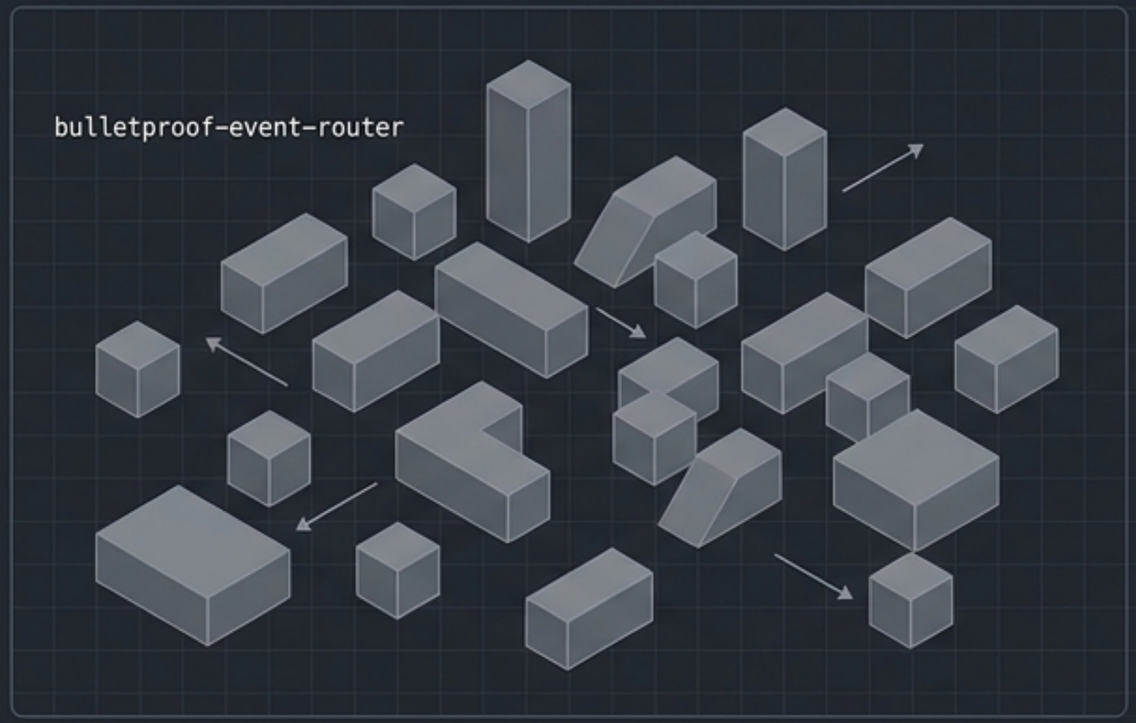
bulletproof-metrics-engine

Outcome Metrics and Predictive Scaling from the Event Stream

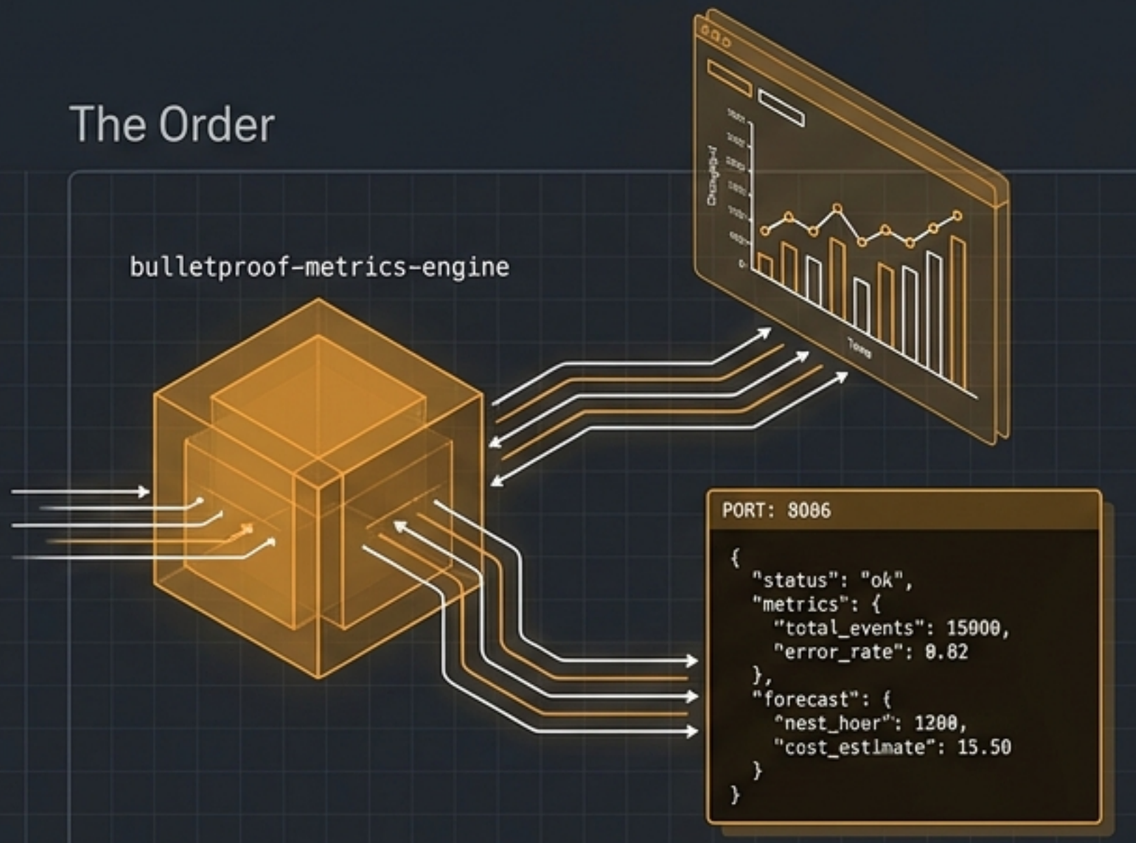
Box 1	Box 2	Box 3
A FastAPI service providing the intelligence layer for agent operations.	Transforms raw task lifecycle events into APIs and an HTML dashboard.	No machine learning required—powered by deterministic lifecycles and statistical forecasting.

From Raw Logs to Operational Wisdom

The Chaos



The Order

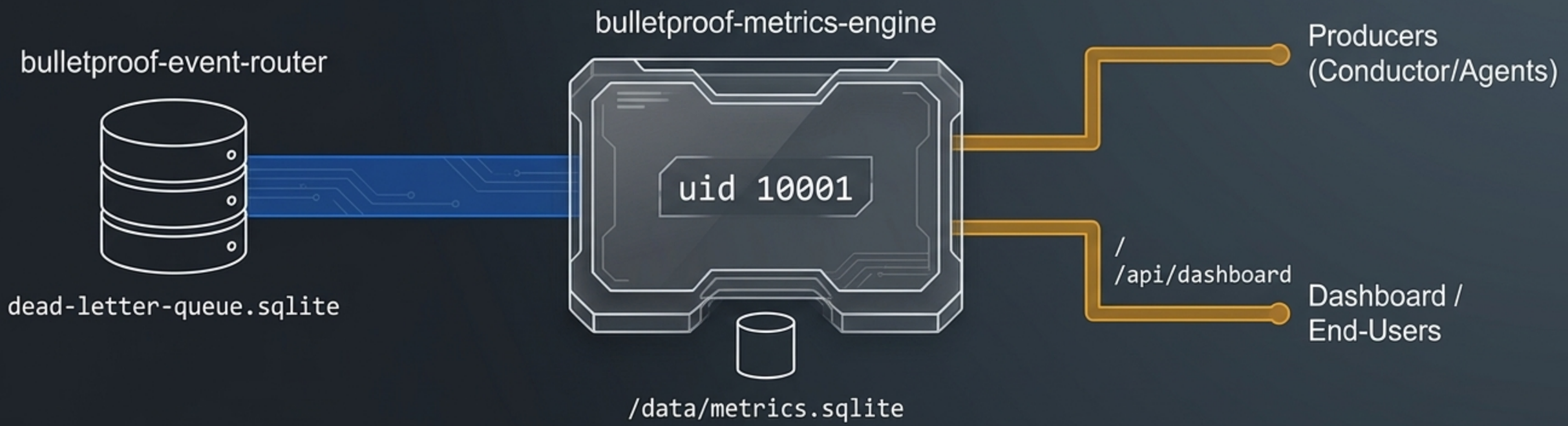


The engine ingests chaotic, read-only event streams.

Synthesizes hindsight: How well is work actually getting done?

Projects foresight: What are the demand and cost trajectories?

Architectural Position in the Stack



- Operates as a standalone Python 3.12 container.
- Requires no heavy database server—persistence relies on a single local SQLite file.
- Designed to sit alongside the event router or accept direct event pushes.

Dual Ingestion Pathways

The Pull Method (Automatic)

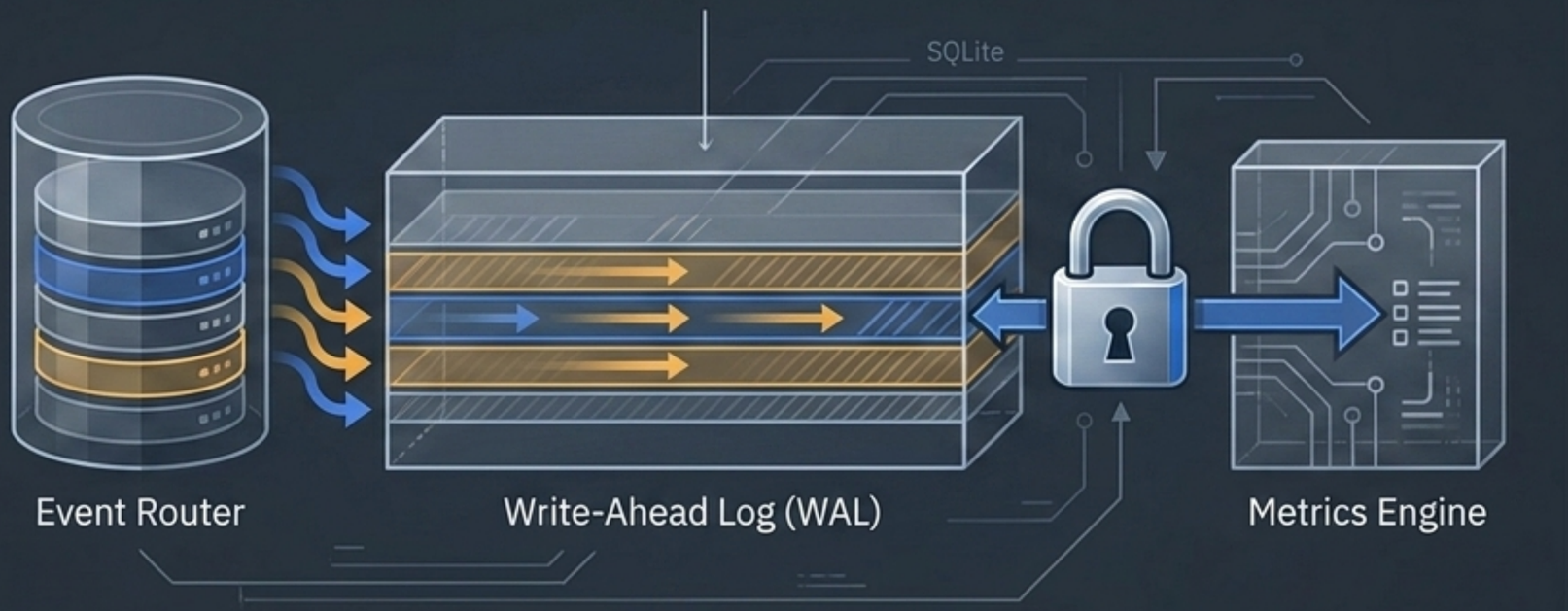
Source:	`EVENT_ROUTER_DB`
Mode:	Background loop reading every `AGGREGATE_INTERVAL_SEC` (default 60s).
Setup:	Read-only SQLite mount. Maps `category.type` to internal taxonomy.
State:	Idempotent; tracks last-seen `event_id` in `sync_state`.

The Push Method (Direct)

Source:	Conductor or remote Agents.
Mode:	Direct POST to `/api/events/ingest`
Setup:	JSON payload requiring only `event_id` and `event_type`.
State:	Idempotent; rejects duplicates by checking existing IDs.

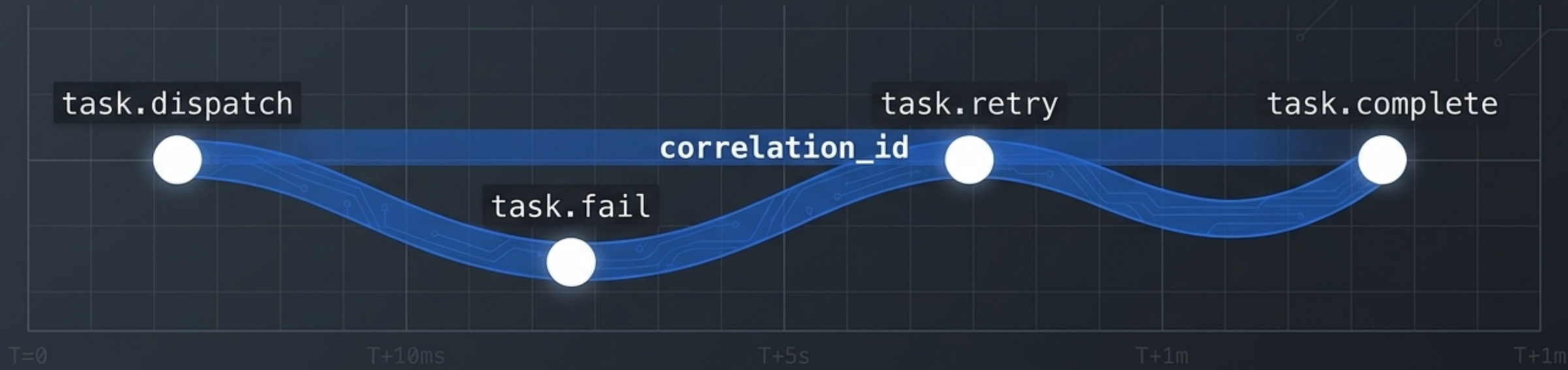
Safely Reading the Router's WAL

`file:...?mode=ro&immutable=1`



1. The event router runs in WAL (Write-Ahead Logging) mode.
2. A standard read-only (`mode=ro`) connection would attempt to create sidecar files on the read-only mount and fail.
3. The Solution: The metrics engine connects using `immutable=1`.
4. This asserts the file will not change under the reader, ensuring zero risk of corrupting the router's data.

The Lifecycle Thread



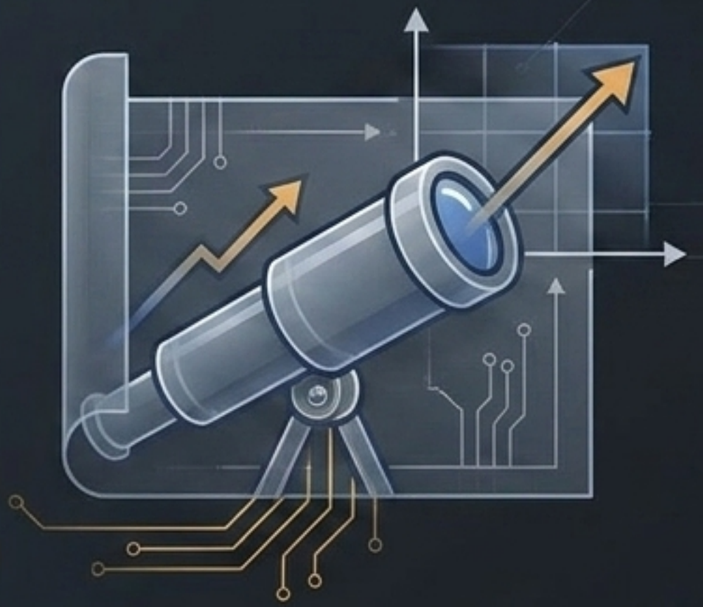
Raw events are meaningless in isolation.

The engine groups events into a **Lifecycle**: a set of events sharing a single `correlation_id`.

This thread makes metrics mathematically possible (e.g., calculating the time between dispatch and terminal state).

Join-compatible with the router's `/events` pipeline.

The Engine’s Dual Mandate



Hindsight (PRD 15): Outcome Metrics

Deterministic evaluation of what happened over a 7-day default window. Based strictly on completed lifecycles.

Foresight (PRD 16): Predictive Scaling

Statistical projections of what will happen. Based on empirical quantiles and configurable horizons.

Hindsight: Throughput and Speed

completion_rate	Formula: `count(terminal == task.complete) / count(any terminal state)`
	The percentage of tasks that successfully finished vs. failed or escalated. Terminal state is defined as the last event in the lifecycle.
ttr (Time-to-Resolution)	Formula: Seconds between `task.dispatch` and a terminal event. Returns `median_sec`, `p95_sec`, `mean_sec`, and sample counts.

Hindsight: Friction and Quality

`first_pass_rate` Of completed lifecycles, the fraction with strictly zero `task.retry` events.	**`rework_frequency`** The mean number of `task.retry` events per completed lifecycle.
`recovery_rate` Of lifecycles that experienced at least one `task.fail`, the fraction that eventually ended in `task.complete`.	**`quality_trend`** The average `quality_score` grouped by day, returning a time-series of dates, values, and sample sizes.

Hindsight: Resource Efficiency

context_efficiency

Formula:

```
sum(output_tokens) /
(sum(input_tokens) +
 sum(output_tokens))
```

Insight:

Measures how much context is consumed to produce an outcome.

cost_per_outcome

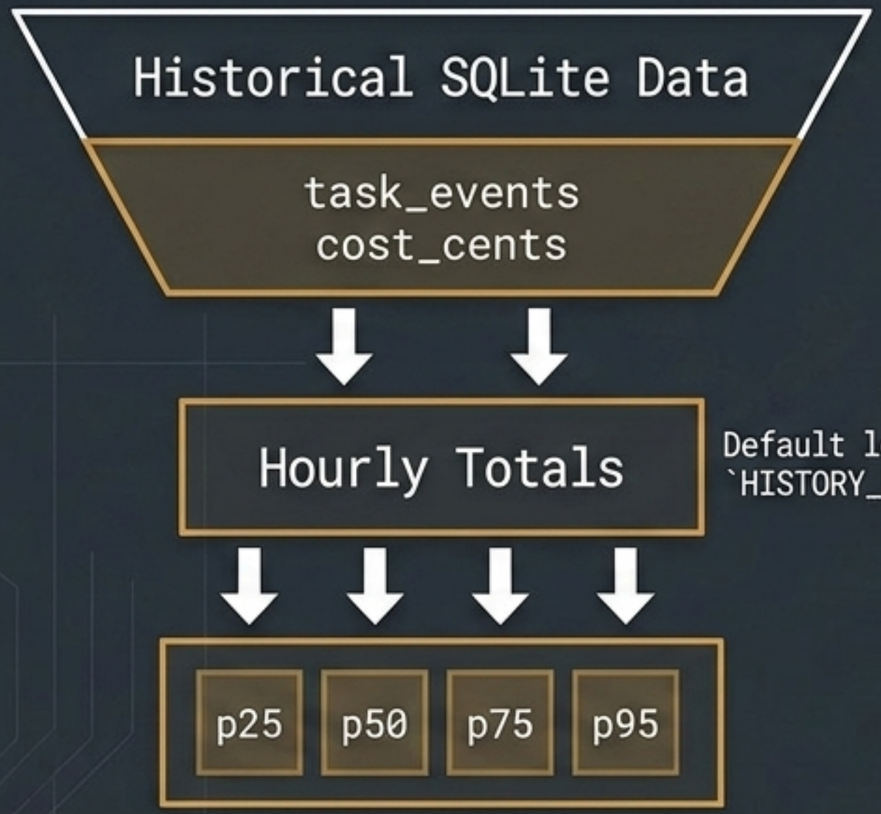
Formula:

```
sum(cost_cents over completed
        lifecycles) /
    count(task.complete)
```

Insight:

Tracks the true financial cost of successful work. The engine aggregates `cost_cents` carried on events.

Predictive Scaling Architecture



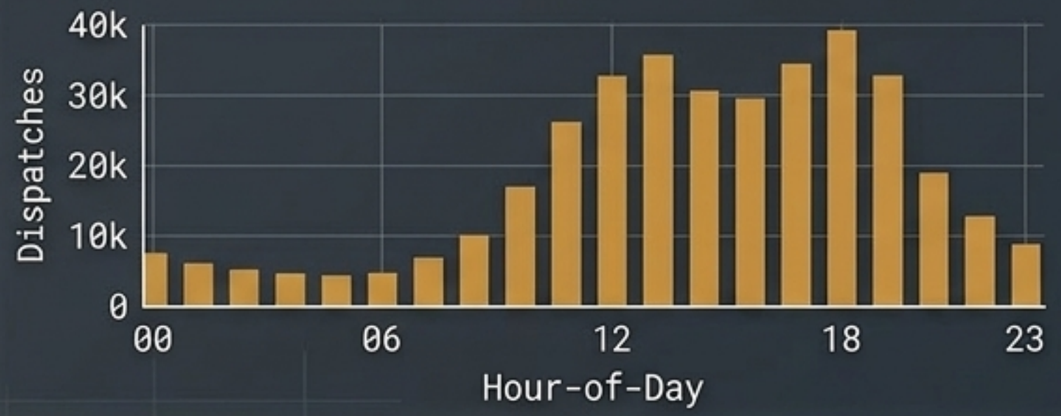
Default lookback controlled by `HISTORY_DAYS_FOR_DEMAND_CURVE=30`

Horizon-based Projections (24h, 7d, 30d)

- Statistical Only: Contains zero machine learning models.
- Predictive power relies purely on empirical quantiles and historical averages.
- Graceful Degradation: Forecasts return `insufficient_data` if fewer than 2 hourly buckets exist.

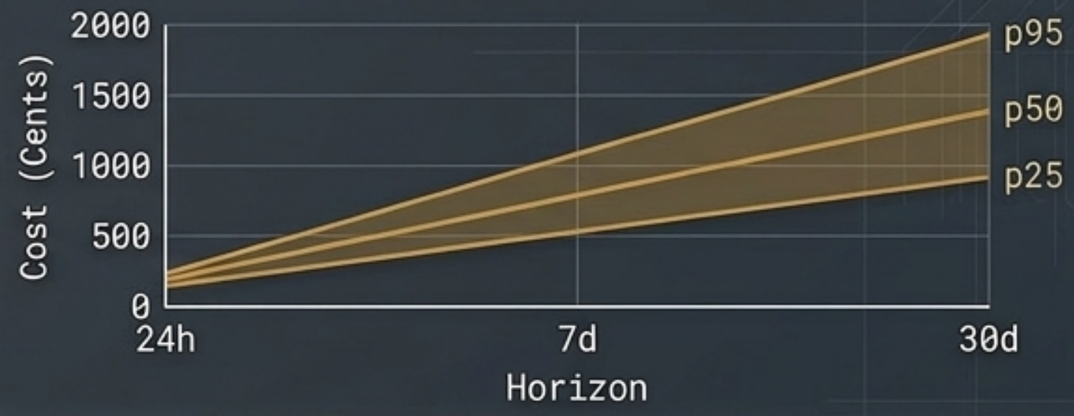
Foresight Outputs

Demand Curves



Average dispatches for hour[illegible]an.

Cost Forecast



Projected cents across selected horizons. Persisted with a confidence score.

Session Trajectory

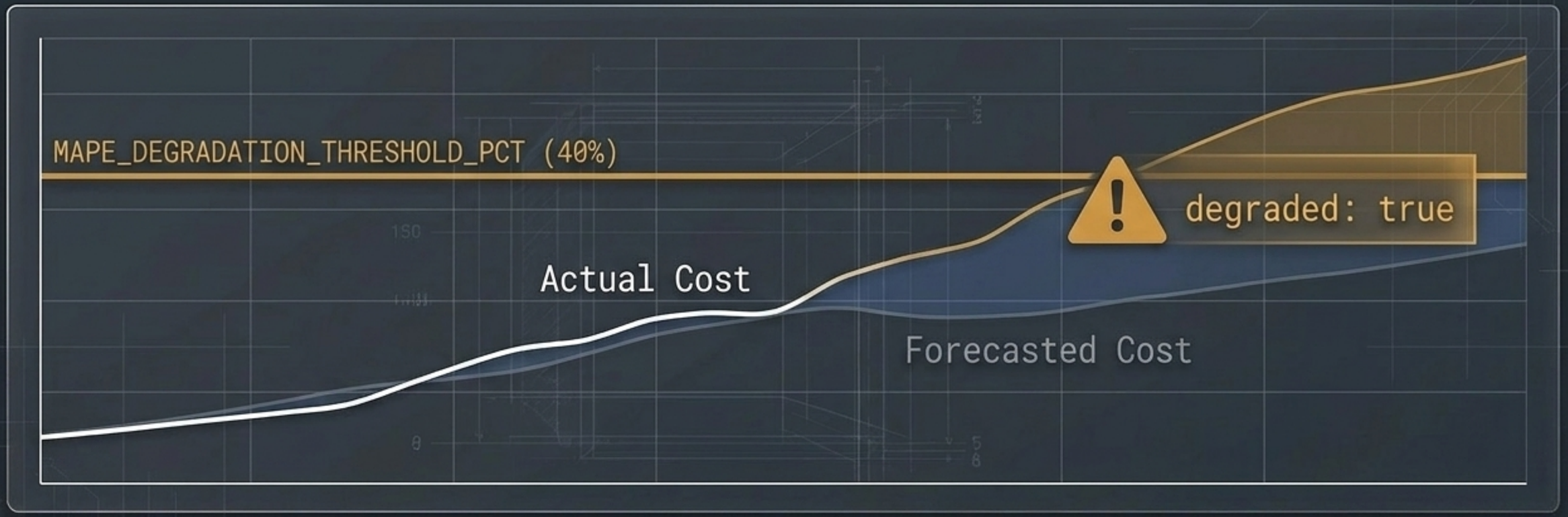
Requires minimum `SESSION_TRAJECTORY_MIN_TASKS` (default 3) before projecting.

Cache Pre-warm

1	agent-alpha-v1	frequency
2	agent-beta-v2	2 activity
3	agent-gamma-v1	2 activity
4	agent-delta-v3	5 calls/7d
5	agent-epsilon-v1	6 calls/7d

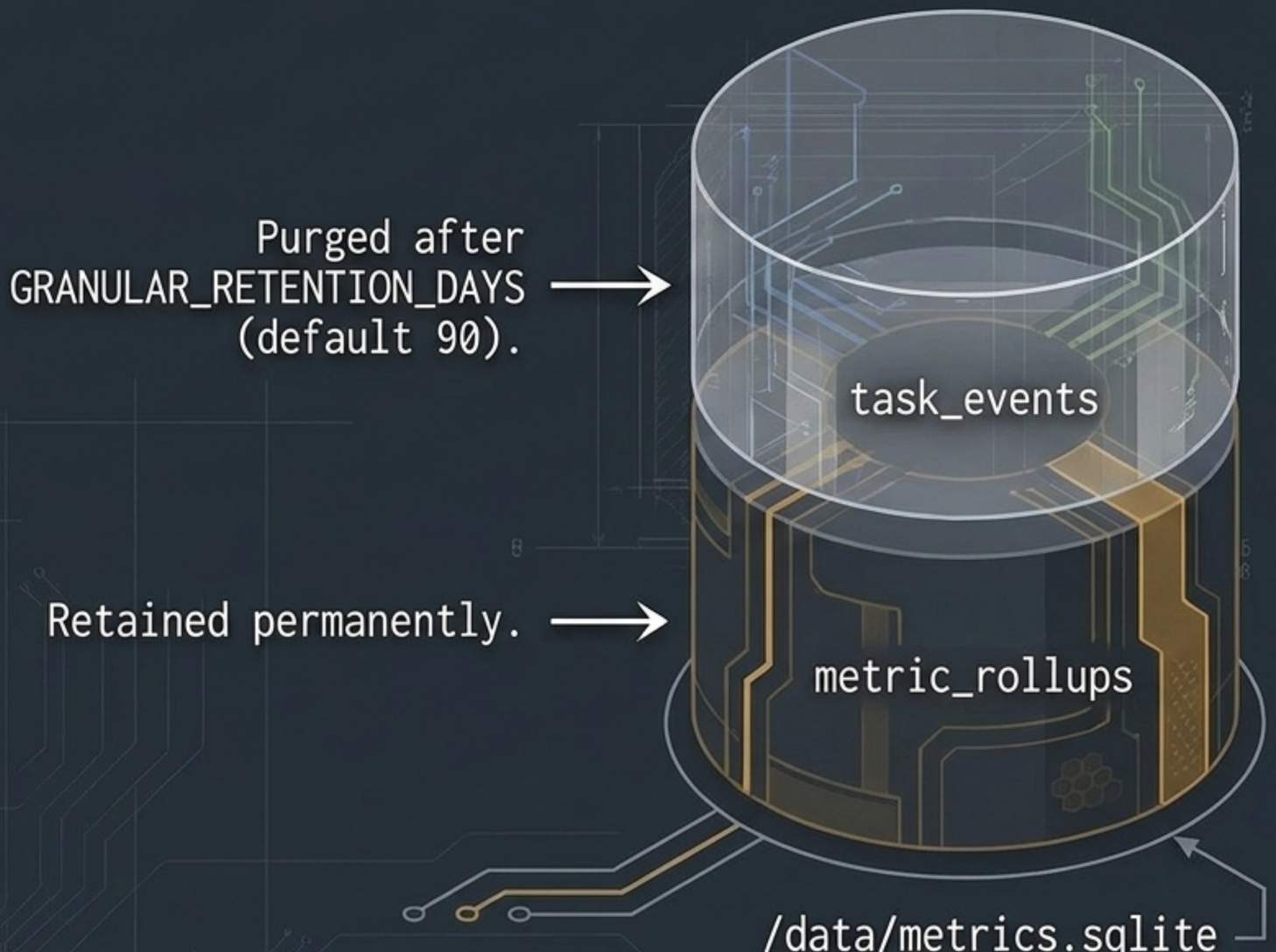
High-frequency agents (>5 calls/7d), targeted daily at `CACHE_PREWARM_HOUR` (UTC).

The MAPE Accuracy Monitor



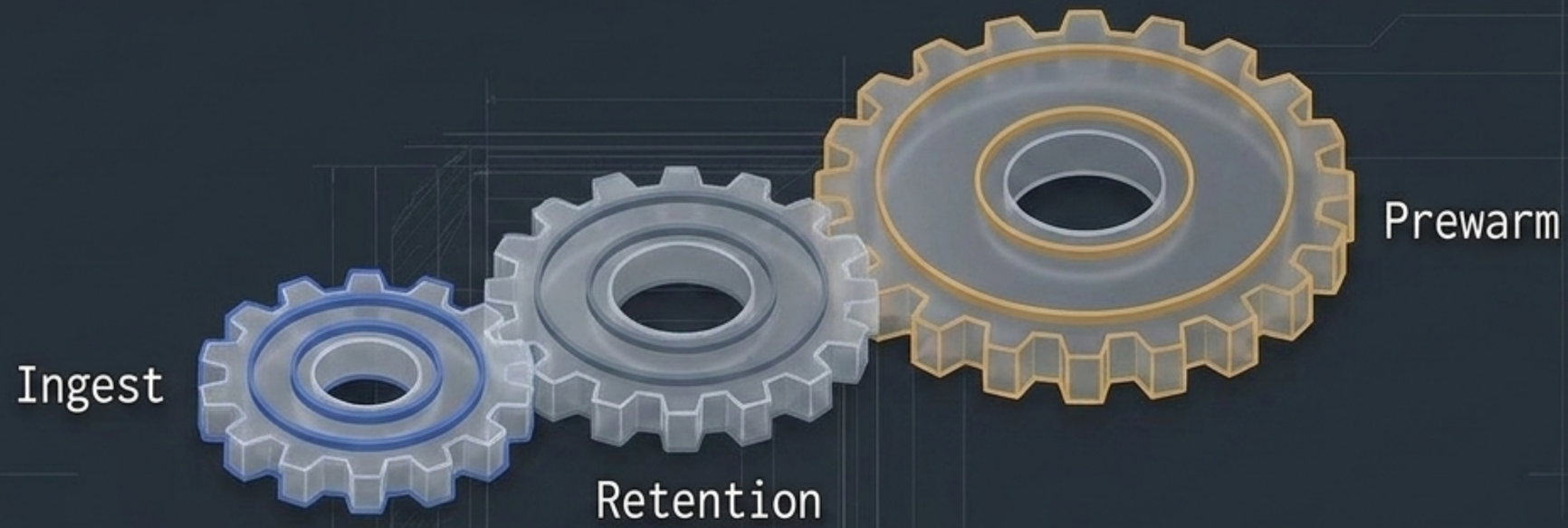
- The engine continuously calculates Mean Absolute Percentage Error (MAPE).
- Samples are logged to the `prediction_quality` table.
- If mean error exceeds 40%, the API flags forecasts as degraded, ensuring scaling systems never rely on blindly drifting projections.

Storage and Retention Policy



- **Single File Persistence**: All tables live in one SQLite WAL database.
- **Granular Retention**: Raw events are aggressively purged after 90 days.
- **Indefinite Rollups**: Before purging, daily aggregates are rolled up.
- **Schema Lifecycle**: Created idempotently on startup; zero complex migration frameworks.

The Operational Trinity



Ingest	Retention	Prewarm
_ingest_loop	_retention_loop	_prewarm_loop
Runs every AGGREGATE_INTERVAL_SEC	Runs hourly	Runs once daily at CACHE_PREWARM_HOUR
Pulls up to 5000 rows per pass. Safe no-op if router DB is missing.	Rolls up daily data and purges old granular events.	Uses timedelta(days=1) to avoid month-end errors.

Three asyncio loops govern the entire lifecycle, canceling cleanly on shutdown.

Secure by Default

Non-Root Container

Runs as `appuser` (uid 10001). Requires host volume ownership tuning for the `/data` mount.

Read-Only Operations

The event router mount is strictly `:ro`. The engine inherently cannot corrupt upstream data.

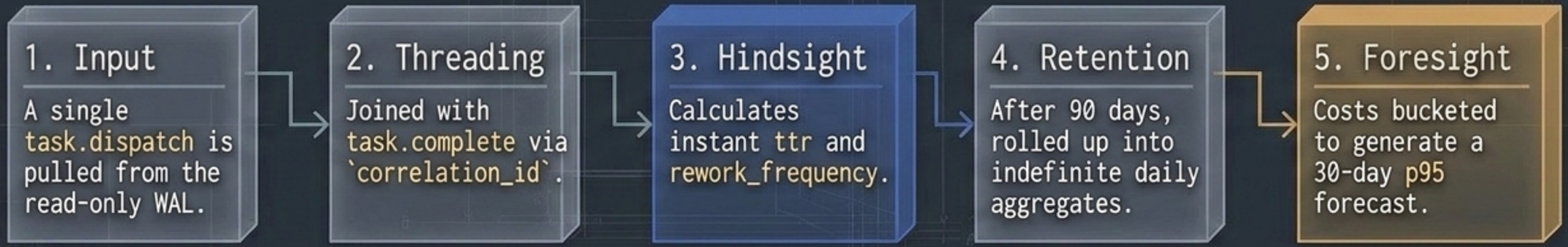
No Built-In Auth

APIs (e.g., `/api/events/ingest`) require network controls or ingress auth. Not intended for untrusted networks.

Supply Chain Integrity

Pinned GitHub Actions, pinned Python dependencies, and Subresource Integrity (SRI) hashes on Dashboard CDN scripts.

The Data-to-Wisdom Pipeline



From a scattered dispatch to a 30-day forecast. bulletproof-metrics-engine provides the statistical clarity required to scale agent operations reliably.