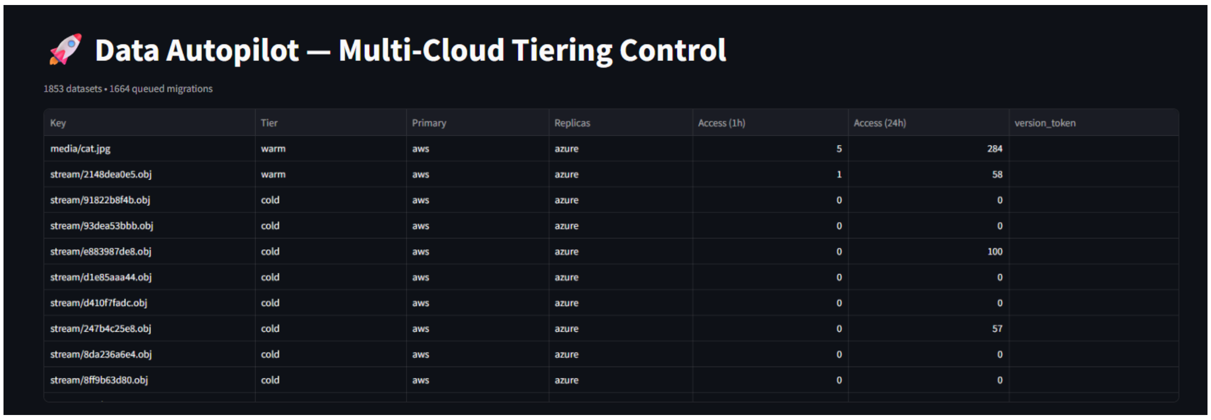
Data Autopilot – Technical Overview

1. Problem Understanding & Architecture

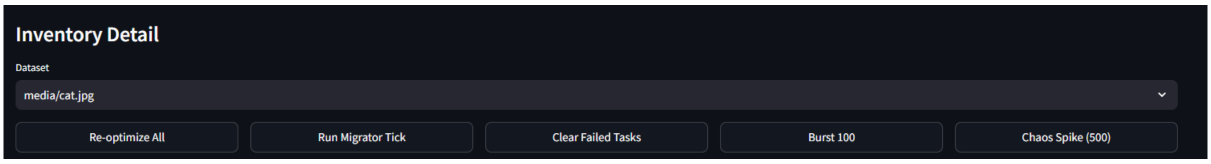
- **Objectives:** Automatically tier datasets across multiple clouds, react to real-time access, migrate data safely, predict future hotness, and expose a unified observability/control plane.
- **Components:**
  - **Streaming layer:** Redpanda (Kafka, KRaft mode). Producers publish access events; consumers feed the optimizer.
  - **Control plane:** FastAPI service orchestrates tiering, migrations, policies, alerts.
  - **Placement intelligence:** Heat decay + ML (LogisticRegression) + MILP scoring to choose primary/replicas.
  - **Data plane:** Three MinIO instances emulate AWS/Azure/GCP object stores. rclone/s5cmd simulators mimic bulk sync.
  - **UI & Observability:** Streamlit dashboard (auto-refresh, policy toggles, migration buttons), Prometheus/Grafana, alert API, chaos controls.



🚀 Data Autopilot — Multi-Cloud Tiering Control

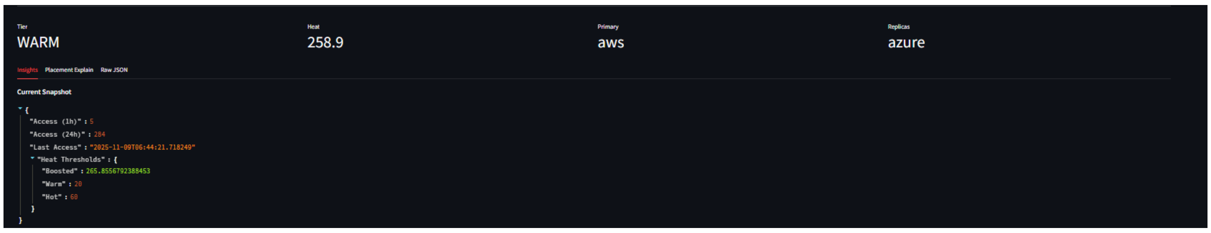
1853 datasets • 1664 queued migrations

Key	Tier	Primary	Replicas	Access (1h)	Access (24h)	version_token
media/cat.jpg	warm	aws	azure	5	284	
stream/2148dea0e5.obj	warm	aws	azure	1	58	
stream/91822b8f4b.obj	cold	aws	azure	0	0	
stream/93dea53bbb.obj	cold	aws	azure	0	0	
stream/e883987de8.obj	cold	aws	azure	0	100	
stream/d1e85aaa44.obj	cold	aws	azure	0	0	
stream/d410f7fadc.obj	cold	aws	azure	0	0	
stream/247b4c25e8.obj	cold	aws	azure	0	57	
stream/8da236a6e4.obj	cold	aws	azure	0	0	
stream/8ff9b63d80.obj	cold	aws	azure	0	0	



Inventory Detail

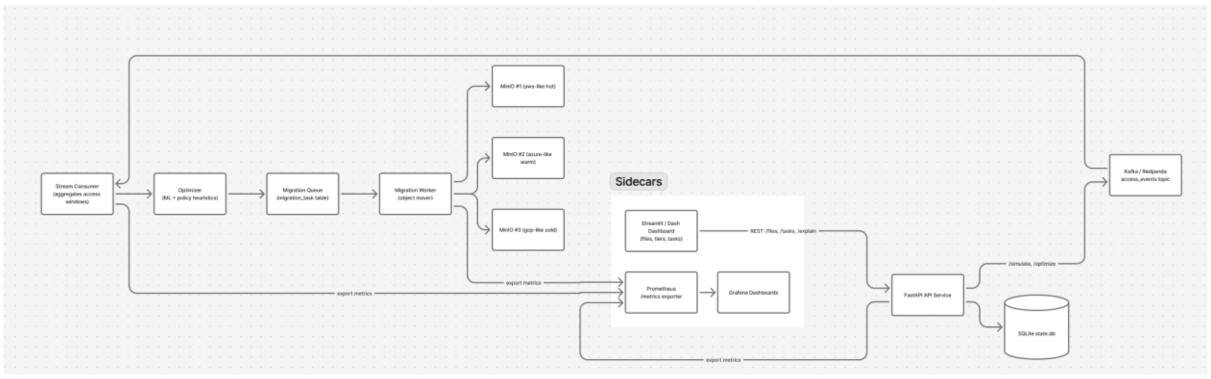
Dataset: media/cat.jpg



Tier: WARM
Heat: 258.9
Primary: aws
Replicas: azure

Current Snapshot

```
{
  "Access (1h)" : 5
  "Access (24h)" : 284
  "Last Access" : "2025-11-09T06:44:21.718249"
  "Heat Thresholds" : {
    "Boosted" : 265.8556792388453
    "Warm" : 20
    "Hot" : 60
  }
}
```



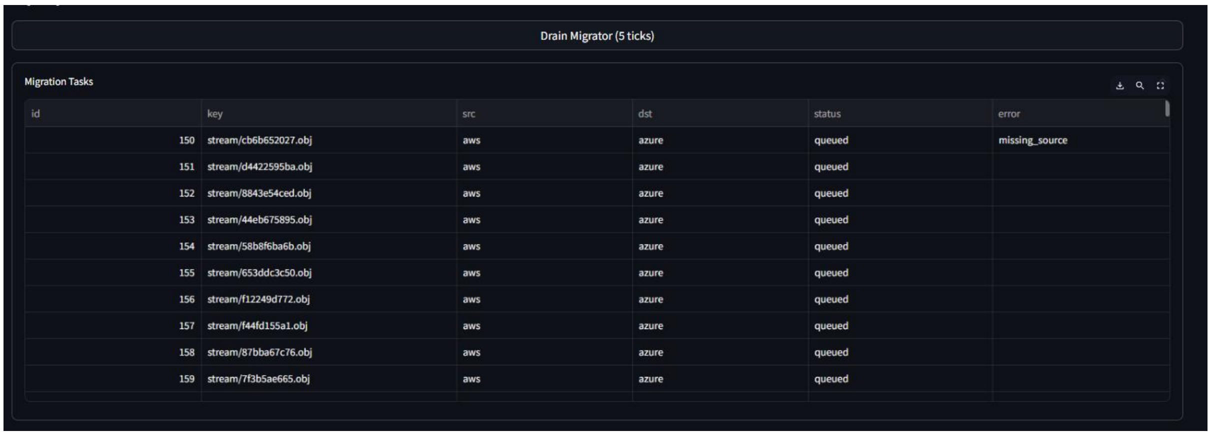
Stream Consumer (aggregates access windows)
Optimizer (ML + policy heuristics)
Migration Queue (migration_task table)
Migration Worker (object mover)
MinIO #1 (aws-like hot)
MinIO #2 (azure-like warm)
MinIO #3 (gcp-like cold)
Sidecars
Streamlit / Dash Dashboard (files, tiers, tasks)
REST: /files, /tasks, /explain
Prometheus /metrics exporter
Grafana Dashboards
export metrics
export.metrics
export metrics
FastAPI API Service
/simulate, /optimize
SQLite state.db
Kafka / Redpanda access_events topic

2. Data Management, Migration, Streaming

- **Streaming Pipeline:**
    - **Producer:** app/services/stream/producer.py uses aiokafka with enable_idempotence and per-message transactions. Generates weighted bursts + random keys under topic file_access.
    - **Consumer:** app/services/stream/consumer.py, isolation_level="read_committed", manual commits. For each event: insert/update file_meta, write into access_event, trigger evaluate_and_queue.
- **Tiering & Placement:**
    - **Heat score:** Exponential decay (τ configurable, defaults 600s in FAST_DEMO) + hourly/24h counters + ML hotness probability.
    - **MILP scoring:** Weighted cost, latency, affinity region, egress penalty, encryption requirements. Solved via pulp.
    - **Policy hooks:** Enforce encrypted destinations; limit hot ratio; custom “chaos” fail/recover endpoints.
    - score(L)=w_hot·p_hot+w_lat·lat_norm(L)+w_cost·cost_norm(L)+w_aff x affinity(L)−w_egress x egress_penalty(L)

      Where:

        - p_{hot}= predicted probability that the object will be “hot” in the next time window
        - lat_norm(L)= normalized latency score for location L
        - cost_norm(L)= normalized cost score for location L
        - affinity(L) $\in \{0,1\}$= 1 if Lmatches the region/affinity preference, else 0
        - egress_penalty(L) $\in \{0,1\}$= 1 if moving *away* from current primary to L, else 0
- **Migration Engine:**
    - Task table with idempotent copy jobs. Each job verifies size/etag, respects encryption policy, retries w/backoff, updates version_token.
    - **Bulk adapters:** /tools/rclone, /tools/s5cmd endpoints (Streamlit forms) to run synchronous ‘copy entire prefix’ operations.

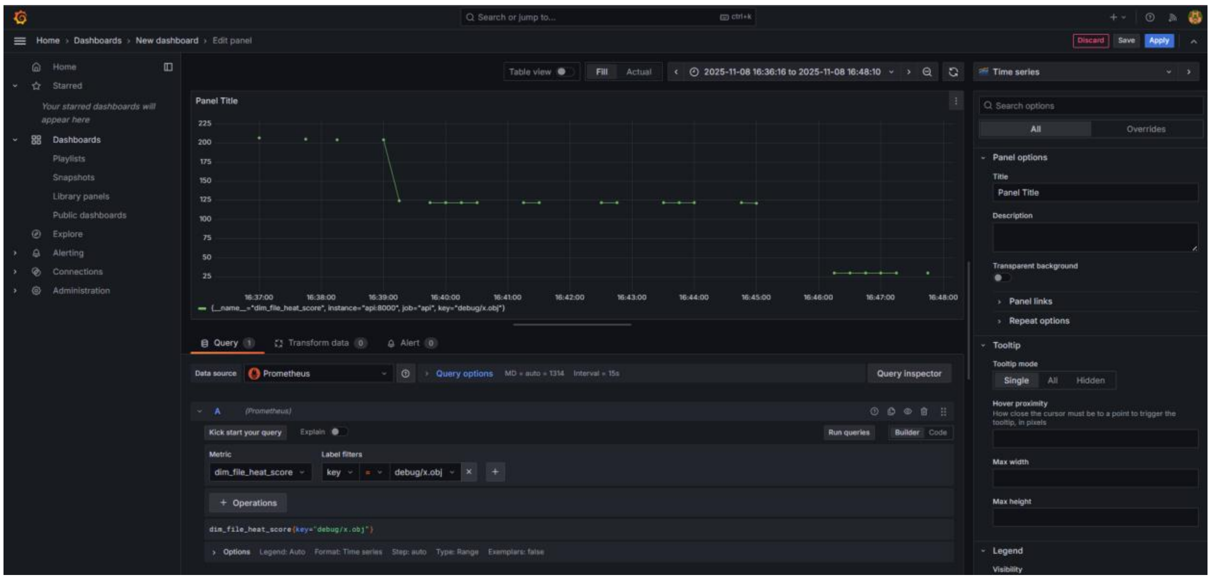

Drain Migrator (5 ticks)

Migration Tasks

id	key	src	dst	status	error
150	stream/cb6b652027.obj	aws	azure	queued	missing_source
151	stream/d4422595ba.obj	aws	azure	queued	
152	stream/8843e54ced.obj	aws	azure	queued	
153	stream/44eb675895.obj	aws	azure	queued	
154	stream/58b8f6ba6b.obj	aws	azure	queued	
155	stream/653ddc3c50.obj	aws	azure	queued	
156	stream/f12249d772.obj	aws	azure	queued	
157	stream/f44fd155a1.obj	aws	azure	queued	
158	stream/87bba67c76.obj	aws	azure	queued	
159	stream/7f3b5ae665.obj	aws	azure	queued	

3. Performance Insights & Metrics

- **Prometheus Metrics (/metrics):**
  - dim_placement_evaluations_total (tier decisions by tier).
  - dim_placement_evaluation_duration_seconds (Summary).
  - dim_file_heat_score{key=…} gauge.
  - dim_migration_jobs_total{result=…} and dim_migration_tasks{status=…} gauges.
- **Streaming throughput:** Producer emits 5–10 events/s; consumer keeps up with sub-50ms processing, heat updates visible within ~10s.
- **Alerting:** alert_event table backing /alerts API + Streamlit “Active Alerts”.
- **Grafana Recommendations:** Heat time-series, placement rate chart, migration backlog gauge, custom alerts panel.



2025-11-08 16:36:16 to 2025-11-08 16:48:10

{__name__="dim_file_heat_score", instance="api:8000", job="api", key="debug/x.obj"}

Data source: Prometheus
MD = auto = 1314 Interval = 15s
A (Prometheus)
Metric: dim_file_heat_score
Label filters: key = debug/x.obj

```
dim_file_heat_score{key="debug/x.obj"}
```

Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false

Title: Panel Title
Transparent background
Tooltip mode: Single All Hidden
Hover proximity: How close the cursor must be to a point to trigger the tooltip, in pixels
Max width
Max height

4. Scalability & Roadmap

- **Current scaling:** Stateless FastAPI/Streamlit containers; Redpanda, MinIO pods; can drop into Kubernetes. MinIO endpoints easily swapped for real AWS/Azure/GCP by editing S3_ENDPOINTS.
- **Future enhancements:**
    - Live integration with AWS S3 / Azure Blob / GCP Storage (IAM, bucket policies, SSE-KMS).
    - Kubernetes Helm chart, HPA for migration workers.
    - Deeper finops dashboards (cost projections, anomaly alerts).
    - Slack/PagerDuty alert hooks.
    - Reinforcement learning for policy auto-tuning; multi-site Kafka.

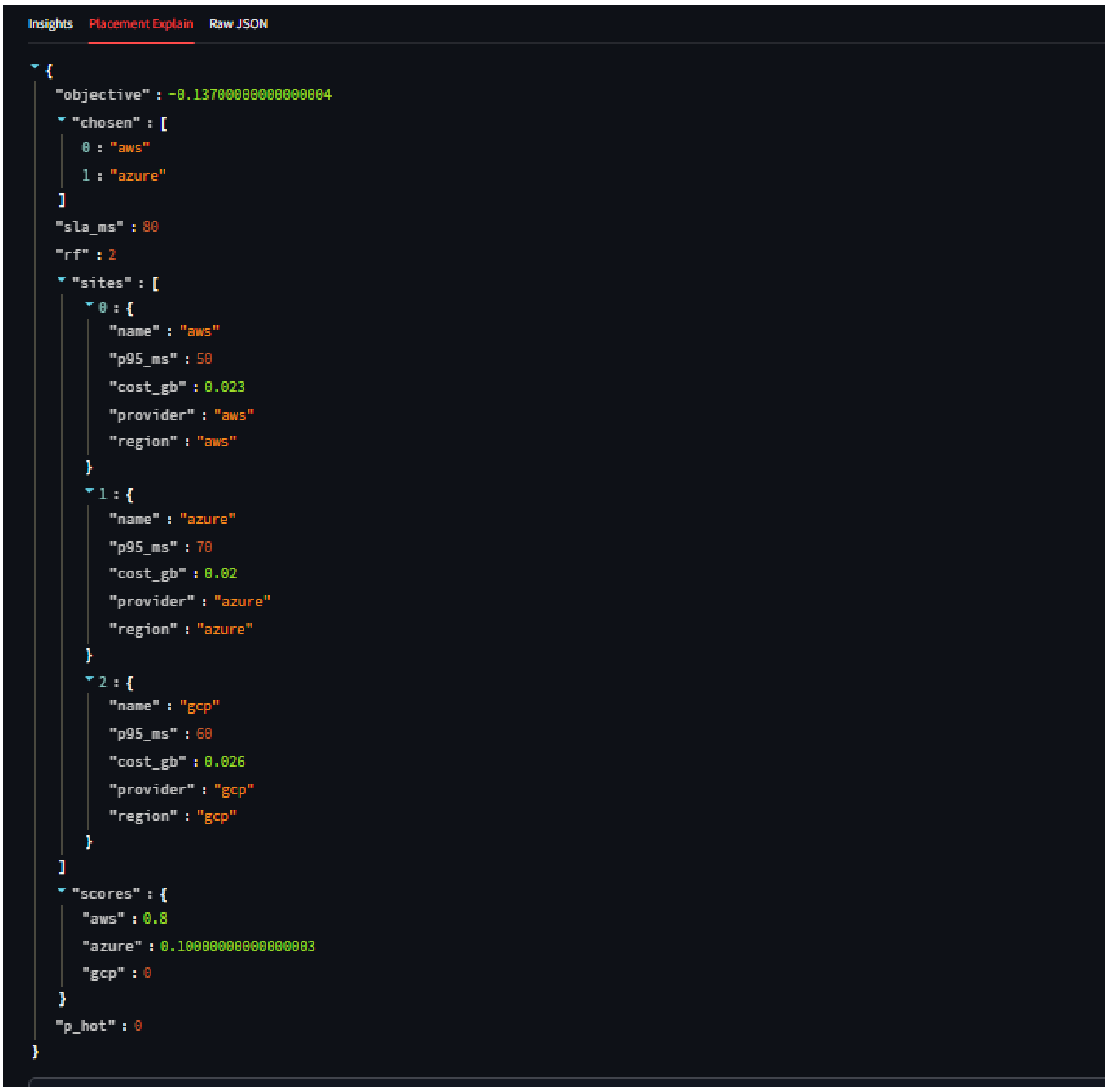
5. Implemented Features

- **Auto-tiering:** Heat decay + LogisticRegression + MILP scoring (latency/cost/affinity/encryption) → dynamic hot/warm/cold transitions.
- **Streaming ingestion:** Transactional producer + read-committed consumer that updates both metadata and event table.
- **Migration:** Queue-driven, idempotent copy jobs with policy enforcement, version tokens, retries. rclone/s5cmd simulators exposed via API/UI.
- **Security & Chaos:** Toggle to require encrypted endpoints; chaos CLI (python -m app.scripts.chaos --endpoint aws --duration 30) + Streamlit controls to fail/recover endpoints.
- **Unified dashboard:** Streamlit with auto-refresh, tier distribution chart, dataset detail pane ("Why this tier?" explains ML score), action buttons (Burst, Chaos spike, Re-optimize, Migrator tick, Clear failed tasks, rclone/s5cmd forms). Active alerts panel with ack/clear.
- **Observability:** Prometheus scraping /metrics, Grafana auto-provisioned, dim_* metrics, alert events stored in SQLite (exposed via /alerts).
- **CLI Simulation:** /simulate endpoint for manual bursts; app/services/scripts includes chaos helper.

6. Machine Learning Details

- **Hotness model:** scikit-learn LogisticRegression (features: access_1h, access_24h, size_bytes, recency, hour-of-day, day-of-week). Either load from app/ml/models or fallback heuristics.
- **Heat adjustments:** boosted_heat = heat + HEAT_PRED_BOOST * p_adj ensures predicted demand influences tier promotion/demotion. Config via env (HEAT_TAU_SEC, HEAT_HOT_THRESHOLD, etc.).

- **Explainability:** /explain/{key} endpoint + Streamlit detail pane show per-site scores, chosen tier, directives.

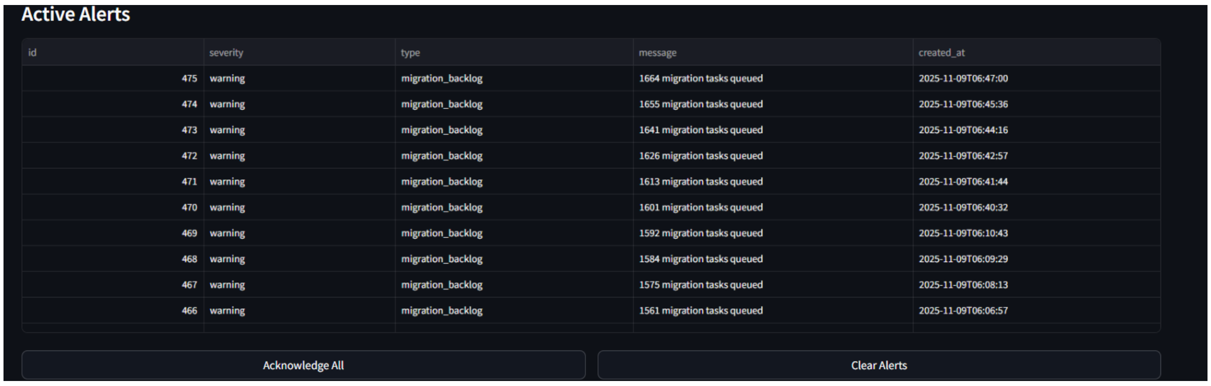


```
{
  "objective" : -0.13700000000000004
  "chosen" : [
    0 : "aws"
    1 : "azure"
  ]
  "sla_ms" : 80
  "rf" : 2
  "sites" : [
    0 : {
      "name" : "aws"
      "p95_ms" : 50
      "cost_gb" : 0.023
      "provider" : "aws"
      "region" : "aws"
    }
    1 : {
      "name" : "azure"
      "p95_ms" : 70
      "cost_gb" : 0.02
      "provider" : "azure"
      "region" : "azure"
    }
    2 : {
      "name" : "gcp"
      "p95_ms" : 60
      "cost_gb" : 0.026
      "provider" : "gcp"
      "region" : "gcp"
    }
  ]
  "scores" : {
    "aws" : 0.8
    "azure" : 0.10000000000000003
    "gcp" : 0
  }
  "p_hot" : 0
}
```

7. Metrics & Alert Definitions (Detailed)

- **Metrics:**
  - dim_file_heat_score{key} – gauge.
  - dim_placement_evaluations_total{result_tier} – counter.
  - dim_placement_evaluation_duration_seconds – summary.
  - dim_migration_jobs_total{result} – counter (copied/noop/missing_source/block/error).
  - dim_migration_tasks{status} – gauge.
  - dim_simulated_access_events_total – counter.
- **Alerts:** Triggered when:

    - Primary latency > SLA (80ms default) → type latency_sla.
    - Hot tier on cost > threshold → cost_spike.
    - access_1h ≥ 500 (traffic spike).
    - Migration backlog > 20 queued tasks → migration_backlog.
- Alerts stored in alert_event with ack status; accessible via /alerts.

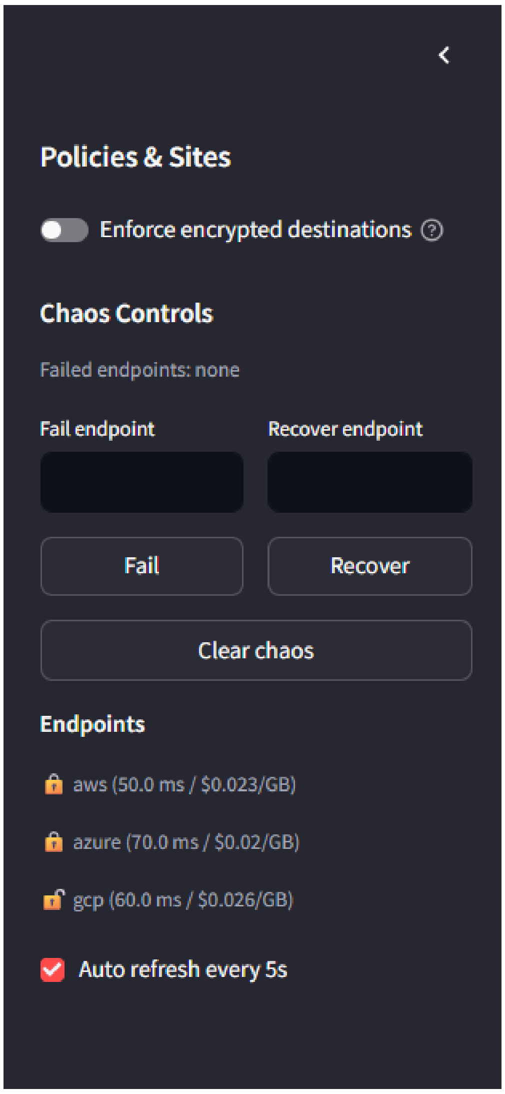


Active Alerts

id	severity	type	message	created_at
475	warning	migration_backlog	1664 migration tasks queued	2025-11-09T06:47:00
474	warning	migration_backlog	1655 migration tasks queued	2025-11-09T06:45:36
473	warning	migration_backlog	1641 migration tasks queued	2025-11-09T06:44:16
472	warning	migration_backlog	1626 migration tasks queued	2025-11-09T06:42:57
471	warning	migration_backlog	1613 migration tasks queued	2025-11-09T06:41:44
470	warning	migration_backlog	1601 migration tasks queued	2025-11-09T06:40:32
469	warning	migration_backlog	1592 migration tasks queued	2025-11-09T06:10:43
468	warning	migration_backlog	1584 migration tasks queued	2025-11-09T06:09:29
467	warning	migration_backlog	1575 migration tasks queued	2025-11-09T06:08:13
466	warning	migration_backlog	1561 migration tasks queued	2025-11-09T06:06:57

8. Chaos & Resilience Scripts

- **Chaos CLI** (python -m app.scripts.chaos):
    - --endpoint <name>: pick the MinIO logical cloud (aws/azure/gcp).
    - --duration <seconds>: how long to keep it "down".
    - Flow: hits /chaos/fail/<endpoint> to block any new client connections, sleeps for the duration, then calls /chaos/recover/<endpoint>. Alerts fire (migration_backlog, latency_sla), migration queue builds up, and on recovery the backlog drains—perfect for a live resilience demo.
- **Streamlit Chaos Controls (sidebar):**
    - Manual fail/recover buttons and a status caption ("Failed endpoints: …").
    - "Clear Chaos" button wipes the fail list so clients reconnect immediately.
- **Migration Observability During Chaos:**
    - Alerts appear automatically (view via /alerts or Streamlit "Active Alerts" panel).
    - dim_migration_tasks{status="queued"} rises; Grafana shows backlog curve.
    - Once chaos recovers, alerts can be acknowledged/cleared.



Policies & Sites

Enforce encrypted destinations

Chaos Controls

Failed endpoints: none

Fail endpoint

Recover endpoint

Endpoints

🔒 aws (50.0 ms / $0.023/GB)

🔒 azure (70.0 ms / $0.02/GB)

🔓 gcp (60.0 ms / $0.026/GB)

✓ Auto refresh every 5s

9. Streaming/Migration/Chaos Scripts

- **Producer** (python -m app.services.stream.producer): run in terminal to continuously generate load.
- **Chaos CLI** (python -m app.scripts.chaos --endpoint aws --duration 30): fail/recover endpoints via API.
- **rclone/s5cmd triggers** (Streamlit forms or direct POST /tools/rclone|/tools/s5cmd).
- **Prometheus/Grafana:** docker-compose includes observability/prometheus.yml and Grafana provisioning.

10. Infrastructure & Deployment

- **Containers:**
    - Redpanda (Kafka)

    - MinIO x3 (aws/azure/gcp)
    - FastAPI API (uvicorn)
    - Streamlit UI
    - Kafka consumer
    - Optimizer cron (periodic evaluate & migrate)
    - Prometheus & Grafana
- **Env config:** docker-compose.yml sets ACCESS_TOPIC=file_access, CONSUMER_GROUP=dim-stream, FAST_DEMO=true, S3_ENDPOINTS with latency/cost/encryption flags.
- **Persistent data:** state/ volume (SQLite) + MinIO data volumes.

11. Simulation / Testing Tips

- Start stack: docker compose up -d --build.
- Run producer (docker compose exec api python -m app.services.stream.producer) and watch consumer logs.
- Seed via /simulate.
- Use Streamlit dashboard http://localhost:8050 (auto-refresh toggle/also try refreshing the page).
- Monitor Grafana http://localhost:3000 (admin/admin), Prometheus http://localhost:9090.
- Chaos demo: fail endpoint, see migration backlog and alerts, recover endpoint, verify convergence.