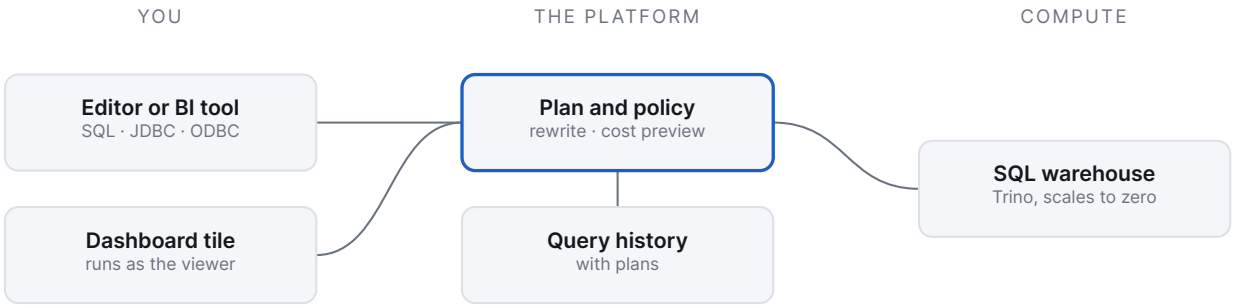
A dashboard is a set of tiles, each a query and a chart, that runs as whoever opens it. Refresh it on a schedule, drill from a bar into the rows behind it, and alert when a result crosses a line.

Your BI tools, connected

Connect over JDBC and ODBC; result caching and materialized views keep repeated questions fast.

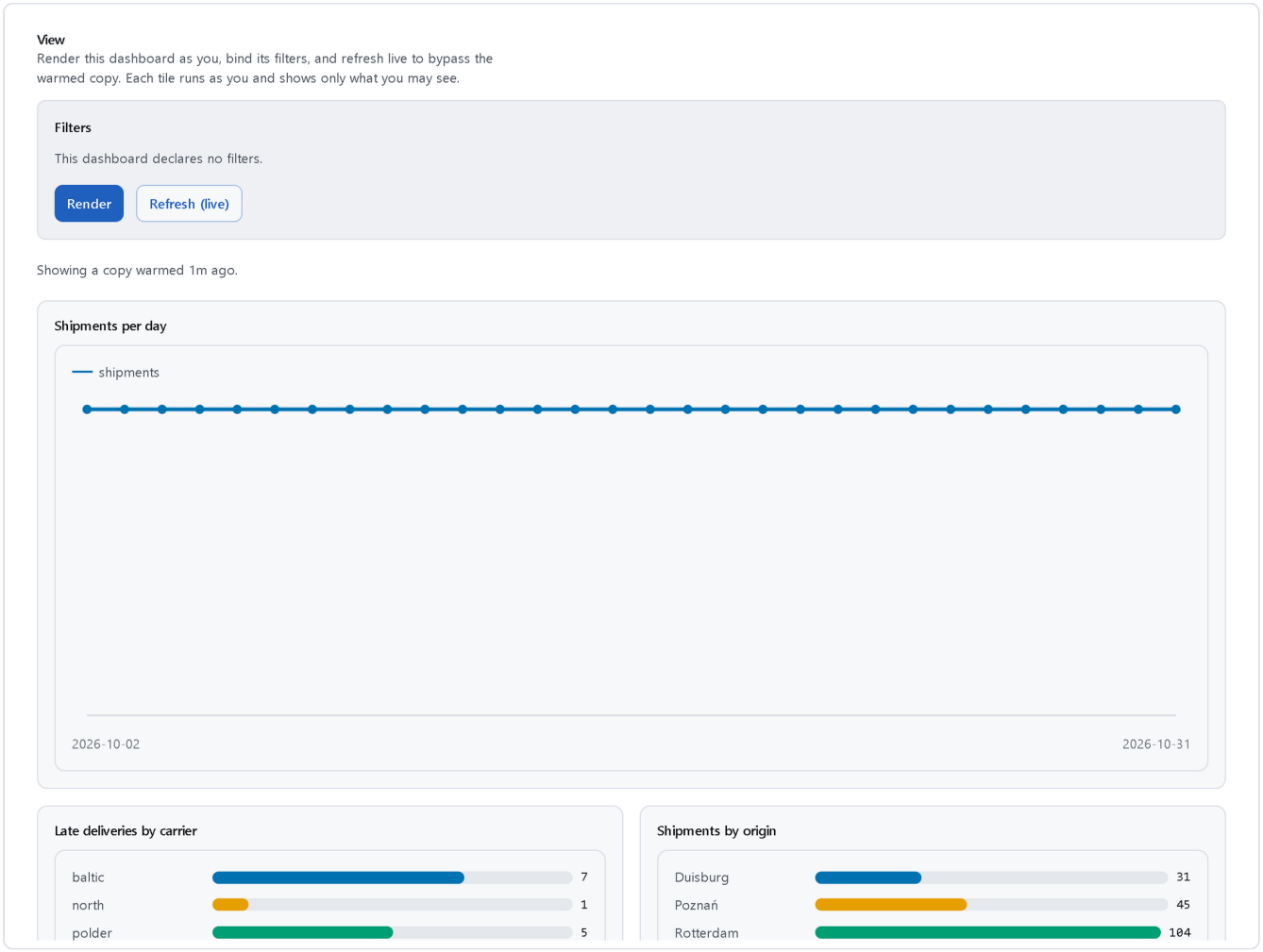
Existing BI tools connect over JDBC and ODBC and read through the same rules as the editor. Repeated queries are answered from the result cache, and heavy ones can be materialized.

How it fits together

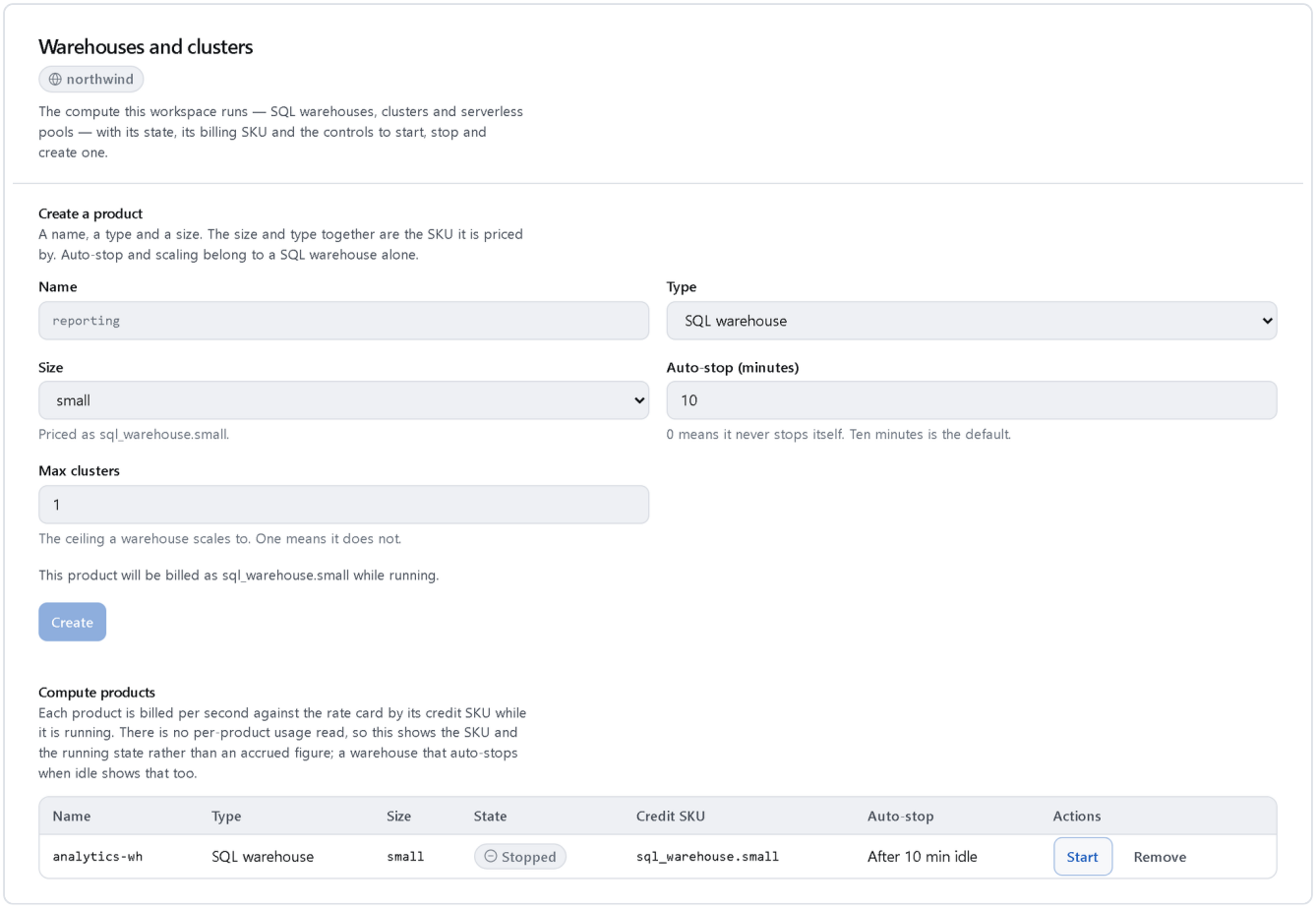


A statement from the editor, a dashboard tile or a BI tool is planned first: policy rewrites it to the rows and columns the asker may see and the cost is estimated. A warehouse then runs it on Trino, and the result and its plan are kept in the history.

In the console



A dashboard whose every tile runs as the person viewing it. Screens shown with demo data.



SQL warehouses by size, each billed per second by its SKU. Screens shown with demo data.

Key ideas

SQL warehouse

A named pool of SQL compute with a size and an auto-stop. Its type and size are the SKU it is billed by, per second, while it runs.

Cost preview

What a statement would do before it does it: the SQL after policy, and an upper bound on the bytes it would scan.

Federation

Querying an external source alongside your own tables in one statement, without copying it in first.

In depth

Policy in the plan
Row filters and column masks are applied to the query plan, so the editor, dashboards, BI tools and the API all see the same narrowed answer.

Spend controls
Queues set priority and concurrency, a query can be killed on demand or by policy on runtime or cost, and compute is held when the credit balance runs out.

Natural language, checked
Questions in plain language become SQL grounded in the semantic layer. The SQL is always shown, and permission-checked before it runs.

How it works

01 **Start a warehouse** · Console
Choose a size and an auto-stop; the size and type are the SKU it is billed by, per second.

02 **Preview, then run** · SQL
See the SQL policy will run and what it will scan, then run it and export the rows.

03 **Chart it** · Console
Save tiles to a dashboard, refresh it on a schedule and drill into the rows.

04 **Find it again** · API
Every statement is in a searchable, shareable history with its plan and profile.

Who it is for

Analysts
Answers from governed tables without waiting for an extract.

An editor with the schema you may read, a cost preview, and results you can export or chart.

Operations leads
A dashboard the whole team can open without anyone seeing more than they should.

Dashboards whose every tile runs as the viewer, with scheduled refresh and drill-down.

Finance and platform owners
SQL spend that does not run all weekend.

Warehouses that scale to zero, queues with limits, and queries killed by policy on runtime or cost.

Works with

SQL editor · Dashboards · JDBC and ODBC · API · Python SDK

Questions buyers ask

Which SQL engine runs our queries?
Trino, over your Iceberg tables and, through federation, over the external sources you connect.

Can we see what a query costs before running it?
Yes. A preview returns the SQL that would actually run after policy is applied, and an upper bound on the bytes it would scan.

Do our BI tools work?
Tools that connect over JDBC or ODBC can query through the same row and column rules as the editor.

Who sees what on a shared dashboard?
Each tile runs as the viewer, so two people opening the same dashboard each see only what they are allowed to.

How is it priced?
SQL warehouses are billed per second by size at published rates in Platform Credits, and stop when idle. The rates are on the pricing page.

Pricing

Current published rates, in your currency: https://adaydev.com/pricing/products/sql_warehouse