ProjectX

ML & GENAI · DATASHEET

Models you can trace, and AI that reads only what it may.

Experiments, a model registry and serving, with promotion gated on a model card and an approval and every version traced to the snapshot it trained on. For generative AI: corpora, embeddings and a per-tenant vector index, with retrieval filtered by the person asking.

Why teams choose it

01
Every model traces to its data
A version records the run that made it and the exact table snapshot that run read, so “what was this trained on?” has an answer.

02
Nothing reaches production undocumented
Promotion to production is refused until the version has a model card and a reviewer’s approval.

03
Retrieval sees what the asker sees
Retrieval is filtered by the requesting user at query time, so an assistant never surfaces a document its user could not open.

04
Inference where you decide
Route to self-hosted or external models by policy, or turn on sovereign mode and forbid external inference entirely.

In numbers

2
separate sign-offs, card then approval, before production

4
rollout strategies: canary, shadow, A/B and blue-green

1
vector index per tenant, never shared

What it does

Experiments, training and evaluation

Track parameters, metrics and artifacts; train on Spark and Ray with GPUs; search hyperparameters and evaluate by slice.

Every run records what it tried and how it scored, and pins the table snapshots it read. Training scales out on Spark or Ray, sweeps search the parameter space, and evaluation breaks results down by slice and checks fairness.

A registry with a gate

Versions, stages and approvals, with a model card required before production and lineage back to the training snapshot.

Documenting a version is the author's job and approving it the reviewer's, held apart on purpose. Production is refused until both are done, and the version always traces to the run and snapshot behind it.

Serving and monitoring

Autoscaling real-time endpoints, batch inference, canary, shadow, A/B and blue-green rollouts, and drift monitoring.

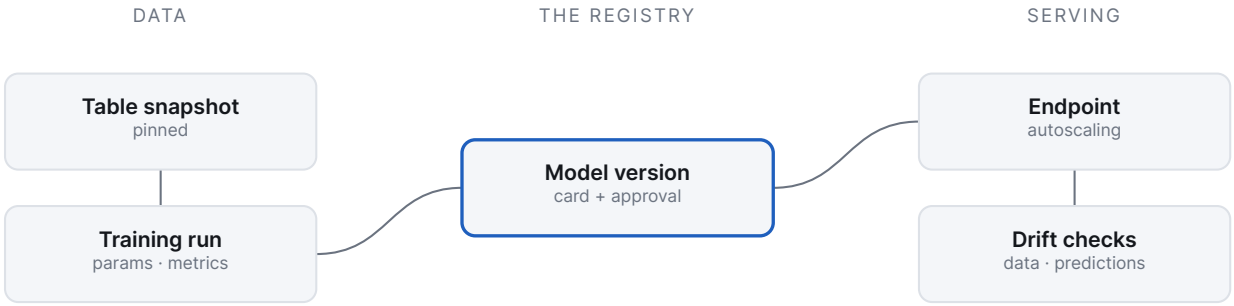
An endpoint scales with traffic and a new version can be rolled out to a slice first, shadowed, or swapped blue-green. Drift checks watch the data, the concept and the predictions once it is live.

Retrieval and generation, governed

Corpora with configurable chunking, embeddings, a per-tenant HNSW index, hybrid search and reranking, composed into RAG pipelines with guardrails and budgets.

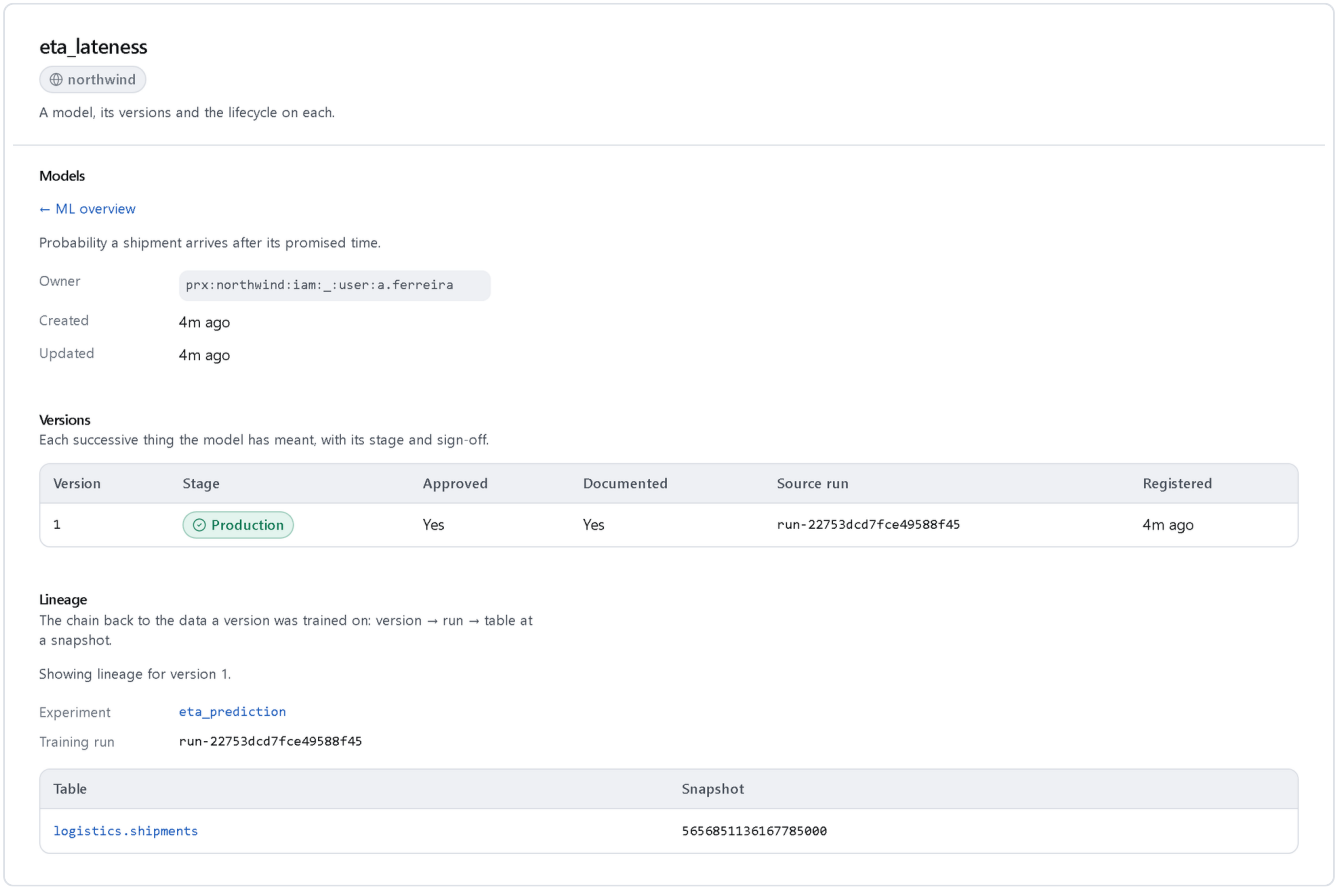
Documents are chunked and embedded into an index that belongs to your tenant alone. Retrieval fuses dense and keyword search, reranks, and filters by the asker's permissions; pipelines add guardrails, output schemas and token budgets.

How it fits together

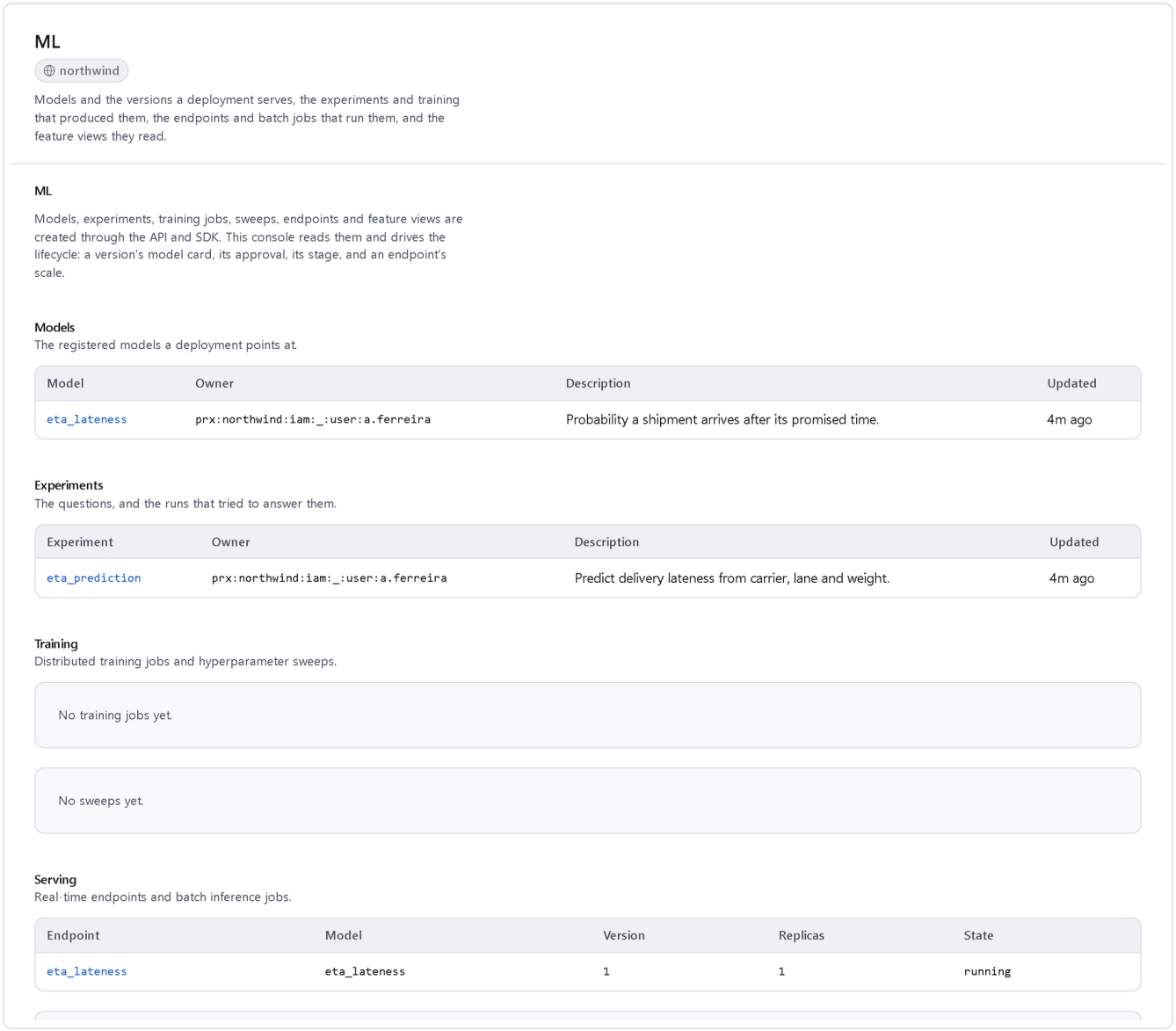


A run reads tables at pinned snapshots and records its results. The best run becomes a registered version, which needs a model card and an approval before it may be promoted, and only then goes behind a serving endpoint.

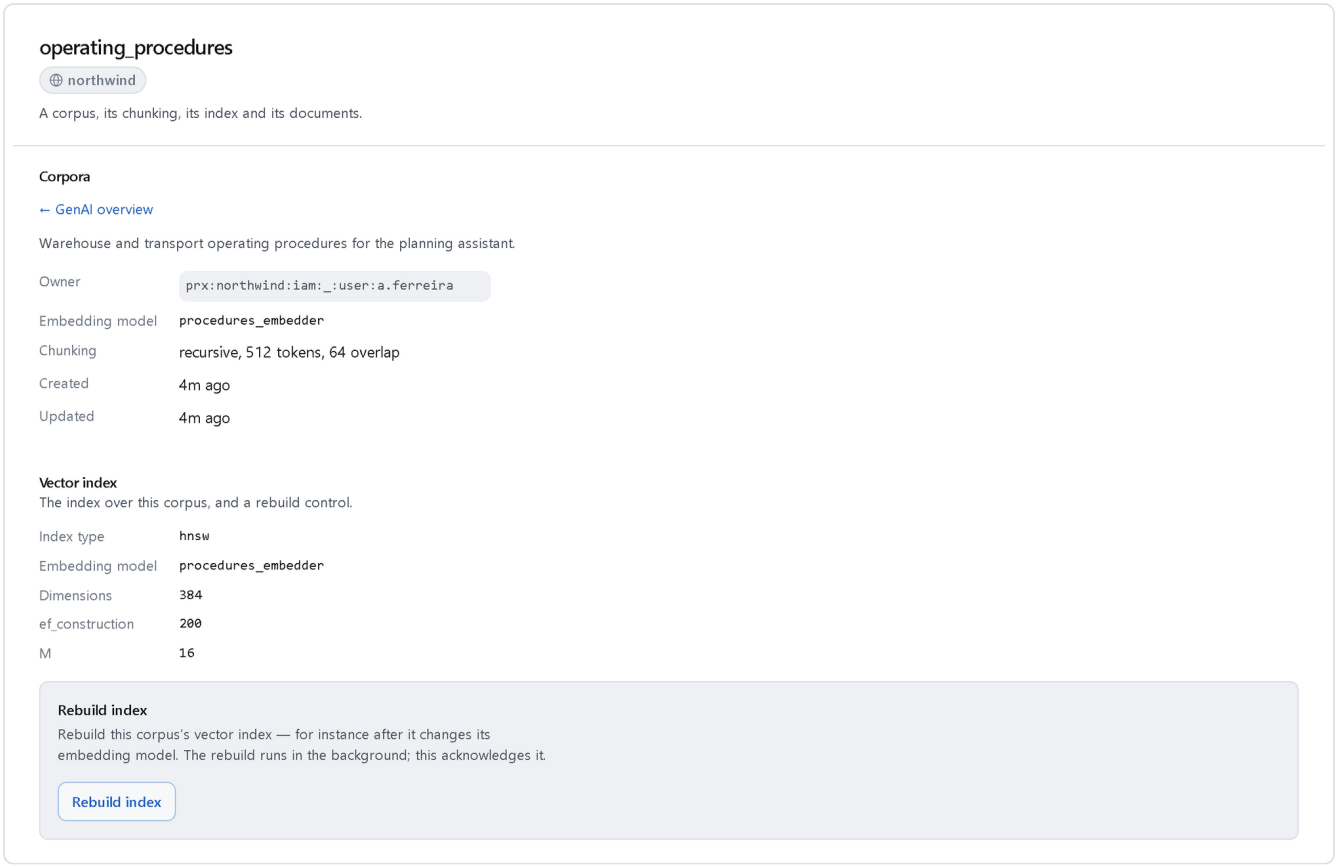
In the console



A production model traced to the snapshot it was trained on. Screens shown with demo data.



Models, experiments and the endpoints that serve them. Screens shown with demo data.



A corpus, its chunking and the vector index over it. Screens shown with demo data.

Key ideas

Model card

What a version is for, what it was trained on, where it falls down and who it could affect. Required before production.

Lineage to a snapshot

The chain version → run → table at a snapshot, so the training data is the exact rows the run read, not today's table.

Permission-aware retrieval

Retrieval that filters candidate documents by the requesting user's permissions at query time, before anything reaches a model.

In depth

Features without leakage
A feature store keeps features offline in Iceberg and online for serving, with point-in-time correct retrieval so training never reads the future.

Guardrails and budgets
Pipelines redact personal data, defend against prompt injection, filter toxicity, validate output against a schema and stop at a token budget per tenant, user or agent.

Evaluation you can repeat
An evaluation harness scores groundedness, relevance, latency and cost, and prompts are versioned so a change can be A/B tested.

How it works

01 **Track the runs** · Python SDK
Log each run's parameters, metrics and the table snapshots it read.

02 **Register and document** · API
Register the best run as a version and write its model card.

03 **Approve and promote** · Console
A reviewer approves it; only then can it move to production.

04 **Serve it** · API
Put it behind an autoscaling endpoint and roll later versions out gradually.

Who it is for

Data science teams
Experiments, versions and deployments without a separate platform to run.

Tracked runs, a registry with stages and approvals, and endpoints with canary and blue-green rollouts.

Risk and compliance
Proof of what a production model is, what it was trained on and who signed it off.

A model card and approval for every production version, and lineage to the training snapshot.

Teams building assistants
Answers grounded in company documents without leaking them to the wrong person.

Per-tenant vector indexes, permission-aware retrieval, guardrails and token budgets.

Works with

Python SDK · API · Console · Spark · Ray

Questions buyers ask

Can we prove what a model was trained on?
Yes. A version records the run that produced it, and the run pins the snapshot of every table it read, so the chain from production back to the data is on the record.

What stops an unreviewed model reaching production?
Promotion to production is refused until the version has a model card and an approval from a reviewer.

Which language models can we use?
The gateway routes to self-hosted models or to external providers by policy, with provider keys kept in the vault. Sovereign mode forbids external inference entirely.

Can an assistant leak documents?
Retrieval is filtered by the requesting user at query time, and guardrails redact personal data and defend against prompt injection.

How is it priced?
Training and serving compute is billed per second by cluster size, and model tokens by the token, at published rates in Platform Credits. The rates are on the pricing page.

Pricing

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