hiflylabs

Apps Factory

Build, run, and monitor 10, 50, or 1,000+ agent-native apps on Databricks with enterprise reliability, at startup speed.

Templates, quality gates, readiness checklists, cost control, observability setup, and more...

Scale from 1 to 1000 apps. Fast, controlled, self-sufficient.

Picture this:

Management wants to see what delays customer service in real time. Your Data team ships an analytics agent that lets them query live operational data in plain English.

Floor supervisor needs to hit daily output targets without downtime. Your Engineering team spins up a real-time monitoring app that reads conveyor belt data and sends alerts the moment cycle time drifts beyond the threshold.

Both applications provisioned in hours, shipped in days, governed by the same rules that govern your Lakehouse. Everything is observable, auditable, and cost-accountable from a single control panel. All done in-house—you don't need external development help or weeks of approvals from security and IT.

Now imagine it's not just 2 apps, but 4, 16, 256…

With Databricks Apps slashing development cycles from months to weeks, and agentic engineering further shrinking the timelines to days, this is a realistic scenario.

But with great speed gains come even greater operational challenges.

Every new app without a shared standard makes the next one harder to manage, govern, and run.

What happens when you meet the operational reality

When your organization outgrows sporadic PoCs and AI experiments, you inevitably face a compounding problem that gets harder with every workflow and app added to the count. Moving from 1 or 2 to 10, 50, or even 1000+ agent-native apps is a complex operational challenge.

Does this look familiar?

Every team builds differently. Without **shared templates, standardized tooling**, and **production readiness gates**, every new build starts from scratch. Your **architecture** is not prepared for an ever-growing scale.

No **unified CI/CD standards, governance**, and **logging**, so everything is manual and inconsistent. No **clear ownership** and **access control**, so it's impossible to find out who's accountable when something breaks.

You can't see what's going on. Without **observability**, you have no way to prevent issues before they reach production. Without precise **cost attribution**, you don't know where you spend and can't spot cost leaks.

So how do you scale from two apps to hundreds and not lose governance, cost control, or your team's sanity?

Few organizations have solved it

Only **13%** of organizations see AI agents integrated into broader workflows. BCG, June 2025

- **Enterprise adoption is slow and doesn't live up to the hype:** No operational layer means no way to scale automation safely.
- **IT controls block self-service enablement:** Business and engineering move at different speeds. IT review rounds block business from moving at a competitive pace.
- **Integration complexity compounds.** Fragmented platforms, legacy data sources, and undocumented dependencies build up the challenge.
- **AI spend grows invisibly without observability.** Token cost goes up, IT has no way to track it, audit it, or know who owns it.

Apps Factory

One operational framework to build and run hundreds of agent-native apps on Databricks

What is Apps Factory?

Apps Factory is a 4-week review and setup program that makes your Databricks environment enterprise-ready and gives your teams everything they need to self-sufficiently build secure, repeatable data and AI applications fast—without reinventing patterns and hiring external help.

- Tooling
- Standards
- Checklists
- Templates
- Guardrails
- Boilerplates
- Repositories
- Observability
- Cost monitoring

100+ dev days per application saved

3× faster time to deploy

4 weeks to set up

Apps Factory is the bridge that helps your teams deliver business applications faster—reliably, repeatably, at scale.

What does Apps Factory do?

Apps Factory resolves the compounding challenge of growing from 5 or 10 to 1000+ apps. It creates the environment where teams across data, AI, engineering, and business departments can independently build and deploy AI applications and agentic workflows. At speed, but without sacrificing enterprise quality and standards.

Fast time to production

Standardized app skeletons, reusable templates, repeatable CI/CD pipelines, and clear framework selection guidance remove the need to start from scratch.

Security and governance by design

Standardized OBO auth, RBAC, and service principal patterns paired with Unity Catalog policies fully aligned to app access, enforced as repeatable policy-as-code.

Clear ownership and accountability

Defined RACI, production-readiness checklists, quality gates, and lifecycle management for every app.

Consistent architecture

Unified templates for analytics apps, chat and agent apps, and Genie-powered applications. Every team builds the same way, regardless of who built the first one.

Observability and cost control

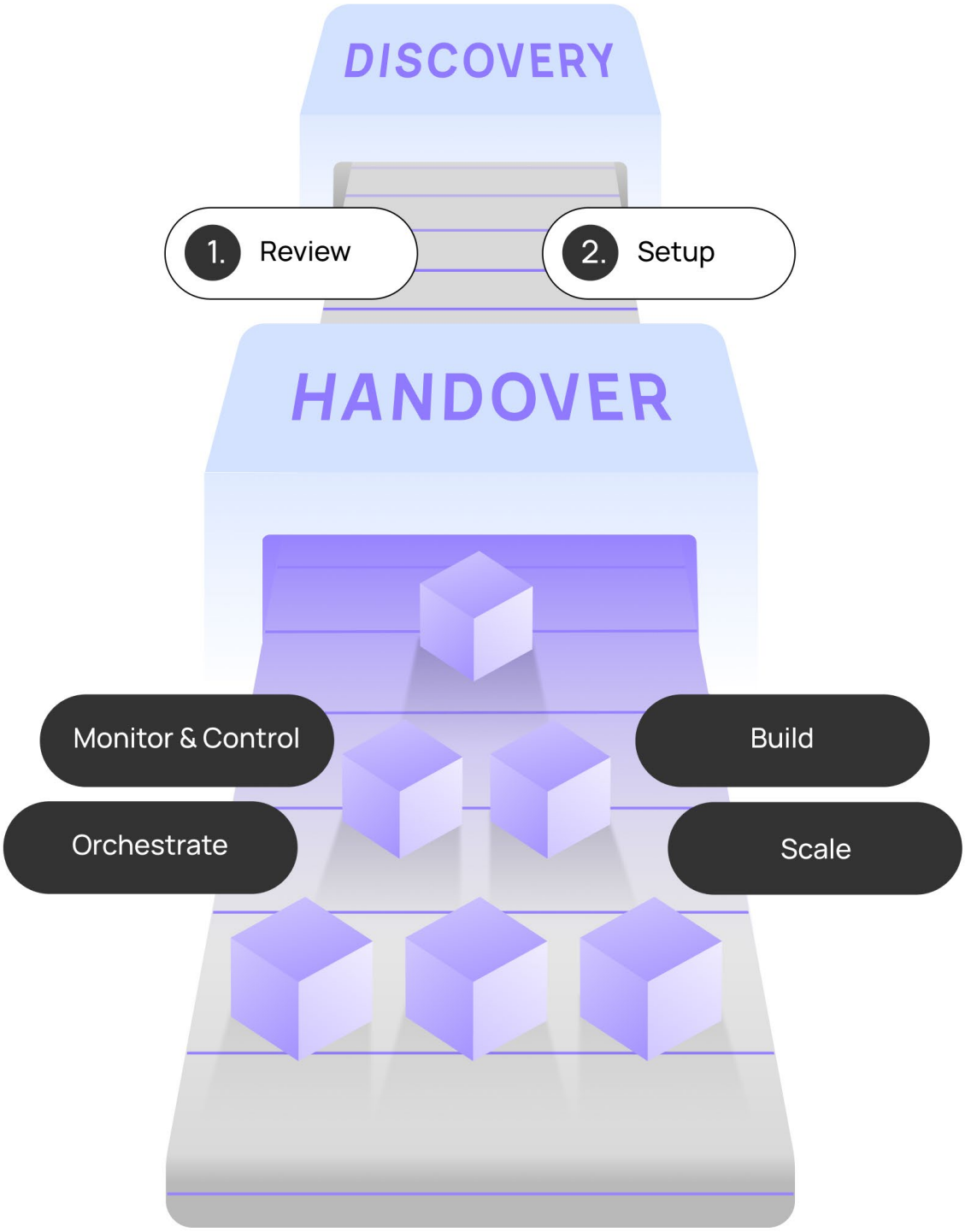
Structured logging, tracing, and usage metrics per app. Compute cost attribution is built in. Rate limits, query limits, and timeouts are enforced as guardrails so costs do not grow invisibly.

Scalable delivery model

Teams can ship many apps without multiplying governance overhead, architectural debt, or operational risk.

How does Apps Factory work?

Apps Factory runs in phases. First, we review what you have and set up the foundation your teams ship from. Then hand it to you, so you and your teams can build and run an entire ecosystem of apps—on your own and at scale.



What we build and what you own

Your operational foundation is production-ready in weeks. Let's break down how our engagement starts, what the Apps Factory setup consists of, how it's handed to your team, what it enables, and when you can start building independently.

Discovery Review Setup Handover Enablement

START – WEEK 0

We align on use cases

Discovery Workshop

HIFLY REVIEWS – WEEKS 1 – 2

We audit your environment

Architecture review

HIFLY BUILDS – WEEKS 3 – 5

The complete operational foundation

Every technical layer your ecosystem needs: standardized app skeletons, agentic engineering scaffolding, and unified governance and observability.

Your teams will never have to build from scratch again.

Architecture setup
Templates
Auto-provisioning
CI/CD setup
Security
Agentic engineering
Scaling
Testing
Unified logging
Observability
Cost control
Cost optimization
Production-readiness

HANDOVER TRANSITION

Live knowledge transfer

Handover Workshop

YOU BUILD

Your team is enabled to ship at will

AI apps
Data apps
Agentic workflows
Embedded automation

From review to enablement in 4 weeks

Review

Assessing your current setup

We run a Discovery Workshop and review to establish a precise baseline of your Databricks environment before anything is standardized or built.

- Audit across 7 dimensions: workspace topology, identity and auth, Unity Catalog governance, data and state architecture, AI and agentic components, CI/CD and observability, and cost attribution.
- Validate up to 3 business use cases and map them to the right Databricks Apps patterns.
- Identify gaps, risks, and quick wins across security, governance, and operational readiness.

Deliverables

- as-is architecture map
- maturity assessment across all 7 dimensions
- gap and risk register
- use case feasibility assessment with recommended implementation patterns
- target reference architecture
- prioritized action plan and quick wins

Setup

Building up the Apps Factory

We create the technical foundation that every subsequent app is built on: templates, standards, shared patterns, and technical governance.

- Deploy standardized app templates for analytics, chatbot, and Genie-powered applications.
- Establish shared technical patterns across all templates: secret management, structured logging, CI/CD, async jobs, MCP server, and inter-app communication.
- Hand over the full template library to your teams with an enablement workshop.

Deliverables

- production-ready app skeletons
- shared pattern library
- framework selection guidance
- local development and remote debugging setup
- OBO auth and RBAC patterns
- DABs templates
- observability and cost monitoring framework

Architecture review

Architecture review is a foundational step that defines whether your current Databricks environment can scale to thousands of apps and agentic workflows without breaking or creating technical debt.

Key points to review and define:

1. Platform readiness and workspace architecture

- Databricks workspace topology
- Current usage patterns
- App deployment approach
- Runtime and supported app patterns
- Compute provisioning strategy
- Semantic layer readiness

2. Identity, authentication, and least privilege

- OBO decision points
- Dedicated service principal model per app
- Group-based RBAC strategy
- Sharing model and standard roles across apps

3. Governance and data integrity with Unity Catalog

- Catalog/domain design, ownership model, and policy scaling strategy
- Data integrity control mechanisms
- Privilege model and audit requirements
- Row/column controls and where they are required
- External data access via UC Connections

4. Operational readiness (CI/CD, observability, FinOps)

- CI/CD maturity and environment promotion
- IaC approach
- UI framework choices
- Observability baseline
- Cost attribution

5. AI, agentic and Genie components

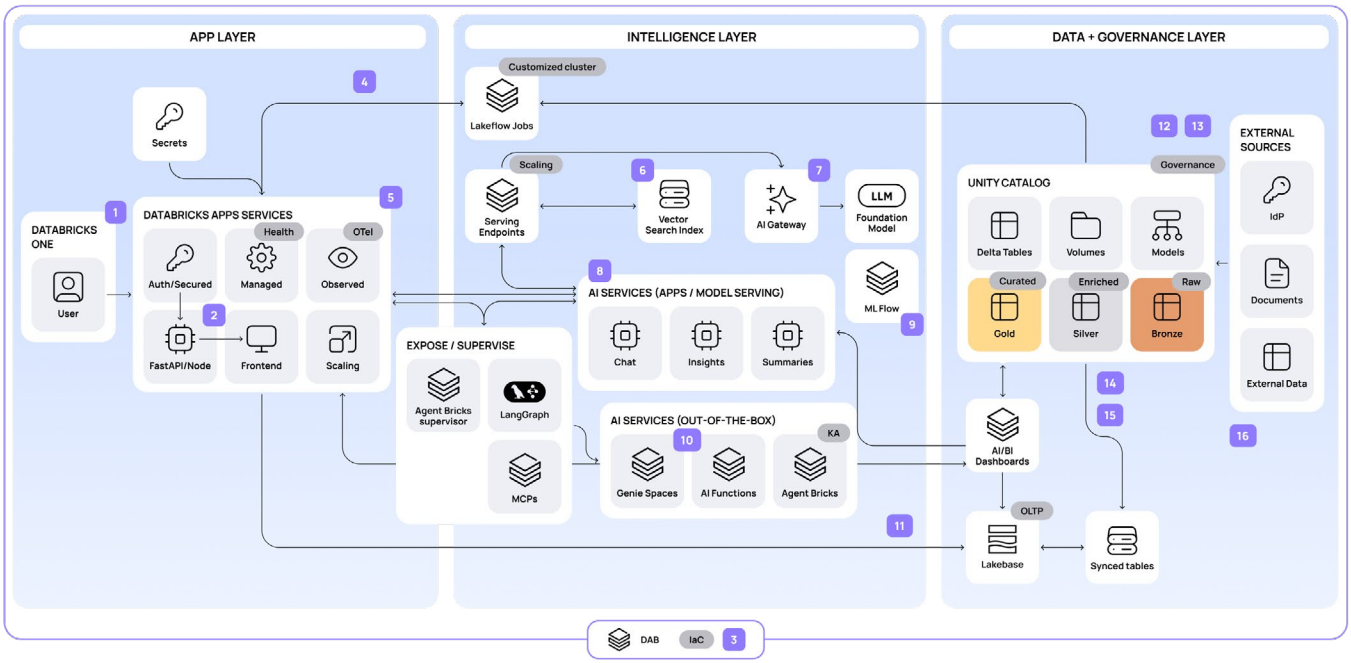
- Serving Endpoints
- Agent architecture
- RAG pipeline maturity
- Conversation history pattern
- Genie readiness
- Security guardrails

6. Data and state architecture (Lakehouse + Lakebase)

- Read access via governed tables/views
- Controlled write patterns, audit trails
- Delta table sync patterns to Lakebase
- Lakebase fit for transactional app state

Foundational architecture setup

The fewer integration points your architecture has, the fewer things can break, slow down, or create security gaps. A well-structured Databricks environment is organized into three independently scalable layers. Each is Databricks-native, each growing on its own terms without destabilizing the others.



1. Business-user entry: Apps, Genie, Dashboards + Search
2. apx/AppKit
3. CI/CD templates, enforced rules. Service provisioning.
4. Unified system to online and offline data ingestion, which involves: parsing, tagging, enrichment, embedding, conversion. Utilizes AI functions.
5. Managed OLTP collector: Zerobus UC ingest - otel_ logs, otel_spans, otel_metrics
6. Hybrid search: chunks + metadata. Reranking, Evals.
7. AI Gateway centralized model access:
   - auth + rate limits + cost
   - provider routing
   - logging/governance
8. Agent runtime contract, MLflow AgentServer + ResponsesAgent
9. Inference Tables + MLflow Tracing + Evals
10. Knowledge store, benchmarks, feedback, research
11. Low-latency reads, scale to 0, branch, PITR, compute-store separation
12. Policy enforcement: ABAC (tags), RLS/CLS, audit
13. System tables:
    - Audit logs (system.access.audit)
    - Query history (system.query.history)
    - Lineage (UC lineage + lineage system tables)
    - AI Gateway usage (system.ai_gateway.usage)
    - Inference tables (payload + traces)
    - Billing
14. Keep Bronze immutable for replay. Build Silver for quality. Gold for consumption + eval sets.
15. Unity Catalog Metric Views (+ semantic metadata/ synonyms for humans and LLMs) + materialization for performance
16. Databricks/Partner Connectors, Zerobus, Lakehouse federation, Delta Sharing

Apps templates and patterns

Templates, shared patterns, and technical governance give your teams everything they need to ship without starting from scratch. Components and blueprints for every build, from a lightweight app to a complex multi-workflow enterprise system. Every new app automatically inherits the right foundation.

REPEATABLE DEVELOPMENT | REDUCED TIME TO PRODUCTION | CUSTOMIZABLE FEATURES, BRANDED UI

App templates (GitHub repos):

Dashboard and analytics apps

- Delta table sync
- Lakebase synced tables
- Async DB connection patterns
- Row-level security
- Dashboards and visualizations

Container app with custom chat

- Router for Genie API
- Chat UI
- Inline LangGraph agent/Serving Endpoint integration
- LLM streaming
- Chat history persistence, citation, and user rating/feedback
- Conversation guardrails and source attribution

Container app with Genie integration

- Genie Space embedding/custom Genie API frontend
- Full Genie response handling
- Trusted query display, chart editing, data download
- Rating/feedback loop, send-to-review flow
- Deep research

Shared technical patterns (for all templates)

Project structure, dependency management, secret management, local development, background services, MCP server, and apps-to-apps communication.

UI framework and decision guidance

Decision matrix to choose between Streamlit/Gradio, full-stack React (with Node.js or Python/FastAPI backend), or hybrid approach depending on your use case.

Auto-provisioning

Apps auto-provisioning

Auto-provisioning removes the usual delays and bottlenecks from app creation. Any team with a validated use case can go from an idea to a deployed, governed application without waiting on IT. A single app descriptor defines everything: identity, storage, workspace targets, and deployment configuration. From there, Azure services and DABs handle the rest automatically.

CI/CD

Unified CI/CD setup

CI/CD configurations and quality gates make sure every deployment in your ecosystem follows the same standardized, automated path from development to production.

Apps Factory standardizes deployment through DABs, GitHub Actions-based CI/CD pipelines, and environment promotion, with Lakebase branching built in where needed.

CI/CD pipeline and deployment with Apps Factory:

- **DABs:** bundle-based deployment as the standard
- **Environment promotion:** Dev → Stage → Prod with GitHub integration
- **Pipeline templates for all resource types**
- **Migrations:** automated DB migrations as part of CI/CD and Lakebase strategy
- **Quality gates:** automated tests running from CI/CD pipelines
- **IaC:** parameterized resource deployment

Each deployment is controlled across three independent axes. So every environment is fully isolated and promotable without manual coordination.

PIPELINE AUTOMATION | CONSISTENT DEPLOYMENTS | NO MANUAL CONFIGURATIONS

AXIS	PRIMITIVE	CONTROLS	EXAMPLE VALUES
Code	Git branch	what source ships	feature/x · dev · stg · prod
Deployment	DAB target	where & how it lands	user · dev · stg · prod
Data	Lakebase branch	which DB you read/write	dev-<user> · pr-<slug> production

Lakebase branching

Lakebase branching gives every developer or feature branch its own isolated database copy. So teams can develop and test against real data without touching shared environments.

Environment promotion approach

- Lakebase branching for isolated, per-branch database state
- Standard migration tooling for environment promotion across separate workspaces

Security

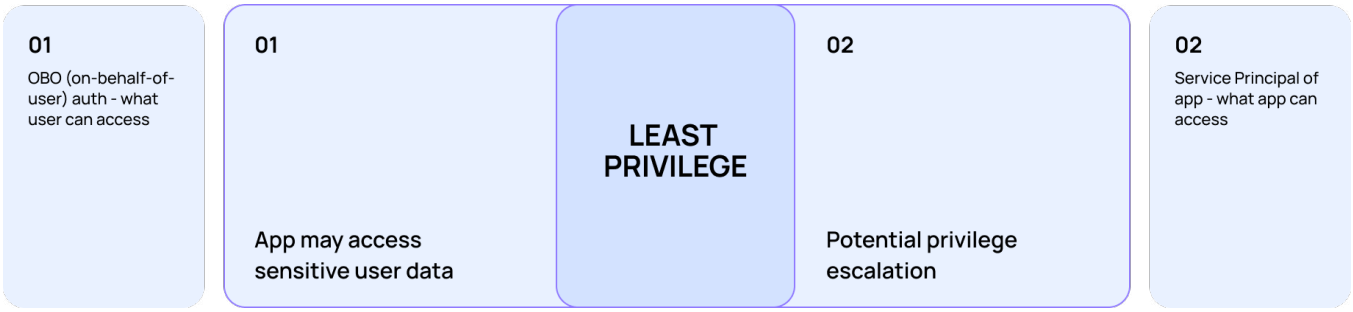
Security and access control

Standardized OBO auth, RBAC, and service principal patterns create one security and access framework applicable at every layer and enforced as repeatable policy-as-code.

Security and access control (documentation and implemented configuration):

- **OBO (On-Behalf-Of) authentication:** users interact with data through their own identity—no escalation.
- **Service Principal (SP) governance:** documented and approved SP usage per app, least-privilege principle.
- **RBAC (Role-Based Access Control):** group-based roles (CAN USE / CAN MANAGE), no direct user grants.
- **RLS (Row-Level Security):** user-based RLS in Lakebase, Unity Catalog enforced for all data access.
- **Write patterns:** validation layer → async commit job or controlled API → full audit trail, plus the ability for multi-table transaction management.
- **AI-specific security:** approved model list, tool usage policy, rate limits, prompt injection mitigation.
- **Custom access handling:** process-, workspace-, and team-based permission model that maps governance to actual organizational structures.

RELIABLE ACCESS CONTROL | INHERITED GOVERNANCE | AI RISK MITIGATION



Guardrails and best practices for AI security and governance against: data leaks, jailbreaks, model collapse, hallucinations, and more...

Read→

Agentic engineering setup and tooling

Databricks Apps cut development time **from months to weeks**. Agentic engineering with Databricks-aware toolkits cut development time **from weeks to hours.**

Databricks-native tools and scaffolders give your coding agents the templates, knowledge, and execution power grounded in your actual Lakehouse environment—schemas, metadata, logs, deployment states, etc. So your teams can actually trust and rely on the output.

Apps Factory includes a complete agentic engineering setup from context to guardrails, starter repos for low-code, full-stack apps, Gen UI features, and a tooling decision matrix for engineers to know what to use, when, and how.

Databricks Agent Skills
Official skills library for teams coding outside the workspace who need robust, guardrail-enforced Databricks guidance loaded into their assistant

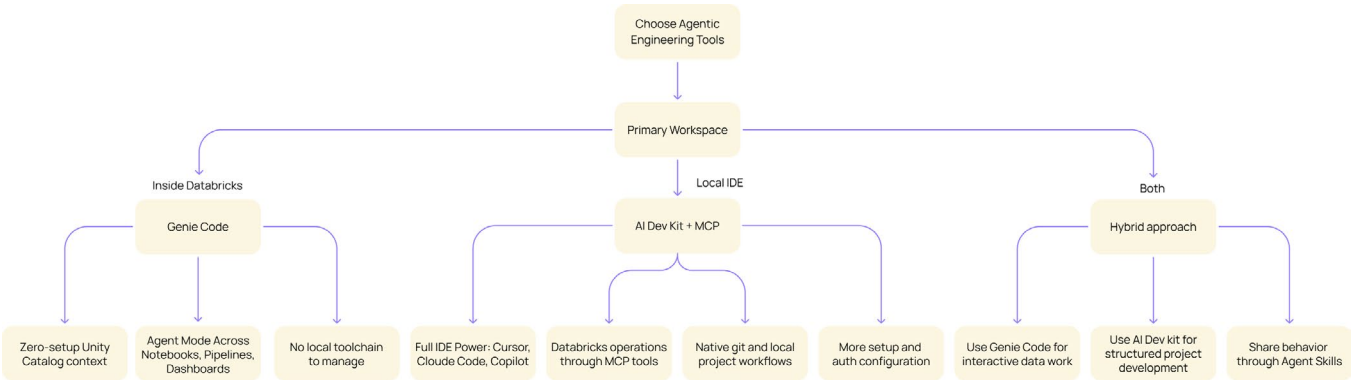
AppKit
Official Node.js + React SDK with pre-built plugins for full-stack engineers building modular, production-ready apps

Genie Code
Workspace-native AI agent for data engineers and analysts doing data-native work directly inside Databricks

AI Dev Kit
26 skills + 50 MCP tools for full-stack apps and multifaceted Databricks projects across the ecosystem

apx
Rust-based app scaffolder for rapidly generating complex, highly customized React + FastAPI applications

Agent Bricks
Official product for building production-ready agents for data teams who need governed, deployed agents fast



Proprietary scaling framework

In most cases, apps should be architected to stay thin. Databricks provides powerful services to offload compute-heavy work to other layers: serving endpoints, jobs, and clusters. When scaling the app layer is genuinely needed, Hiflylabs' proprietary framework provides approaches beyond current Databricks' capabilities—engineered, tested, and measured in production.

Vertical scaling

- Natively available
- Minimal instance: 70-100 QPS for a complex AI task
- **Use-cases:** Stateful apps w/ session affinity and zero-downtime deployments

Horizontal scaling

- Multiply performance by adding instances manually
- Avg. 30ms extra latency
- **Use-cases:** Long-running app functions, serving specific use cases like ingestion

Native horizontal scaling is coming soon; cross-app scaling is a Hiflylabs-engineered approach until official support is released.

How to offload work

Serving endpoints

- Unlimited endpoints, scaling automatically
- Avg. 20ms extra latency
- **Use-cases:** Large number of simultaneous (AI) operations

Lakeflow Jobs

- 10-15 second startup latency
- Customizable clusters for heavy-duty jobs
- **Use-cases**: Ingestion, document parsing, embedding, conversion, image manipulation

Testing

Unified testing

Apps Factory provides a structured, automated testing stack that covers every layer of your Databricks App, from business logic to UI behavior to production load.

Tests run automatically on every pull request. Expensive checks run before release, cheap ones run every PR.

Unified testing with Apps Factory covers:

- **Unit:** business logic, routes, and helpers.
- **Workspace unit:** fast pytest iteration directly inside Databricks.
- **Data quality:** data contracts on pipeline datasets before users see the data.
- **API smoke:** auth, app boot, and backend route validation.
- **E2E/regression:** full UI and API user path verification.
- **Performance:** sustained load, latency benchmarks, and failure thresholds.

Powered by:

- **Playwright + DAB:** deploys the app, runs UI and API regression as a Databricks job, stores artifacts in Unity Catalog.
- **Locust:** simulates concurrent users, measures p50/p95/p99 latency, generates HTML evidence reports.
- **GitHub Actions + DABs:** CI pipeline that validates, deploys, and runs tests on every PR automatically.

Layer	Tool	What it proves	App-related?
Unit	`pytest` , `vitest/jest`	Business logic, routes, helpers	☑ app code
Workspace unit	Databricks Python unit testing UI	Fast pytest iteration in workspace	workspace-adjacent
Data quality	Lakeflow, Genie code	App data is valid before users see it	data-adjacent
API smoke	Databricks App `/api/`	Auth, app boot, backend route works	☑ app runtime
E2E/regression	Playwright + DAB job	UI/API user path still works	☑ app behavior
Performance	Locust	Sustained users, latency, failures	☑ app capacity

Logging

Unified logging and governance

One Databricks-native admin view that answers across every app: **Who used it? What happened? What failed? Where was latency? What did it cost?** Achieve complete traceability from user request to agent reasoning to billing—observable and queryable from a single telemetry plane.

Every producer feeds the same telemetry plane

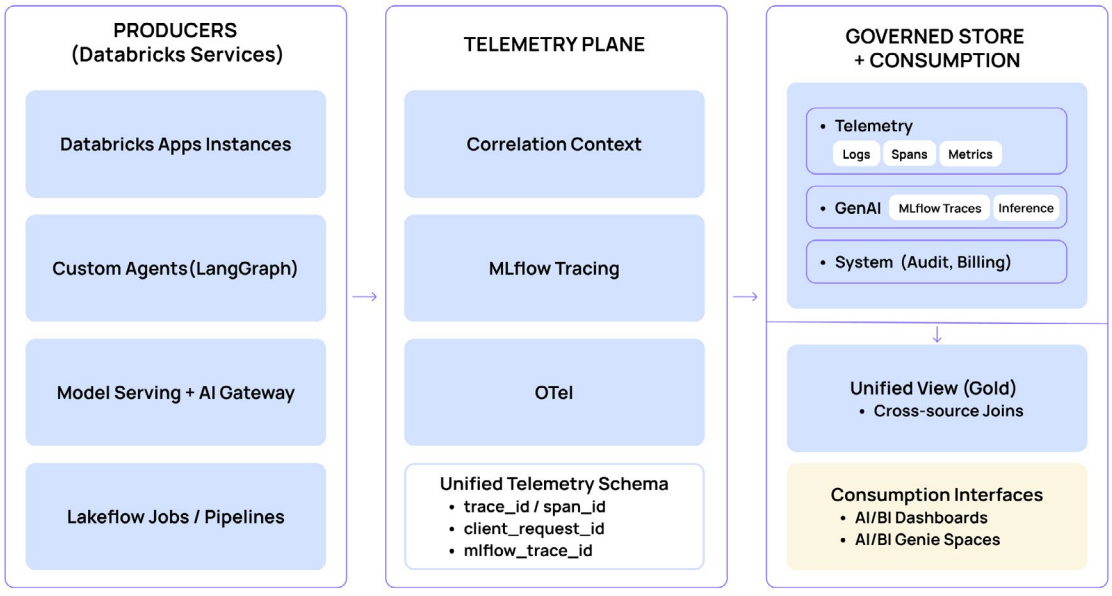
- **Databricks Apps:** OTel sidecars ship logs and metrics directly to Unity Catalog tables.
- **Custom Agents:** MLflow Tracing captures agent reasoning, tool calls, and LLM spans.
- **Model Serving + Gateway:** Unity AI Gateway tracks token usage, latency, and request tags.
- **System tables** monitor batch execution, evaluation runs, and pipeline health.

Everything flows through a unified telemetry layer

client_request_id, session_id, and trace_id go across every service for seamless joins.

All of it lands in one governed, queryable store

- **Unity Catalog Observability:** Unified tables for OTel logs, spans, and metrics joined with system audit data.
- **Unified Trace View:** MLflow trace waterfalls materialized via SQL views for rapid debugging and cost attribution.
- **Consumption Interfaces:** AI/BI Dashboards for SLOs and Genie Q&A grounded on curated silver/gold log views.



Observability

360° operational view: See every app, user, and event

A single operational command center for every agentic workflow, every app, and every cost with full traceability and auditing across the entire ecosystem.

What you can see across every app

Every component exposes unified, OpenTelemetry-compatible metrics. So health, usage, and performance are visible from one place.

Observed metrics:

- **Service health:** Heartbeat metrics and uptime status per app.
- **Performance (OTel):** p95/p99 latency, HTTP errors, and stack traces.
- **Usage stats:** DAU, session journeys, and version adoption.
- **Access control:** Audit trails for logins and permission changes.
- **Cost allocation:** AI spend split by agent, team, and cost center.
- **Agent quality:** Relevance, faithfulness, and toxicity eval scores.
- **Data health:** Freshness, null rates, and expectation failures.
- **Security:** Nightly dependency audits and vulnerability scans.

Alerting thresholds:

- **Availability drops:** Uptime < 99% triggers immediate on-call pages.
- **Latency spikes:** p95 shifts >2x over 7-day medians alert app owners.
- **Cost anomalies:** Daily spend exceeding 2σ triggers FinOps alerts.
- **Quality regressions:** Significant drops in model evaluation scores.
- **Data failures:** Freshness or DQ check failures in upstream pipelines.
- **Access breaches:** Suspected leakage or unauthorized permission changes.

Cost control

Cost monitoring

Control AI spend, trace every cent

Use governed tags to enforce cost-center visibility across every app, model call, and compute resource. So you can track all spend and prevent cost leaks before they happen.

Monitorable components

Apps

- Trackable by name or ID
- Costs are shared across all usage

Serving Endpoints

- Trackable by name or ID
- AI gateway token count can be tracked on a per user level if OBO is enabled
- External service health

Compute

- Trackable by name, ID or tags
- Costs are shared
- Monitor uptime, utilization, health and more

Vector Search

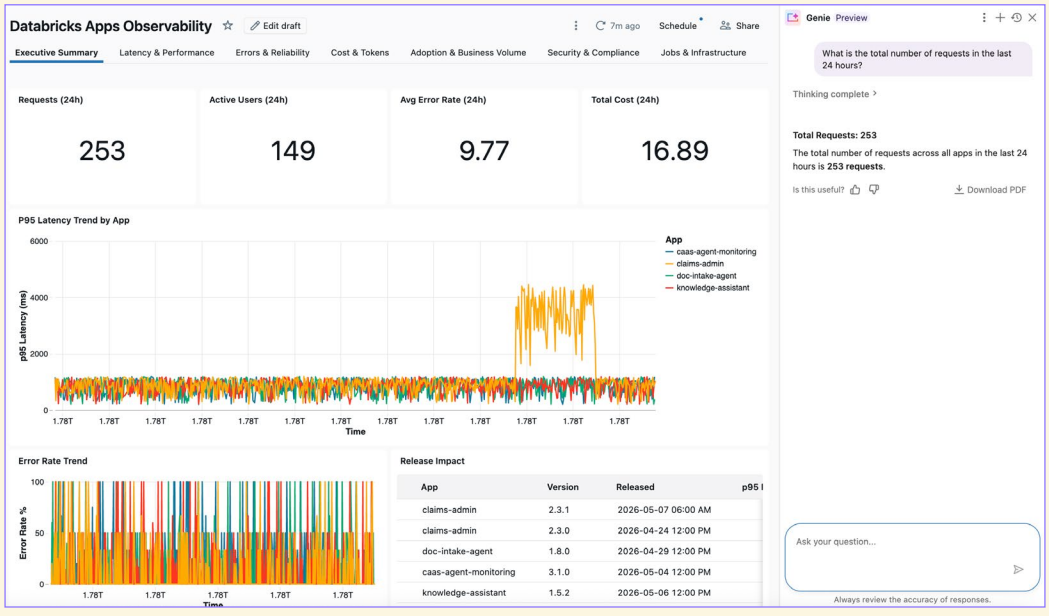
- Trackable by name or ID
- Costs are shared

Lakebase

- Trackable by name or ID
- Costs are shared across all usage

Workflows

- Trackable by name, ID or tags
- Cost sharing depends on usage
- Monitor pipeline health and more



Cost optimization

Cost optimization

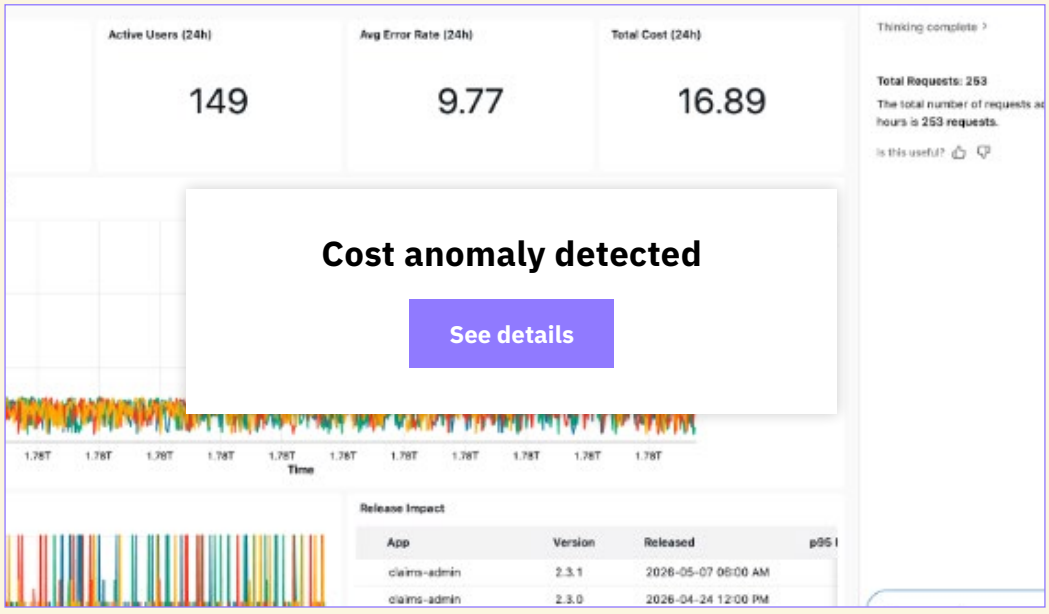
Drill down costs, spot saving opportunities, optimize spend

Apps Factory includes a custom AI/BI dashboard combined with a dedicated cost analyst agent—a Genie-powered application grounded in your Databricks data that answers cost questions on demand.

- Identify which teams, endpoints, or agents are driving spend and rank them by dollar impact so you know where to act first.
- Detect dormant endpoints billing with zero requests and flag them as immediate shutdown candidates.
- Compare model costs side by side and calculate projected savings from routing traffic to a more efficient model.
- Set up automated overspend guardrails to block cost overruns.
- Surface untagged workloads that make chargeback impossible and generate a prioritized fix list ranked by spend.

Instead of a number on a dashboard, you get an answer with an action plan in seconds.
Ask Genie: “Which apps grew AI cost >50% week-over-week?” and get the answer in seconds.

One room for every cost conversation: all apps, agents, and compute monitored in a single space.



Production-readiness checklists

One quality gate for 100% production-ready apps

The production readiness checklist is the quality gate between built and deployed. Nothing ships without passing this. It makes sure every app meets the same enterprise-grade standard, no matter who built it and what templates they used.

Production readiness + SDLC + Software architecture checklist for data and AI apps

Apps layer:

- ☑ Auth and SSO
- ☑ App access to analytical data
- ☑ Observability and monitoring
- ☑ Deployment and CI/CD environment setup
- ☑ Backup and recovery
- ☑ SLA and compliance

AI layer:

- ☑ Build (Agent Bricks functions, agents, MCPs)
- ☑ Retrieve (Vector Search)
- ☑ Serve (Model Serving + Gateway)
- ☑ Trace & evaluate (MLflow)

Data layer:

- ☑ Metadata (business/semantic description, lineage, relations, value constraints, ownership)
- ☑ Code-defined structure and migrations (incl. historical data content)
- ☑ Deployment and CI/CD environment setup
- ☑ Access control (dev / stage / prod, plus row-level security)

Non-transactional Data:

- ☑ Data Quality checks
- ☑ Load-time/Generation-time Data Quality checks
- ☑ Pipeline monitoring (operations, failures, and performance) and notification/alerting
- ☑ Auditability tracing (for ingestion and transformations), including history

Software development lifecycle:

- ☑ Global features
- ☑ Data and AI integration
- ☑ Auth and security
- ☑ Databases
- ☑ Documentation
- ☑ Source code version control
- ☑ Deployment
- ☑ UI
- ☑ SLA support and maintenance
- ☑ Scale and user
- ☑ Monitoring, testing, logging, and error handling
- ☑ Other non-functional requirements
- ☑ Traffic
- ☑ Cost and budget
- ☑ Backup and recovery

Handover

Handover workshop: own it, run it, scale it

Apps Factory equips your teams with everything they need to build, scale, and operate hundreds of agents and apps on Databricks. We equip you with everything you need for easy enablement and smooth adoption.

Handover workshop: Live walkthrough across all deliverables with adoption playbook, knowledge transfer, and Q&A

- **Reference architecture diagram:** visual overview of the entire Databricks ecosystem your organization now runs on.
- **Framework selection guide:** decision tree for when to use Streamlit/Gradio vs React/FastAPI vs full Node-based scope.
- **Production readiness checklist:** gate criteria every app must meet before promotion to production.
- **Agentic engineering guide:** setup and tools to accelerate app development inside your environment.
- **Developer onboarding guide step-by-step:** clone template → configure → deploy → monitor.
- **Supplier requirements document:** standards for external development partners covering code quality, testing, and CI/CD compliance.
- **Governance and policy-as-code: l**ifecycle management rules codified and enforceable across every team and supplier.

Now you have the standards, the tooling, and the knowledge to build real agentic automation at scale—secure and governed by design.

You can spin up apps and automation workflows in days. Cut 90% of your team's menial, repeatable work. So they can focus on the remaining 10% where core decisions happen: complex judgment calls, reactive problem-solving, and client relationships.

The Apps Factory is set. Start building.

Apps range

What can you build with Apps Factory?

Apps Factory gives your teams the foundation to build standalone applications, fully automated AI workflows, or a combination of both.

Data apps

Dashboards, workflows, and self-service analytics built on top of your warehouse data. So your business teams query and act on live data directly, without an analyst in the loop.

AI apps

Chat interfaces and interactive applications that deliver AI outputs securely within the same governed Databricks ecosystem—guardrails and auth already in place.

AI agent workflows

Automated pipelines where AI agents handle multi-step processes end-to-end, with humans stepping in only where it truly matters.

AI automation embedded in apps

The best of both: intelligent automation running inside a user-facing application, keeping humans in the loop in more natural, intuitive user experiences.

Apps tiers

How far can you take it?

From a focused single use case to a company-wide ecosystem of apps and agents: Apps Factory supports any scale of ambition.

Tier 1

Single use case

One app or automation targeting a specific workflow or business need. The fastest way to get value on the ground.

~~3 months~~ → 4 weeks

Save 40 dev days

Tier 2

Multi use case

A flagship app or agentic system covering multiple related workflows, functions, or teams under one roof.

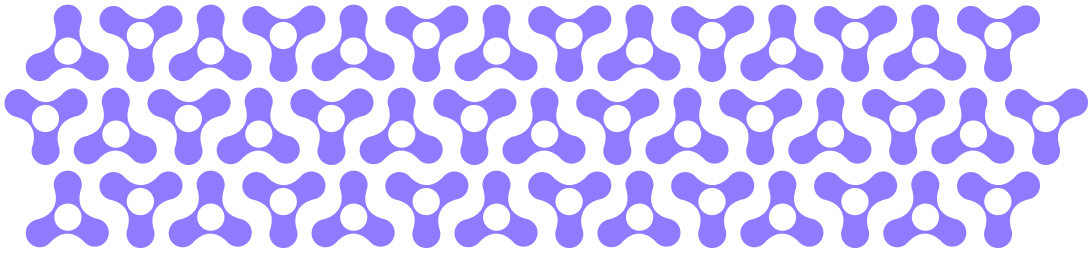
~~6 months~~ → 3 months

Save 65 dev days

Tier 3

Ecosystem

The full realization of an AI-native enterprise: A fleet of apps and agents spanning departments and functions across the organization.



4-7 weeks to set up

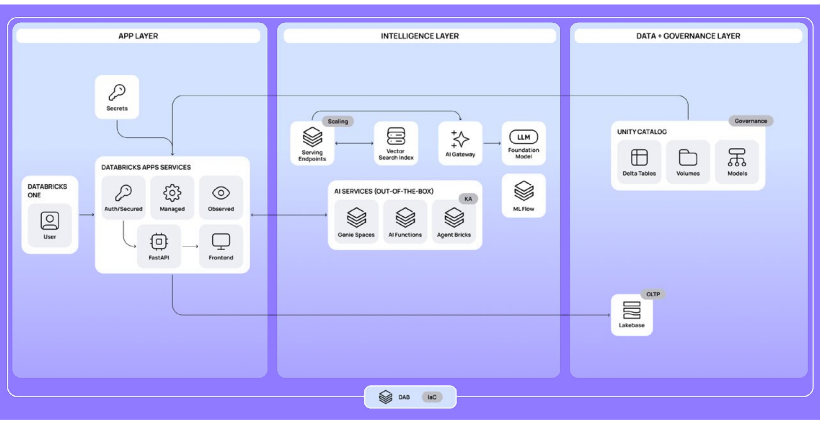
Save 100 dev days per app

Examples

Boilerplate examples from real-world use cases

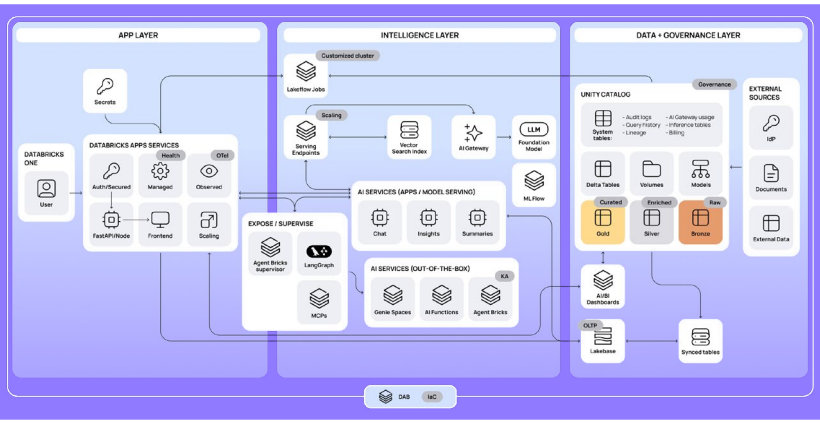
Single use case

High-performing agent-native app to automate targeted workflows and data-intensive tasks.



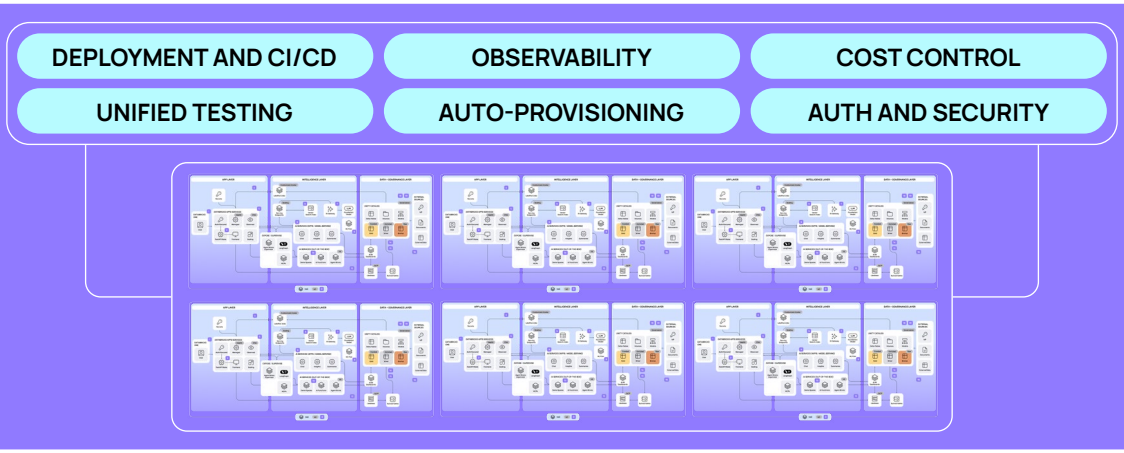
Multi use case

Flagship full-stack application with low-latency access, rich UI, and custom workflows.



Ecosystem

Enterprise ecosystem of 100+ apps with a strong data foundation and advanced AI.



Healthcare organization spins up enterprise-wide AI automation with Apps Factory

Hero

Integrated healthcare system including 7 hospitals and over 12.3M annual outpatient visits. The organization performs diverse functions across clinical care, healthcare research and education, running complex corporate operations.

Operational challenge

Despite a mature centralized digital backbone and shared IT governance, the organization lacks a standardized framework to rapidly automate high-effort manual processes and gain operational efficiency at scale.

Approach

The client teamed up with Hiflylabs to define high-potential automation use cases across workflows and departments and set up templates, tooling, and an operational layer to build and deploy applications and AI automation at scale.

Result

4 flagship use cases

40% doctor time saved on admin

Selected use cases include

- **Patient 360:** Integrated patient data and journey, surfacing decision support, medical data summaries with full traceability, and history 24/7 without manual lookup.
- **Automated cohort finder:** Processes records at 90%+ accuracy and reduces cohort build time from weeks to minutes. So research teams get to insights faster without data team bottlenecks.
- **Legal and contract risk management:** Cuts vetting time, flags risky clauses automatically, and gives full visibility into vendor and doctor contracts.
- **Supply chain management:** Predicts shortages, analyzes physician preferences, and monitors vendor contracts, eliminating last-minute procurement and supply gaps.

Set up your own Apps Factory

Your second-in-command to build and run a fleet of agent-native apps

Start with a Discovery Workshop to see how Apps Factory can accelerate development, set up fail-proof operations, and kick off safe and controlled agentic automation at scale in your organization.

Book your Discovery Workshop

No strings attached. No obligations to move on if it's not what you're looking for.

The team behind Apps Factory

Created by Databricks Apps pioneers

About the author

Domonkos Pál is Head of Digital Products at Hiflylabs, recognized for architectural thinking, fast tech adoption, and production-grade delivery standards on the Databricks platform.

- World's first Databricks Apps MVP (2025–2026)
- Member of the Databricks Product Advisory Board
- Works directly with the Databricks Apps and Lakebase product teams
- Speaker at DAIS 2025–2026 and leading data and AI events across Europe and the US

About Hiflylabs

Hiflylabs is a certified Databricks delivery partner that helps global enterprises turn data into decisions, AI into ROI, and pilots into production since 2015.

- 250+ data and AI experts in-house across offices in the US, UK, DACH, and Hungary
- 2 in-house Databricks MVPs, 50+ Databricks certifications
- Developer of the largest Databricks Apps projects running live at Fortune 500 scale
- End-to-end FDE partner: Forward Deployed Engineers with production experience

Author: Domonkos Pál
Contributors: András Zimmer, Gergely Kocsis, Zsombor Földesi, Péter Varga
Editing: Márk Lőrinczy, Alexandra Bondina
Design: Anna Füzes, Kristóf Turzó

We make AI, data, and digital products work

Domonkos Pál

Databricks Apps MVP

DOMONKOS.PAL@HIFLYLABS.COM

LINKEDIN

Hiflylabs

Databricks FDE partner

HELLO@HIFLYLABS.COM

LINKEDIN