Overview (Single Source of Truth)

This document is the authoritative specification for this repo/product. If anything conflicts with other docs, **this doc wins**.

Product Goal

Build a **simple, on-prem / hybrid "junior Databricks"**:

- **Control plane**: manage workspaces, connections, catalogs, jobs, runs, users (simple in dev), audit basics, and an admin UI/API.
- **Data plane**: run open-source engines on Kubernetes to ingest, store, transform, and serve analytics/ML data.

Target domains include utilities (water/electric), banks, and government. We stay domain-agnostic: domain specifics live in connectors and data models, not the core platform.

Core Principles

- **Keep it simple**: prefer fewer components, boring tech, and clear boundaries.
- **Kubernetes-first**: one deployment story; local dev uses lightweight equivalents.
- **Iceberg-first**: open table format for multi-engine interoperability.
- **Trino for SQL/BI**: interactive SQL is first-class.
- **Batch-first** initially: streaming comes after a stable control plane + data plane foundation.

Default Stack (v1)

Data Plane (v1)

- **Object storage**: MinIO (S3-compatible)
- **Table format**: Apache Iceberg
- **Catalog**: Iceberg REST catalog
- **SQL engine**: Trino (queries Iceberg tables)
- **Batch compute**: Spark on Kubernetes (ETL/ML feature jobs)
- **Orchestration/execution**: start with a simple control-plane worker submitting jobs; integrate a mature workflow engine later

Control Plane (v1)

- **Backend**: Python + FastAPI
- **DB**: Postgres (platform metadata: workspaces, connections, jobs, runs, audit)
- **Frontend**: Admin GUI (simple) — can be a small React/Next.js app or even server-rendered pages early if speed matters
- **Auth (dev phase)**: simple practical auth module (local users, hashed passwords, roles)
- **Auth (prod target)**: OIDC (e.g., Keycloak/AzureAD/Okta) — planned, not required for Phase 1

What We Are Building (Definition)

Control Plane Responsibilities

- **Workspace/projects**
  - Single-org model with multiple workspaces
- **Connections**
  - Store and validate connection configs (MinIO, Trino, Spark, Postgres, etc.)
  - Secrets handled safely (k8s secrets in v1; external secret manager later)
- **Catalog registration**
  - Register datasets/tables (logical name, owner, location, format=Iceberg, tags)
- **Jobs**
  - Job types: Trino SQL job, Spark batch job
  - Versioned job definitions, parameters, and schedules (optional in v1)
- **Runs**
  - Submit, monitor, stop, retry
  - Persist run state + logs links
- **Admin GUI**
  - Workspaces, connections, jobs, runs, basic health view
- **Audit (baseline)**
  - "who did what and when" for key actions

Data Plane Responsibilities

- Provide reliable, scalable execution for:
  - Batch ingest + transforms (Spark)
  - Interactive SQL for BI (Trino)
  - Durable ACID tables (Iceberg on MinIO)

Non-Goals (v1)

- No custom compute engine (we orchestrate existing engines).
- No full governance suite (fine-grained masking/lineage) in Phase 1.
- No streaming pipeline requirements in Phase 1 (batch only).

Phased Build Plan (High-Level)

Phase 0 — Repo + Local Dev Skeleton

- FastAPI service scaffold + Postgres schema migrations
- Minimal Admin UI scaffold
- Local dev environment (docker-compose or kind/minikube)
- "Hello run": submit a trivial Trino query and record a run

Phase 1 — Minimal Kubernetes Foundation (Data Plane Baseline)

- Kubernetes baseline manifests/Helm chart structure
- MinIO deployed and reachable
- Iceberg REST catalog deployed
- Trino deployed and able to query a sample Iceberg table
- Spark on k8s able to write an Iceberg table to MinIO

- Basic observability hooks (health endpoints; minimal metrics later)

Exit criteria: We can ingest sample batch data -> write Iceberg -> query in Trino -> show results in UI.

Phase 2 — Minimal Control Plane MVP

- Workspaces + connections + job definitions + runs (persisted)
- Admin GUI: manage connections/jobs/runs
- Simple auth module (dev practical)
- Role-based checks for admin vs user actions

Exit criteria: From UI/API: create connection -> create job -> run -> see status/logs -> rerun.

Phase 3 — Hardening

- OIDC integration (Keycloak or customer IdP)
- Backups + retention policies (MinIO + Postgres)
- Resource isolation / quotas (prevent BI from starving batch)
- Operational docs for install/upgrade

Phase 4 — Streaming (After Foundation is Solid)

- Introduce Kafka (or equivalent) only when requirements justify it
- Streaming ingestion to Iceberg with clear late-data + dedupe strategy
- Control-plane support for streaming jobs and monitoring

Design Constraints and Portability

This product must run in different environments:

- Minimal assumptions: Kubernetes cluster + storage + network + basic DNS/TLS.
- Cloud/hybrid is allowed, but the platform must work on-prem without vendor lock-in.
- Components should be replaceable behind interfaces (object store, catalog, auth provider).

Repo Structure

```
control_plane/
  api/          # FastAPI application
  worker/       # Async execution/submission workers
  db/           # Database migrations and models
admin_ui/       # Admin GUI (to be added)
infra/
  helm/         # Helm charts for data plane components
  k8s/          # Plain manifests if needed
  examples/     # Sample configs and demos
docs/
  overview.md   # This file
  runbooks.md   # Operational runbooks (to be added)
  architecture.md # Architecture diagrams (to be added)
```

Glossary

- **Workspace**: A logical grouping of resources (connections, jobs, datasets) within a single organization.
- **Connection**: A configuration that defines how to connect to an external service (MinIO, Trino, Spark, Postgres, etc.), including credentials stored securely.
- **Job**: A reusable definition of work to be executed (e.g., a Trino SQL query, a Spark batch job). Jobs can be parameterized and scheduled.
- **Run**: A single execution instance of a job. Tracks state (pending, running, succeeded, failed), logs, and results.
- **Catalog**: The metadata service that tracks Iceberg tables (their schemas, partitions, snapshots). In v1, we use Iceberg REST catalog.
- **Iceberg REST catalog**: A REST API-based catalog implementation for Iceberg tables, allowing multiple engines (Trino, Spark) to discover and manage tables.
- **Data plane**: The infrastructure layer that executes data operations (storage, compute, query engines).
- **Control plane**: The management layer that orchestrates and tracks data plane operations (workspaces, connections, jobs, runs).

How to Run Locally (Phase 0/1)

Prerequisites

- Kubernetes cluster (local: `kind`, `minikube`, or `k3d`)
- `kubectl` configured
- `helm` v3.x installed
- `docker` (for building images)

Quick Start (Data Plane Only)

```
# 1. Deploy data plane components via Helm
cd infra/helm
helm install dataplane ./dataplane

# 2. Verify components are running
kubectl get pods -n default

# 3. Run demo: Spark writes Iceberg -> Trino reads
cd ../../infra/examples
./demo-spark-to-trino.sh
```

Local Development (Control Plane)

```
# 1. Start Postgres locally (or use k8s)
docker run -d -p 5432:5432 -e POSTGRES_PASSWORD=dev postgres:15

# 2. Run migrations
cd control_plane
alembic upgrade head
```

```
# 3. Start FastAPI dev server
uvicorn api.main:app --reload

# 4. Test API
curl http://localhost:8000/health
```

Open Decisions (Tracked Here)

- ✅ **Iceberg catalog**: REST catalog (decided)
- ✅ **Deployment**: Helm charts (decided)
- ⏳ **Workflow engine**: To be decided in Phase 3+ (Temporal vs Airflow/Argo integration)
- ⏳ **UI framework**: To be decided in Phase 2 (React/Next.js vs minimal server-rendered admin)