Enterprise AI Ops

A Scaling Playbook for Industrial Operators

From a single command-center copilot to a Databricks-anchored, edge-distributed decision intelligence platform.

Author	Ila Brandt DeLany — Solution Architect, YTS
Audience	CIO / COO / VP Operations and their solution architects
Reference build	YTS AI OPS Command (aiops.brotherila.com)
Maturity stage	Stage 2 — Advisory Copilot, with a path to Stage 3+
Version	1.0 · May 2026

1. Executive Summary

Industrial operators are converging on a repeatable architecture for AI in operations. The pattern fuses live multi-source telemetry into a normalized state, runs an LLM with tool-calling and structured output as a reasoning layer, and surfaces ranked, cited, confidence-scored recommendations in a single command surface. YTS AI OPS Command is a production reference of this pattern.

This playbook shows where it sits in the market, how it scales from a single tenant to a Databricks-backed enterprise lakehouse with edge inference at the asset, and what it takes to graduate from an advisory copilot to approved-action and eventually closed-loop autonomy.

Is this common?

Yes. Every major industrial software stack now ships some version of it: Microsoft Copilot for Industry, Palantir AIP, C3 AI, Cognite Data Fusion + Atlas AI, GE Vernova SmartSignal, Siemens Industrial Copilot, AWS Bedrock Agents for Industrial, and Databricks Mosaic AI Agent Framework. The shape is identical — only the substrate differs.

The four-layer pattern

Layer	Job	What YTS uses today
Context	Fuse live + historical signals into one state	EIA, NOAA, USGS, Open-Meteo + internal facility telemetry
Reasoning	Rank, explain, recommend — with citations	Gemini 3 Flash via Lovable AI Gateway, structured tool-call output
Action	Execute or stage actions with HITL	Read-only today; Slack/email/MOC drafts gated behind architect
Trust	Audit, RBAC, confidence, replay	Confidence %, cited rationale, named-author guardrails

Maturity curve

Stage	Name	Behavior	Risk profile
1	Descriptive	Dashboards, alerts on thresholds	Low — status quo
2	Advisory copilot	LLM ranks, explains, recommends. Human acts.	Low — read-only
3	Approved action	Agent drafts and executes after human approval	Medium — needs MOC + audit
4	Closed-loop	Agent acts within bounded envelopes; human on exception	High — SIS/PHA gated

YTS AI OPS Command sits at Stage 2. That is intentional and correct — no credible O&G; operator is shipping Stage 4 today. The playbook below describes how to scale horizontally (more facilities, more feeds) and vertically (toward Stage 3+).

2. Reference Architecture

The diagram below shows the canonical topology. The YTS reference implementation is the right-most column compressed into a single Cloudflare Worker; the scaled topology fans the same pattern across edge gateways and a Databricks lakehouse.

Topology

```
EDGE (per facility) CORE (Databricks) EXPERIENCE
------------------- --------------------- -----------------
PLC / SCADA / Historian --> Bronze (Auto Loader) --> AI OPS Command
IIoT sensors / vibration --> Silver (DLT, schema) --> Mobile field app
Local inference (ONNX) --> Gold (feature store) --> Ops chat (RAG)
|
v
Mosaic AI Agent Framework
- tool calling
- vector search (UC)
- model serving
|
v
Unity Catalog + Audit (Lakehouse)
```

Figure 1 — Edge-to-lakehouse-to-experience reference topology.

Component responsibilities

Tier	Responsibility	Reference tech
Edge gateway	Buffer telemetry, run sub-second inference, survive WAN loss	Cloudflare Workers / AWS Greengrass / Azure IoT Edge + ONNX Runtime
Streaming ingest	At-least-once delivery, schema evolution	Kafka / Event Hubs → Databricks Auto Loader
Lakehouse	Bronze/Silver/Gold, governed by Unity Catalog	Databricks DLT, Delta, Unity Catalog
Feature + vector	Online features, doc embeddings for RAG	Databricks Feature Store, Vector Search
Reasoning	LLM with tool calls, structured output, evals	Mosaic AI Agent Framework, Model Serving, MLflow 3
Experience	Command center, mobile, chat, voice	Lovable / TanStack Start / React Native

Why Databricks for the core

- **One governed substrate.** Unity Catalog gives you table-, column-, and row-level RBAC across telemetry, documents, and model outputs — the same policy engine for the data the LLM reads and the actions it can recommend.
- **Lakehouse + agents in one plane.** Mosaic AI Agent Framework, Vector Search, Model Serving, and MLflow 3 evals run next to your Delta tables. No data egress to a separate AI vendor for the hot path.

- **Open formats.** Delta + Iceberg means you are not locked into a proprietary timeseries store. Historian data lands once and serves BI, ML, and agents.
- **Geo + multi-cloud.** A Houston refinery, an Oregon hub, and a Permian field can each land into a regional workspace and federate through Unity Catalog.

3. Edge Strategy

Edge is non-negotiable in physical operations. Refineries and offshore platforms cannot depend on a clean WAN to a cloud region for safety-relevant decisions. The edge tier handles three jobs the cloud cannot.

What runs at the edge

Job	Latency budget	Pattern
Vibration / acoustic anomaly	< 100 ms	ONNX model on gateway, raises local alarm, streams features upstream
Vision (PPE, leak, flare)	< 500 ms	Quantized YOLO/ViT on Jetson or Greengrass, posts events
Setpoint guardrail	< 1 s	Rule + small classifier, vetoes unsafe operator change before sending to PLC
Store-and-forward	Indefinite	Buffer telemetry to local Delta, replay to Bronze when WAN returns

Edge ↔ Lakehouse contract

- **One schema, two tiers.** The edge writes the same Avro/Protobuf schema it streams to Kafka. Auto Loader lands it into Bronze without translation.
- **Models flow down, features flow up.** Train in Databricks, register in Unity Catalog, export to ONNX, sign, push to gateways via OTA. Edge emits the features it computed back into the lakehouse so training and serving see the same view.
- **Disconnected by default.** A facility that loses cloud for 48 hours keeps operating, keeps alarming, and reconciles on reconnect. The copilot at HQ degrades gracefully — it shows the facility as stale, not silent.

Cloud vs edge decision rule

```
if (latency < 1s) OR (safety-relevant) OR (bandwidth-bound):   run at edge
elif (needs cross-facility context) OR (LLM reasoning):       run in lakehouse
else:                                                         default to lakehouse
```

4. Scaling the Reasoning Layer

From one Gateway call to an agent fleet

The YTS reference makes a single LLM call with tool-forced JSON output. To scale, the same pattern decomposes into specialized agents that share context and tools.

Agent	Owns	Tools it can call
Briefing agent	Executive summary, KPIs	metrics.query, market.snapshot
Predictive agent	Asset health, RUL	telemetry.window, model.predict_rul, work_order.lookup
Logistics agent	Shipments, terminals	shipment.status, weather.marine, route.replan
Compliance agent	Permits, MOC, PHA	doc.search (RAG), permit.status, moc.draft
Orchestrator	Routes user intent, merges outputs	all of the above

On Databricks: each agent is a Mosaic AI Agent (LangGraph or built-in framework), registered in Unity Catalog, served via Model Serving, evaluated with MLflow 3 LLM evals on a curated golden set of ops scenarios.

Retrieval

- **Structured retrieval first.** SQL against governed Gold tables beats vector search for anything with a number in it (production, downtime, shipments).
- **Vector for unstructured.** P&IDs;, MOCs, PHA, OEM manuals, incident reports — indexed in Databricks Vector Search, chunked with metadata for facility, asset, and date.
- **Hybrid is the default.** The orchestrator gets both a SQL tool and a doc.search tool and learns when to use which through evals.

Guardrails that scale

Guardrail	How it works	Where enforced
RBAC on data	Unity Catalog row/column policies	Lakehouse — the agent inherits the user's identity
RBAC on actions	Tool allow-lists per role	Agent framework
Output schema	JSON schema / tool-forced output (your current pattern)	Model Serving
Citation required	Reject responses without source IDs	Post-processor
Confidence floor	Suppress < N% confidence from auto-paging	UI / routing layer
Audit trail	Every prompt, tool call, and response to Delta	MLflow tracing + Unity Catalog

5. Rollout Plan

Phased rollout (typical 9-12 months)

Phase	Months	Deliverable	Exit criteria
0. Foundation	0–2	Unity Catalog, Bronze ingest from 1 facility, audit baseline	One facility's telemetry queryable, governed, observable
1. Advisory copilot	2–4	Single-call reasoning over Gold, command center UI	Ops team uses daily; >70% of recs rated useful in evals
2. Multi-agent	4–7	Split into 4–5 agents, RAG over docs, mobile surface	Mean time-to-insight drops; agents pass golden-set evals
3. Edge inference	6–9	Per-facility gateway, anomaly + vision models, store-and-forward	Sub-second local alarms; survives 24h disconnect
4. Approved action	9–12	HITL drafts MOC, work orders, comms; one-click execute	First closed-loop workflow with full audit and rollback

Team you need

- **Solution architect (1).** Owns the contract between edge, lakehouse, and experience. Today: Ila Brandt DeLany.
- **Data + platform engineer (1–2).** DLT pipelines, Unity Catalog, OTA model deploy.
- **ML / agent engineer (1–2).** Agent framework, evals, RAG, fine-tuning when warranted.
- **Edge engineer (0.5–1).** Gateway runtime, ONNX, OTA, OT-network safety.
- **Ops domain SME (embedded).** Owns the golden eval set and the rec-quality rubric. This role is the most underestimated and the most important.

Cost shape (order-of-magnitude)

Bucket	Phase 1–2	Phase 3–4
Databricks compute (DBUs)	Small all-purpose + serverless SQL	+ DLT continuous, Model Serving, Vector Search
LLM tokens (gateway or Mosaic)	Bursty, ~$ low-thousands/mo	Steady, ~$ low-tens-of-thousands/mo at multi-facility
Edge hardware	—	$3–8k per gateway, one-time + lifecycle
People	2–3 FTE	4–6 FTE + embedded SME time

6. The YTS Reference, Mapped to This Playbook

The current YTS AI OPS Command implementation is a clean Stage-2 advisory copilot. Here is where each piece sits and what it becomes at scale.

Today (YTS reference)	Same role at scale (Databricks)
Live feeds: EIA, NOAA, USGS, Open-Meteo fetched in a server function	Auto Loader → Bronze; scheduled DLT pipelines for external feeds
In-memory facility/telemetry state (mock + sim)	Silver Delta tables fed by historian/SCADA via Kafka + Auto Loader
contextToPrompt(): hand-rolled context window	Mosaic AI Agent retrieves from Gold tables + Vector Search
Single Gemini 3 call with tool-forced JSON	Orchestrator + specialized agents on Model Serving, evaluated in MLflow
UI: TanStack Start command center, ops chat	Same UI, now reading from governed agent endpoints
Guardrails: confidence %, cited rationale, named-architect for actions	+ Unity Catalog RBAC, audit to Delta, MLflow tracing, MOC integration

What to do next on this build

- Land one real telemetry feed (even synthetic at fixed cadence) into a Delta table; point the briefing function at it. Proves the lakehouse path end-to-end.
- Add an MLflow eval harness with 20–30 golden ops scenarios. This becomes the regression suite for every model and prompt change.
- Split the single LLM call into briefing + predictive + logistics agents behind the same command surface. No UI change, big quality lift.
- Stand up one edge gateway (a Raspberry Pi or Jetson is fine for the demo) running an ONNX anomaly model. Show store-and-forward to the lakehouse on reconnect.

— End of playbook —