WHITE PAPER

Aizen Foresight

Operational AI at Production Scale

Full stack AI platform to build, deploy, and scale AI solutions on real time data.

Executive Summary

Enterprises generate enormous volumes of real-time data — from IoT sensors and transaction streams to application events and customer interactions. Converting that raw data into accurate, low-latency predictions has historically required months of engineering effort: building bespoke pipelines, hand-tuning feature logic, managing training infrastructure, and maintaining model serving endpoints. By the time a model reaches production, the business problem has often moved on.

Aizen Foresight eliminates this gap. It is a no-code, end-to-end Operational Machine Learning platform that takes enterprises from raw data stream to live prediction endpoint on a unified, visual canvas — without writing a single pipeline script.

Key Platform Outcomes

Reduce ML deployment timelines from months to days with automated pipeline orchestration.

Serve sub-millisecond feature lookups via an in-memory Feature Store built on Apache Arrow.

Train models on any cloud GPU cluster with one click — no infrastructure setup required.

Deploy and monitor ML models in production via REST API. Maintain full data governance and auditability with a built-in Metadata Store and lineage tracking.

This white paper describes the architecture, components, and end-to-end workflow of Aizen Foresight — from data ingestion through feature engineering, AutoML, and real-time model serving — and walks through how a complete ML pipeline is assembled on the platform.

1. The Operational ML Imperative

According to Gartner, context data in business encompasses any real-time environmental signal relevant to a decision — live road conditions, current inventory levels, streaming sensor readings, or a customer's behavior in the last 30 seconds. Acting on this context at machine speed is what analysts now call Operational ML: the use of machine learning not as an offline analytics exercise, but as a continuous, production-grade decision engine embedded in business workflows.

The gap between ML research and ML operations remains wide. Industry surveys consistently show that fewer than 20% of ML models built in enterprise settings ever reach production. The failures are not algorithmic — they are infrastructural: fragmented tooling, brittle data pipelines, no feature reuse, manual deployment processes, and the absence of real-time serving infrastructure.

1.1 Why Traditional Approaches Fall Short

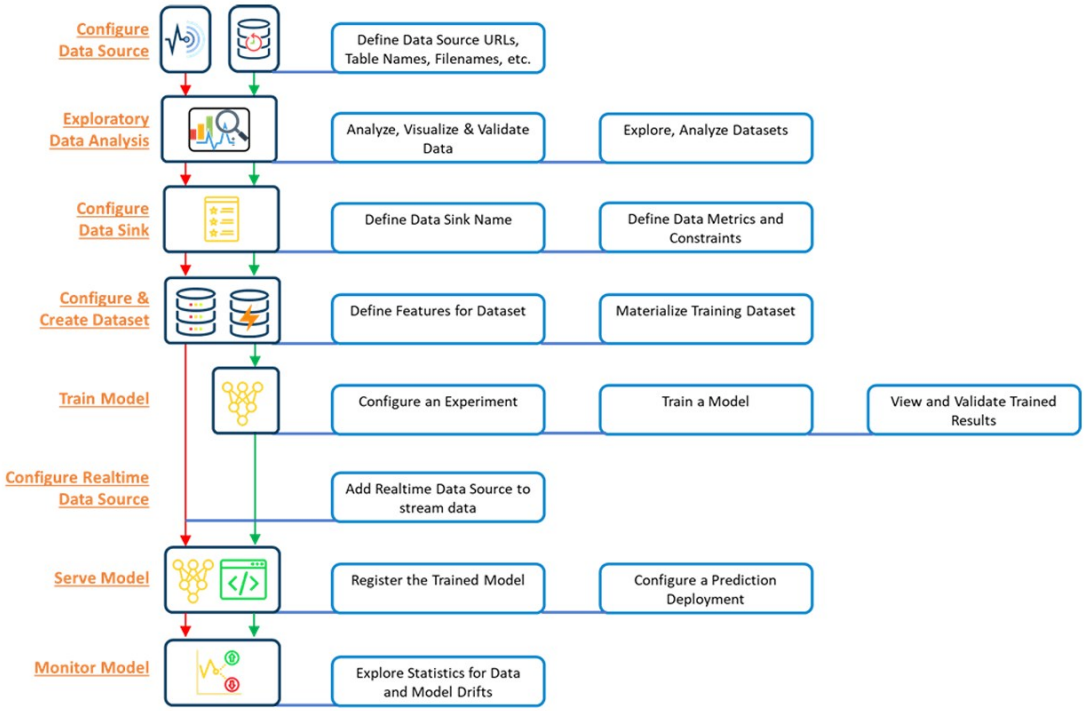
Traditional ML stacks force teams to stitch together a dozen point solutions: a streaming platform, a feature computation layer, a training scheduler, a model registry, and a serving framework — each with its own operational overhead. The result is:

- Training-serving skew: features computed differently at training time versus serving time, causing model degradation in production.
- Operational fragility: any single pipeline failure breaks the entire model lifecycle.
- Slow iteration: re-running experiments requires manual orchestration across disconnected tools.
- Governance blind spots: no unified lineage from raw data to model prediction.

Aizen Foresight was designed from the ground up to eliminate these failure modes — providing a single, cohesive platform that manages every stage of the ML lifecycle with a visual, low-code interface.

2. Aizen Foresight: Platform Overview

Aizen Foresight is a modular, containerized platform built on a streaming-first data architecture. It provides a visual canvas — ML Studio — through which data engineers, data scientists, and ML engineers collaborate to build, train, deploy, and monitor machine learning solutions without writing pipeline code.



The platform is governed by eight core design principles:

- Simple to use — visual, drag-and-drop pipeline construction for all ML lifecycle stages.
- Low latency — streaming ingestion, sub-millisecond feature retrieval, real-time inference.
- Flexible and pluggable — connects to any data store, streaming source, or cloud service.
- Batch and real-time processing — unified handling of both modes in a single pipeline.
- Scale up and out — elastic compute for both training and serving workloads.
- High availability — fault-tolerant architecture with no single points of failure.
- Built-in security and governance — data lineage, access control, and audit trails.
- Cloud or on-premises deployment — full flexibility for regulated and sovereign environments.

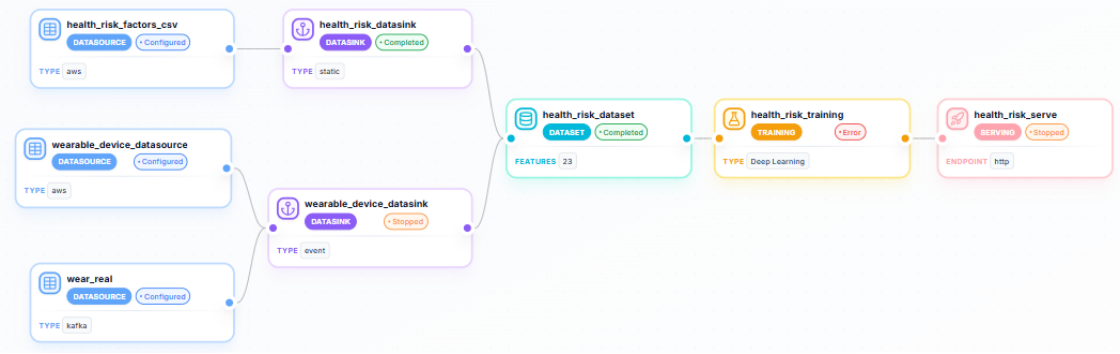
2.1 Core Pipeline Stages

Every Aizen Foresight solution flows through five major stages, each handled by dedicated containerized services:

Stage	Capability
Data Collection	Real-time and batch ingestion from any source — databases, Kafka, APIs, cloud storage.

Feature Engineering	Streaming and historical feature computation with unified training/serving logic.
Feature Store	Four-layer storage architecture: Memory, Offline, Online, and Metadata stores.
AutoML	Automated model selection, hyperparameter tuning, and dynamic GPU-backed training.
Serving	Containerized model deployment, REST API serving, versioning, and monitoring.

In ML Studio, users create end-to-end machine learning pipelines in a visual canvas using configurable nodes such as data sources, data sinks, datasets, training, and serving, with no code involved.



3. Data Collection

Aizen Foresight eliminates the need to individually build data ingestion pipelines. The platform automatically extracts data from multiple heterogeneous sources and routes it to the correct destination for real-time or batch processing — configured entirely through the visual canvas.

3.1 Real-Time Processing

Real-time ingestion involves the immediate capture, processing, and analysis of data as it arrives. Foresight connects to streaming sources such as Apache Kafka, event buses, and live database change streams to ingest data with sub-second latency. Processed data is immediately available downstream for feature computation and model inference.

Real-time processing is essential for time-sensitive domains — fraud detection, algorithmic trading, dynamic pricing, and industrial control systems — where stale data produces incorrect or dangerous decisions. Foresight handles the engineering complexity of high-throughput, low-latency ingestion transparently.

3.2 Batch Processing

For use cases that do not require sub-second freshness — historical reporting, weekly retraining, large-scale backfills — Foresight provides scheduled batch ingestion. Batch jobs can be triggered manually or automatically, and results are stored in the Offline Store for training data generation.

3.3 Data Exploration

Before building features or training models, data scientists need to understand their data. Foresight's exploration interface provides drag-and-drop visualization tools — histograms, scatter plots, bar charts, and box

plots — directly on ingested datasets. This enables rapid exploratory data analysis (EDA) without leaving the platform, ensuring that models are built on well-understood data.

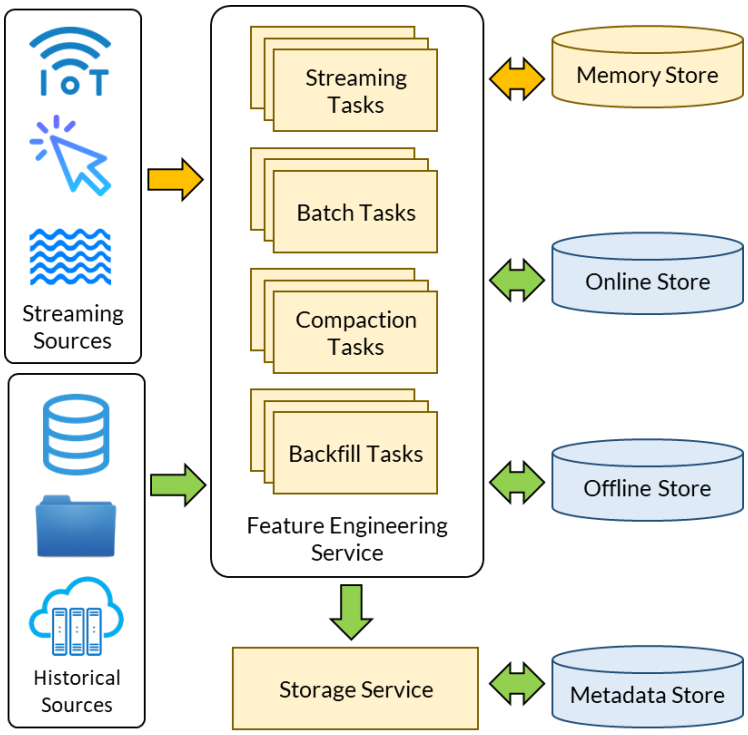
3.4 Data Quality

ML model performance is bound by input data quality. Foresight enforces data quality across multiple dimensions — completeness, consistency, accuracy, timeliness, and uniqueness — before data enters the feature engineering layer. Automated quality checks flag anomalies and prevent corrupt data from contaminating training datasets or live predictions.

4. Feature Engineering

Feature engineering is the most labor-intensive stage of any ML project. Foresight automates this through a dedicated Feature Engineering Service that fetches raw data from historical and live streaming sources, applies transformation logic, and writes computed features to the Feature Store.

Feature engineering jobs in Foresight support four task types: streaming transformation, batch transformation, compaction (merging and deduplication), and backfill (populating historical features for training datasets). All tasks are configured visually and executed as containerized jobs.



4.1 Real-Time Features

Real-time features are critical in industries where models must respond to events that occurred seconds or milliseconds ago. Foresight implements a real-time feature pipeline that listens to raw events published to Kafka by upstream services, applies transformation logic in-flight, and writes results to the Feature Store for immediate serving.

A key architectural principle of Foresight is unified feature logic: the same transformation code is used to generate features at training time (from historical data) and at serving time (from live streams). This eliminates training-serving skew — the most common cause of model degradation in production.

Real-Time Feature Capabilities

Sub-millisecond event processing with complex aggregation and windowing functions.

Single code path for training features and live serving features — no duplication, no skew.

Support for Kafka, database CDC streams, and scheduled batch reads as real-time sources.

Automatic feature freshness tracking and staleness detection.

5. Feature Store Architecture

Aizen Foresight's Feature Store is not merely a feature catalog — it is a full transformation and retrieval engine designed to solve the hard problems of real-time feature engineering. It is organized into four specialized storage layers, each optimized for a different access pattern.

5.1 Memory Store

The Memory Store is an in-memory cache that provides sub-millisecond feature retrieval during live prediction serving. It stores the most recently computed features for active entities — users, devices, accounts — in Apache Arrow columnar format. Arrow's zero-copy read model eliminates serialization overhead, enabling extremely low-latency data interchange between the store and model inference endpoints.

The Memory Store is distributed, fault-tolerant, and horizontally scalable, ensuring that prediction serving SLAs are maintained even under high concurrency.

5.2 Offline Store

The Offline Store persists raw data and computed features in key-sequenced row-format tables. It is purpose-built for storing large volumes of structured and unstructured data without a predefined schema — accommodating the rapidly evolving data models common in modern enterprises. The Offline Store integrates natively with cloud object storage services such as AWS S3 and Google Cloud Storage.

5.3 Online Store

The Online Store holds time-series and aggregated feature data in event-table format. Rows are written and read in Apache Arrow record batches, with each row keyed by a time-series column plus optional entity columns. The Online Store powers the Feature Serve layer during real-time inference, providing the freshest available feature values for incoming prediction requests.

5.4 Metadata Store

The Metadata Store maintains all descriptive metadata associated with features, feature sets, lineage relationships, and data quality records. It uses key-value pair storage built on key-sequenced tables, enabling fast discovery and search of feature definitions, their dependencies, and their associated transformation logic.

5.5 Feature Serve

The Feature Serve layer is the real-time retrieval interface that applications call when they need features for a live prediction. It uses the Online Store for fast feature lookup and exposes each model as an RPC (Remote Procedure Call) endpoint, allowing applications to integrate ML-powered personalization, anomaly detection, or optimization directly into their request paths with minimal latency impact.

6. Training Data Generation & AutoML

6.1 Generating Training Datasets

Aizen Foresight automates training dataset construction from the Feature Store. Users select the relevant feature sets, define the label (target variable), and specify the time window for historical data extraction. Foresight joins the selected features at the correct timestamps, handles missing values, and produces a labeled dataset ready for model training.

The platform also supports dataset previewing — users can inspect sample rows, visualize feature distributions, and validate label quality before committing to a training run.

Training Dataset Workflow

1. Add a Dataset node to the ML Studio canvas.
2. Select feature sets and define basis, aggregate, join-key, and expression features.
3. Preview data with built-in visualization tools.
4. Schedule a dataset generation job with configurable compute resources.
5. Proceed to AutoML with the validated dataset.

6.2 AutoML: Automated Model Selection and Tuning

Aizen Foresight's AutoML engine automates the two most time-consuming stages of model development: algorithm selection and hyperparameter tuning. Given a training dataset and a target metric, AutoML evaluates a search space of model architectures and configurations, identifies the optimal model, and makes it available for deployment — without manual experimentation.

AutoML supports both traditional ML algorithms (gradient boosting, random forests, logistic regression, support vector machines) and deep learning architectures (feed-forward networks, recurrent networks, 1D convolutional networks for time-series data).

6.3 Dynamic Model Training on Cloud GPU Clusters

For computationally intensive training jobs — particularly deep learning models — Foresight integrates with cloud GPU clusters across major providers. Users select their preferred GPU instance type directly from the ML Studio interface. Foresight displays estimated training costs upfront, enabling teams to balance performance against budget before committing to a run.

Once a GPU cluster is selected, Foresight automatically provisions the training container, maps training data from object storage, and initiates the job. When training completes, the model artifact is automatically registered in the model registry and tagged for deployment.

Dynamic Training Benefits

Choose any supported cloud GPU cluster from the ML Studio UI — no DevOps involvement required.

Training cost estimates shown before job submission.

Automatic data mapping from S3 or GCS — no manual storage access configuration.

Trained models automatically versioned and registered for reproducibility.

7. Building a Complete ML Pipeline in ML Studio

Aizen Foresight's ML Studio provides a canvas-based interface where every component of the ML pipeline is assembled visually. This section walks through the complete workflow for creating a production-ready pipeline — from raw data source to live model serving.

7.1 Creating a New Pipeline

A new pipeline begins in ML Studio. Clicking Create Pipeline opens a blank canvas within the context of an ML project. Projects organize related pipelines, datasets, models, and deployment configurations under a shared namespace, enabling team collaboration and asset reuse.

7.2 Configuring Data Sources

The first component added to the canvas is a Data Source node. Clicking the node opens a guided five-step configuration wizard that collects connection parameters, schema definitions, ingestion mode (real-time or batch), and refresh frequency. Supported sources include relational databases, Kafka topics, REST APIs, and cloud object storage.

7.3 Configuring Data Sinks

A Data Sink defines where ingested data lands before feature computation. Sinks can target the Offline Store (for batch and historical data), the Online Store (for time-series features), or external destinations. Once configured, clicking Start generates the initial data load and begins the ingestion schedule.

7.4 Creating Training Datasets

A Dataset node is added to the canvas and connected to one or more Data Sinks. The dataset configuration wizard guides users through defining features across four types:

- **Basis Features —** raw columns passed through directly from the source.
- **Aggregate Features —** computed summaries (sum, average, count, percentile) over time windows or entity groups.
- **Join-Key Features —** features derived by joining across multiple data sources on shared keys.
- **Expression Features —** custom computed columns using formula-based expressions.

7.5 Training the Model

With a validated dataset in place, the next step is training. Foresight supports two training paths on the canvas:

- **ML Training —** traditional machine learning algorithms (classification, regression, clustering, anomaly detection) managed by the AutoML engine.
- **DL Training —** deep learning model architectures for sequence data, time-series, or complex pattern recognition.

The user selects the target variable, the evaluation metric, and the GPU cluster (if required). Clicking Start Training submits the job. Once training completes, the optimal model is automatically registered with a version tag and made available for deployment.

7.6 Adding Real-Time Data Sources for Serving

Before deploying the model, any events-based Data Sink used for contextual features must be connected to a real-time data source. This ensures that the feature values seen during prediction are computed from live data — not stale historical data. Supported real-time sources include Kafka streams and database tables with periodic batch-read schedules.

7.7 Deploying and Serving the Model

Model serving is configured through a Prediction Deployment node added to the canvas. The deployment wizard collects the deployment name, the registered model version, and resource requirements (CPU/GPU, memory, replica count). Clicking Start Prediction provisions a containerized model server and exposes a REST endpoint for prediction requests.

Verifying the Deployment

Check deployment status via the status view on the Prediction Deployment node.

Use the built-in Test Prediction tool to auto-send sample requests and validate model responses.

Monitor prediction latency, throughput, and error rates from the ML Studio dashboard.

View the live endpoint URL in the status output for integration into application code.

7.8 The Complete Pipeline Canvas

When all stages are connected — Data Source → Data Sink → Dataset → Training → Real-Time Source → Prediction Deployment — the ML Studio canvas provides a unified view of the entire ML lifecycle. Each node displays its current status, health metrics, and resource consumption.

8. Container Services

Every job in Aizen Foresight — feature computation, model training, and prediction serving — runs inside a container. Containerization provides environmental consistency: the same model that trained on a developer's pipeline configuration will behave identically in production.

Kubernetes manages scheduling, scaling, and fault recovery for all containerized workloads. Foresight abstracts the Kubernetes complexity entirely — users configure resource requirements through the ML Studio interface, and the platform handles cluster provisioning, pod scheduling, and health monitoring automatically.

9. Application Integration

Aizen Foresight exposes all trained and deployed ML models through a standard REST API, making ML predictions consumable by any application — web, mobile, microservice, or data pipeline — with no ML infrastructure knowledge required on the integration side.

The prediction endpoint accepts input data as a JSON payload and returns predictions in the same format. Foresight handles feature retrieval from the Feature Store, model inference, and response formatting transparently.

Integration Pattern

Endpoint: POST https://<deployment-url>/predict

Request body: JSON object with input feature key-value pairs.

Response: JSON object with prediction output and confidence score.

Authentication: Bearer token or API key, configured at deployment time.

Latency: Sub-50ms end-to-end for typical feature retrieval + inference workloads.

This integration model enables a wide range of production use cases: real-time fraud scoring, dynamic pricing recommendations, predictive maintenance alerts, and personalized content ranking in customer-facing applications.

10. Enterprise Use Cases

Industry	Foresight Application
Financial Services	Real-time fraud detection on transaction streams with sub-100ms scoring via Kafka ingestion and in-memory Feature Store.
Telecommunications	Predictive churn scoring and network anomaly detection using time-series features from CDR streams.
Manufacturing	Predictive maintenance on equipment sensor streams; anomaly detection using 1D CNN models trained on vibration and temperature data.
Retail & E-Commerce	Dynamic pricing and personalized recommendation models updated continuously from clickstream and inventory streams.
Healthcare	Patient risk stratification models trained on EHR data with batch ingestion and daily retraining workflows.
Logistics	Route optimization and demand forecasting using batch-processed shipment history and real-time traffic event streams.

Conclusion

The gap between data and decision is where competitive advantage is won or lost. Enterprises that close this gap — that can turn a sensor reading, a transaction, or a customer action into an accurate, timely ML prediction in production — operate at a fundamentally different speed than those that cannot.

Aizen Foresight was built to close that gap. By unifying data ingestion, feature engineering, AutoML, and model serving on a single visual platform, Foresight eliminates the engineering overhead that has historically made Operational ML a months-long endeavor for specialist teams. With Foresight, a data scientist can assemble a production-grade ML pipeline in ML Studio, train a model on a cloud GPU cluster, and deploy it behind a REST endpoint — all without writing pipeline code or managing infrastructure.

The result is a machine learning system that is not a research artifact — it is a production-grade, continuously improving decision engine embedded in the business, and serving predictions at the speed that modern operations demand.

To Learn More

Contact Aizen Corporation to schedule a technical demonstration of Aizen Foresight.

Request access to a sandbox environment to explore ML Studio and the pipeline canvas.

Engage the Aizen solutions team to design a proof-of-concept for your specific use case.