COELHO REALTIME

A KUBERNETES-NATIVE MLOPS PLATFORM THAT FUSES INCREMENTAL LEARNING WITH BATCH INTELLIGENCE

TRANSACTION FRAUD DETECTION
ESTIMATED TIME OF ARRIVAL
E-COMMERCE CUSTOMER INTERACTIONS

INCREMENTAL MACHINE LEARNING
BATCH MACHINE LEARNING
DELTA LAKE SQL

Rafael Coelho

COELHO REALTIME

ABOUT THE PROJECT

COELHO RealTime is a **demonstrative**, **Kubernetes-native MLOps (Machine Learning Operations)** platform, built to demonstrate production-grade MLOps capabilities through simulated real-world data and scenarios.

It masterfully combines **Incremental Learning** with **Batch Intelligence** via a unique dual-pipeline, enabling both real-time model adaptation and robust batch training.

This platform addresses critical challenges in areas like **Transaction Fraud Detection**, **Estimated Time of Arrival**, and **E-commerce Customer Interactions**.

Orchestrated on K3d with end-to-end observability from **Prometheus**, **Grafana**, and **MLflow**, **COELHO RealTime** ensures the demonstration of accurate, transparent, and continuously optimized machine learning models for dynamic operational demands.

COELHO REALTIME

PLATFORM ARCHITECTURE

01.

INCREMENTAL & BATCH MACHINE LEARNING

The platform combines two complementary machine learning paradigms into a unified dual-pipeline architecture.

The Machine Learning Layer provides real-time model adaptation through River's incremental learning and robust batch training via Scikit-Learn and CatBoost.

The Data & Streaming Layer feeds these models through Apache Kafka event streaming and Apache Spark Structured Streaming, persisting all data as Delta Lake tables on MinIO.

The Application Services Layer exposes model training, predictions, and diagnostics through FastAPI, with Redis caching live models and MLflow tracking every experiment.

02.

INFRASTRUCTURE AS CODE & MICROSERVICES

The entire platform — 15 containerized microservices — is provisioned declaratively through Terraform and orchestrated on Kubernetes via K3d.

The Container Orchestration Layer establishes the cluster foundation, while the GitOps & CI/CD Layer automates the complete delivery pipeline from GitLab CI builds through ArgoCD deployments with automated image updates.

The Observability & Monitoring Layer ensures operational transparency through Prometheus metrics collection, Grafana visualization, and Alertmanager with dependency-aware inhibition rules, all backed by PostgreSQL metadata persistence.

01.

INCREMENTAL & BATCH MACHINE LEARNING

The platform combines two complementary machine learning paradigms into a unified dual-pipeline architecture.

River's incremental learning enables models to adapt in real-time as Kafka streams arrive, while Scikit-Learn and CatBoost deliver robust batch training on historical Delta Lake datasets. Both pipelines share the same data, the same tracking (MLflow), and the same serving layer (FastAPI) — ensuring every model is continuously optimized whether learning from one event or one million rows.

COELHO REALTIME

COELHO REALTIME

MAIN APPLICATIONS

Transaction Fraud Detection

This demonstrative application showcases an adaptive MLOps platform that detects and mitigates fraud in real-time, simulating real-world scenarios to protect assets and maintain trust.

Estimated Time of Arrival

This demonstrative application, simulating real-world data, provides highly accurate and dynamic ETAs by leveraging real-time data and predictive analytics, optimizing logistics and boosting customer satisfaction.

E-Commerce Customer Interactions

A demonstrative application, this project simulates real-world e-commerce scenarios to showcase how personalized recommendations and dynamic content can transform customer journeys, driving engagement, fostering loyalty, and increasing conversion rates.

MAIN APPLICATIONS

COELHO REALTIME

TRANSACTION FRAUD DETECTION

- **Dual-Pipeline Model Deployment:** Showcases combining Incremental Learning (River) for real-time fraud pattern adaptation with Batch Intelligence (Scikit-Learn and CatBoost) for robust historical analysis.
- **Kafka-driven Data Ingestion:** Demonstrates real-time transaction data streaming using Apache Kafka producers.
- **FastAPI Prediction Service:** Exposes TFD model predictions via FastAPI endpoints for low-latency inference.
- **Comprehensive Observability:** Fraud detection metrics are monitored with Prometheus, visualized in Grafana, and model experiments are tracked in MLflow.

ESTIMATED TIME OF ARRIVAL

- **Dual-Pipeline Model Deployment:** Utilizes Incremental Learning (River) for dynamic ETA adjustments based on live changes, and Batch Intelligence (Scikit-Learn and CatBoost) for foundational ETA model training.
- **Kafka-driven Data Streams:** Simulates and streams real-time data (e.g., traffic, weather) influencing ETAs via Apache Kafka.
- **FastAPI Prediction Service:** Provides ETA predictions through FastAPI endpoints.
- **Comprehensive Observability:** ETA model performance and system health are monitored with Prometheus, visualized in Grafana, and experiments are managed in MLflow.

E-COMMERCE CUSTOMER INTERACTIONS

- **Dual-Pipeline Model Deployment:** Combines Incremental Learning (River) for adapting to changing customer preferences and behavior with Batch Intelligence (Scikit-Learn) for comprehensive historical behavioral analysis.
- **Kafka-driven User Data:** Simulates and streams real-time e-commerce interaction data via Apache Kafka producers.
- **FastAPI Prediction Service:** Serves ECCI-related predictions through FastAPI.
- **Comprehensive Observability:** Customer interaction model metrics are tracked using Prometheus, visualized in Grafana, and experiment management with MLflow.

FOUNDATIONAL MLOPS COMPONENTS

Incremental Machine Learning

Leveraging frameworks like River, this component enables models (TFD, ETA, ECCI) to continuously learn from real-time data streams. This ensures dynamic adaptation and sustained accuracy, effectively combating concept drift in evolving environments.

Batch Machine Learning

This component establishes robust model baselines and deep insights by training on extensive historical datasets using Apache Spark, Scikit-Learn and CatBoost. It provides a powerful, scalable foundation for all TFD, ETA, and ECCI applications.

Delta Lake SQL

Powered by Delta Lake and accessible via SQL on DuckDB, this foundational layer serves as the central, reliable, and consistent data repository. It seamlessly integrates and supports both high-throughput streaming (from Kafka) and large-scale batch processing for all project data.

COELHO REALTIME

COELHO
REALTIME

02.

INFRASTRUCTURE AS CODE & MICROSERVICES

1 - CONTAINER ORCHESTRATION LAYER

2 - GITOPS & CI/CD LAYER

3 - DATA & STREAMING LAYER

4 - APPLICATION SERVICES LAYER

5 - OBSERVABILITY & MONITORING LAYER

1 - CONTAINER ORCHESTRATION LAYER

TERRAFORM

- Provisions K3d cluster with 1 server + 3 agent nodes through declarative HCL configuration
- Manages Kubernetes, Helm, and K3d providers in a single state file, ensuring consistent cluster setup
- Configures host volume mounts for persistent storage, enabling data survival across cluster rebuilds
- **Why Terraform:** Eliminates OS-specific scripts, providing an agnostic Infrastructure-as-Code approach that works across Linux, Windows, and macOS without requiring translations

K3D

- Runs lightweight K3s distributions inside Docker containers, replicating production Kubernetes locally
- Provides built-in container registry for local image distribution during CI/CD workflows
- Supports multi-node clusters (1 server + 3 agents) with configurable NodePort mappings for service access
- **Why K3d:** Enables a production-like Kubernetes environment on a single development machine with minimal resource overhead, bridging local development and cloud deployment

KUBERNETES

- Orchestrates all 15 containerized services with declarative manifests, managing scheduling, networking, and lifecycle
- Handles service discovery via ClusterIP and external access via NodePort, connecting frontend to backend seamlessly
- Enforces liveness/readiness probes, resource quotas, and automatic pod restart for self-healing operations
- **Why Kubernetes:** The industry-standard orchestration platform that provides the reliability, scalability, and operational patterns required for a production-grade microservice architecture

DOCKER

- Containerizes all custom services (FastAPI, SvelteKit, Kafka Producers) with optimized single-stage Dockerfiles
- Provides BuildKit-accelerated layer caching for rapid image rebuilds during development
- Serves as the runtime foundation for K3d's Kubernetes nodes, running the entire cluster in containers
- **Why Docker:** Standardizes every service into portable, reproducible containers — a prerequisite for Kubernetes orchestration and consistent behavior across development and production

2 - GITOPS & CI/CD LAYER

GITLAB

- Ships a CI/CD-ready .gitlab-ci.yml that builds three Docker images (FastAPI, SvelteKit, Kafka Producers) via Docker-in-Docker, pluggable into any external GitLab instance
- Tags every image with commit SHA and latest, pushing to the K3d local registry for full build traceability
- Triggers automated builds on every master branch commit, bridging code changes to the deployment pipeline within seconds
- **Why GitLab:** Portable pipeline definitions that plug into any GitLab instance, keeping the build-to-registry workflow environment-agnostic

ARGOCD

- Connects to an external ArgoCD via a pre-configured Application manifest declaring the complete Helm chart deployment for the coelho-realtime namespace
- Auto-syncs cluster state to match Git with automated pruning, self-healing, and rollback across all 15 microservices
- Paired with ArgoCD Image Updater, which watches the K3d registry for new images and commits updated tags back to Git
- **Why ArgoCD:** Ready-to-apply GitOps manifests that transform any external ArgoCD into a fully declarative, automated deployment pipeline

SKAFFOLD

- Orchestrates local development with hot-reload file sync for Python and TypeScript directly into running pods
- Manages port forwarding for 13 services simultaneously, exposing the complete platform on localhost
- Intelligent caching and selective rebuilds based on file change detection
- **Why Skaffold:** Eliminates the rebuild-redeploy cycle, enabling sub-second code changes in production-identical container environments

HELM

- Packages all 15 microservices into a single umbrella chart with seven subchart dependencies (MLflow, Redis, MinIO, PostgreSQL, kube-prometheus-stack, Kafka, Spark)
- Declarative values.yaml centralizes configuration for every service: resource limits, ports, environment variables, and feature toggles
- Conditional dependency enablement allows selective service deployment via simple boolean flags
- **Why Helm:** The standard Kubernetes package manager that turns the entire platform into a single, versioned, reproducible deployment unit

3 - DATA & STREAMING LAYER

APACHE KAFKA

- Central event backbone in KRaft mode (no ZooKeeper) across three topics: transaction_fraud_detection, estimated_time_of_arrival, and e_commerce_customer_interactions
- Fed by Python 3.13 producers generating realistic synthetic data (fraud patterns, geospatial deliveries, behavioral signals)
- Auto-creates topics with 3 partitions for parallel consumption by River incremental learners and Spark batch processors
- Why Kafka: Decouples data generation from consumption, letting the same stream feed both real-time and batch ML simultaneously

APACHE SPARK

- Runs three Structured Streaming jobs (kafka_to_delta_tfd.py, kafka_to_delta_eta.py, kafka_to_delta_ecommerce.py) consuming Kafka topics into Delta Lake tables
- Configured with Delta Lake 4.0.0 JARs and Hadoop-AWS connector for native MinIO S3 compatibility
- Deployed via Kubernetes ConfigMaps managed as part of the Helm chart lifecycle
- **Why Spark:** Transforms raw Kafka streams into queryable Delta Lake tables, powering batch ML training with Scikit-Learn and CatBoost

DELTA LAKE

- Stores all streaming data as versioned, ACID-compliant tables on MinIO, supporting both streaming writes and batch reads
- Enables time-travel queries and schema enforcement for data consistency across ML workloads
- Queryable via DuckDB SQL through FastAPI endpoints, providing ad-hoc lakehouse access without Spark sessions
- **Why Delta Lake:** Unifies streaming and batch data into a single lakehouse format with ACID transactions, eliminating separate data stores

MINIO

- Provides S3-compatible storage hosting two buckets: lakehouse for Delta Lake tables and mlflow-artifacts for experiment outputs
- Persists data via hostPath volume mount (/data/minio), ensuring survival across pod restarts and cluster rebuilds
- Exposes S3 API on port 9000, integrating seamlessly with Spark, MLflow, and any S3-compatible tooling
- **Why MinIO:** Production-grade S3 storage locally, enabling cloud-identical patterns without external dependencies or costs

4 - APPLICATION SERVICES LAYER

FASTAPI

- Unified ML backend with three versioned API groups: /api/v1/incremental (River), /api/v1/batch (Scikit-Learn/CatBoost), /api/v1/sql (DuckDB queries)
- Manages training subprocesses with orjson serialization and Redis-backed model caching for low-latency predictions
- Instrumented with Prometheus metrics for request latency, throughput, and ML pipeline monitoring
- **Why FastAPI:** Single entry point unifying incremental learning, batch training, and data querying into one coherent async API surface

SVELTEKIT

- Interactive frontend with SvelteKit 2.20, Vite 6.0, TailwindCSS 4.0, and Plotly.js for ML visualizations
- Dedicated UI sections for TFD, ETA, and ECCI with training controls, predictions, metrics dashboards, and SQL editors
- TypeScript strict mode with pre-bundled Vite caching for optimized container startup
- **Why SvelteKit:** Reactive framework delivering instant UI updates for real-time ML metrics with minimal bundle overhead

REDIS

- Caches live incremental ML models in memory for sub-millisecond prediction serving with serialized River model state
- Manages training status flags for real-time frontend progress display (LIVE/FINISHED) per application
- ClusterIP service with configurable memory limits and eviction policies
- **Why Redis:** In-memory speed bridging model training (seconds) and prediction latency (milliseconds) in the incremental pipeline

MLFLOW

- Tracks every experiment with parameters, metrics, and artifacts across both incremental and batch training pipelines
- Metadata in PostgreSQL, artifacts on MinIO S3 — persistent history and full reproducibility
- Tracking UI with run comparison and model registry integrated into the SvelteKit frontend
- **Why MLflow:** Complete ML lifecycle management making every training run auditable, comparable, and reproducible

5 - OBSERVABILITY & MONITORING LAYER

PROMETHEUS

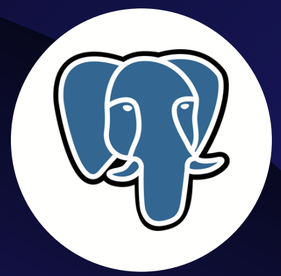
- Collects metrics from all 15 microservices via ServiceMonitors at 30-second intervals, centralizing operational and ML telemetry
- Deploys via kube-prometheus-stack (v80.6.0) with 800+ alerting rules covering infrastructure health and resource utilization
- Includes Node Exporter, kube-state-metrics, and Prometheus Operator for declarative monitoring
- **Why Prometheus:** Cloud-native monitoring standard providing pull-based collection, PromQL querying, and alerting at scale

GRAFANA

- Custom dashboards for ML pipeline throughput, API latency, and infrastructure health visualization
- Shares PostgreSQL backend with MLflow for unified metadata storage and reduced footprint
- Dashboard-as-code provisioning through Helm configuration for version-controlled monitoring
- **Why Grafana:** Transforms raw Prometheus metrics into actionable visual intelligence for rapid pipeline diagnosis

ALERTMANAGER

- Severity-based alert routing with dependency-aware inhibition rules preventing alert storms from cascading failures
- Configured with multi-channel notification for ML pipeline failures and infrastructure degradation
- Integrates with Prometheus rules for proactive incident detection before user impact
- **Why Alertmanager:** Converts detected anomalies into actionable notifications, surfacing issues before manual discovery



POSTGRESQL

- Shared relational backend for MLflow experiment tracking and Grafana dashboard/user storage
- Persists via hostPath mount (/data/postgresql) ensuring metadata survival across restarts and rebuilds
- ClusterIP service with configurable resource limits and connection pooling
- **Why PostgreSQL:** ACID-compliant relational storage for persistent, queryable metadata across ML tracking and monitoring

COELHO REALTIME

PYTHON ECOSYSTEM

1 - ML FRAMEWORKS


River

Scikit-Learn


CatBoost

Imbalanced-learn

PySpark

MLflow

2 - DATA & STORAGE


Pandas


NumPy

DuckDB

3 - VISUALIZATION & DIAGNOSTICS


YellowBrick

Scikit-plot

UMAP

NLTK

4 - WEB FRAMEWORK

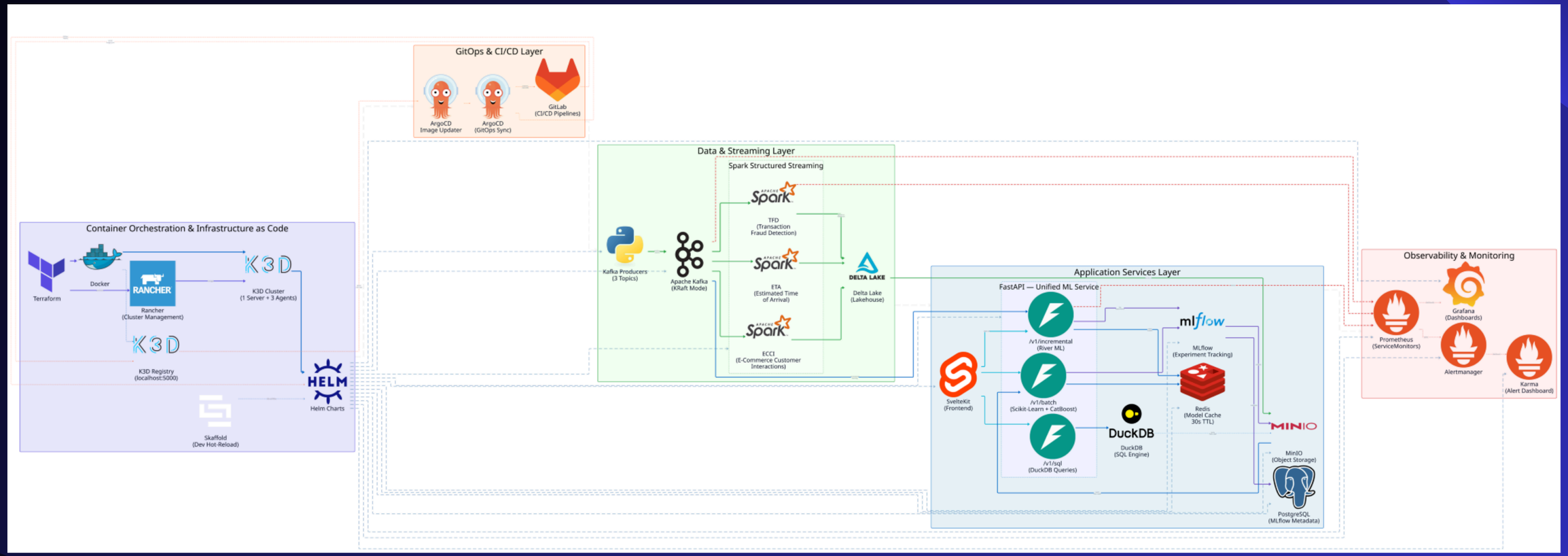
FastAPI


Pydantic

Redis

COELHO REALTIME

PLATFORM ARCHITECTURE



THE PLATFORM IN ACTION

A visual walkthrough of the COELHO RealTime platform — from its three demonstrative applications (Transaction Fraud Detection, Estimated Time of Arrival, and E-Commerce Customer Interactions) to its microservices infrastructure (Kafka, Spark, MLflow, Grafana, ArgoCD) — showcasing Incremental Machine Learning, Batch Machine Learning, and Delta Lake SQL capabilities running end-to-end on Kubernetes.

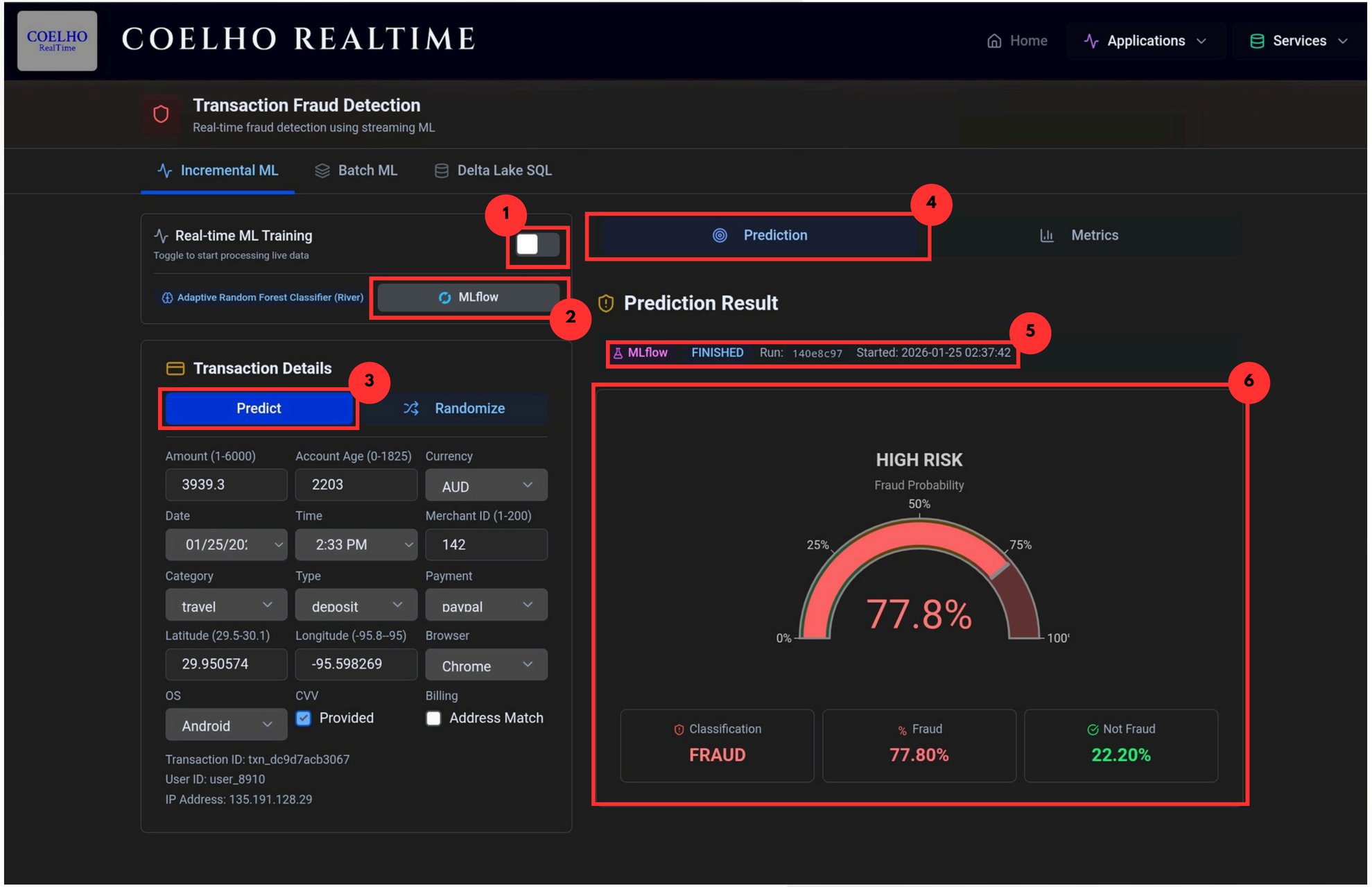
TRANSACTION FRAUD DETECTION

Incremental Machine Learning
Batch Machine Learning
Delta Lake SQL

- **Incremental ML model:** Adaptive Random Forest Classifier (River)
- **Batch ML model:** CatBoost Classifier

COELHO REALTIME

TRANSACTION FRAUD DETECTION - INCREMENTAL MACHINE LEARNING



1 Real-time ML training toggle

Activates or deactivates the continuous, real-time adaptation of the TFD model, enabling it to learn incrementally from incoming data streams.

2 MLflow model page - experiment run

Provides a summary of a specific incremental learning experiment for TFD on MLflow service page, detailing logged parameters, metrics, and associated artifacts within MLflow.

3 Prediction button

Initiates an immediate fraud prediction using the TFD model's current, incrementally updated state, demonstrating low-latency inference.

4 Prediction tab

Displays the interface where real-time fraud prediction outputs are presented.

5 MLflow experiment run informations

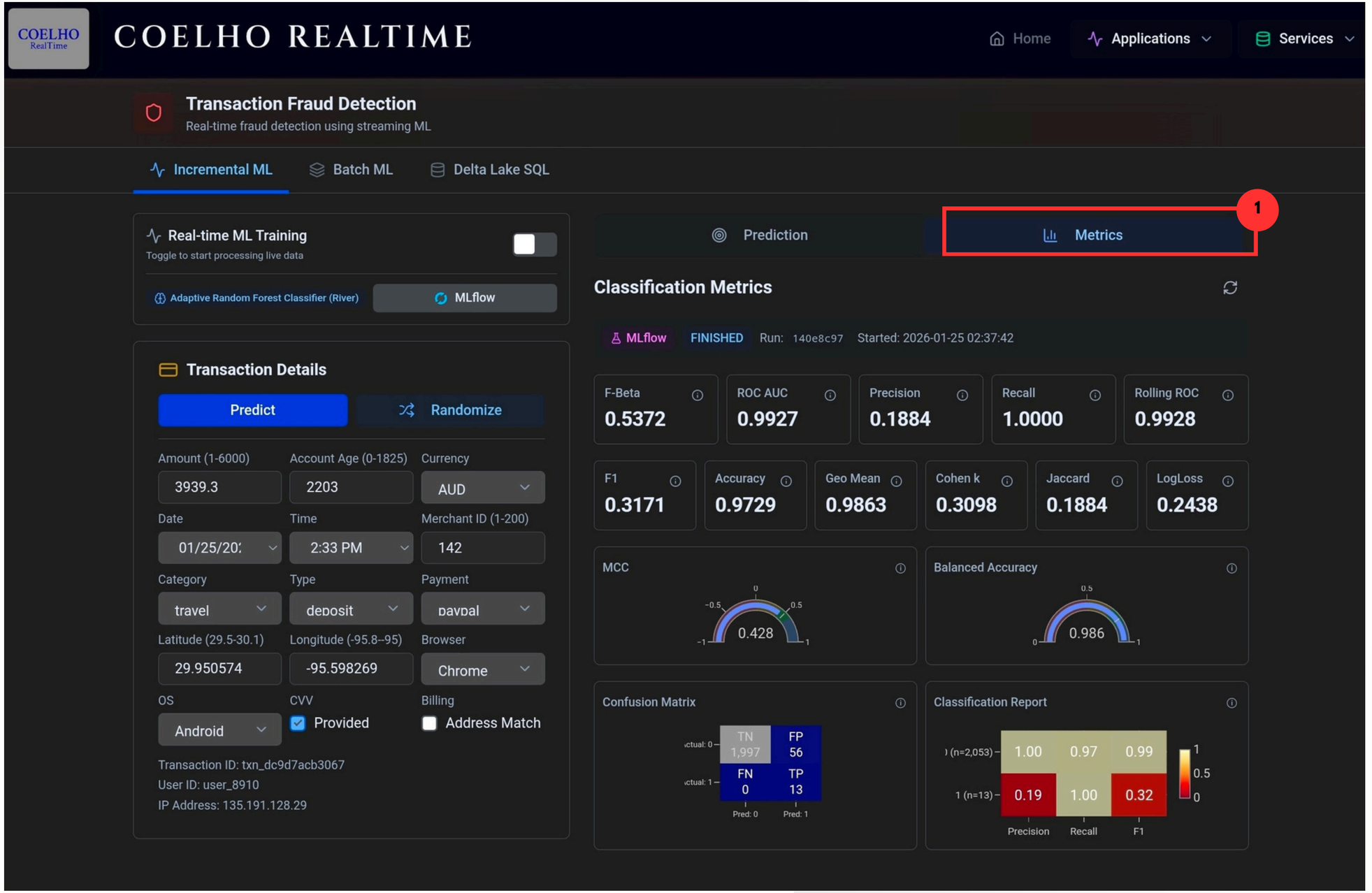
Offers status info (LIVE | FINISHED), MLflow run ID, and the datetime the last real-time ML model training was executed.

6 Prediction results

Presents the output of the TFD model's real-time fraud predictions

COELHO REALTIME

TRANSACTION FRAUD DETECTION - INCREMENTAL MACHINE LEARNING



1 Metrics tab

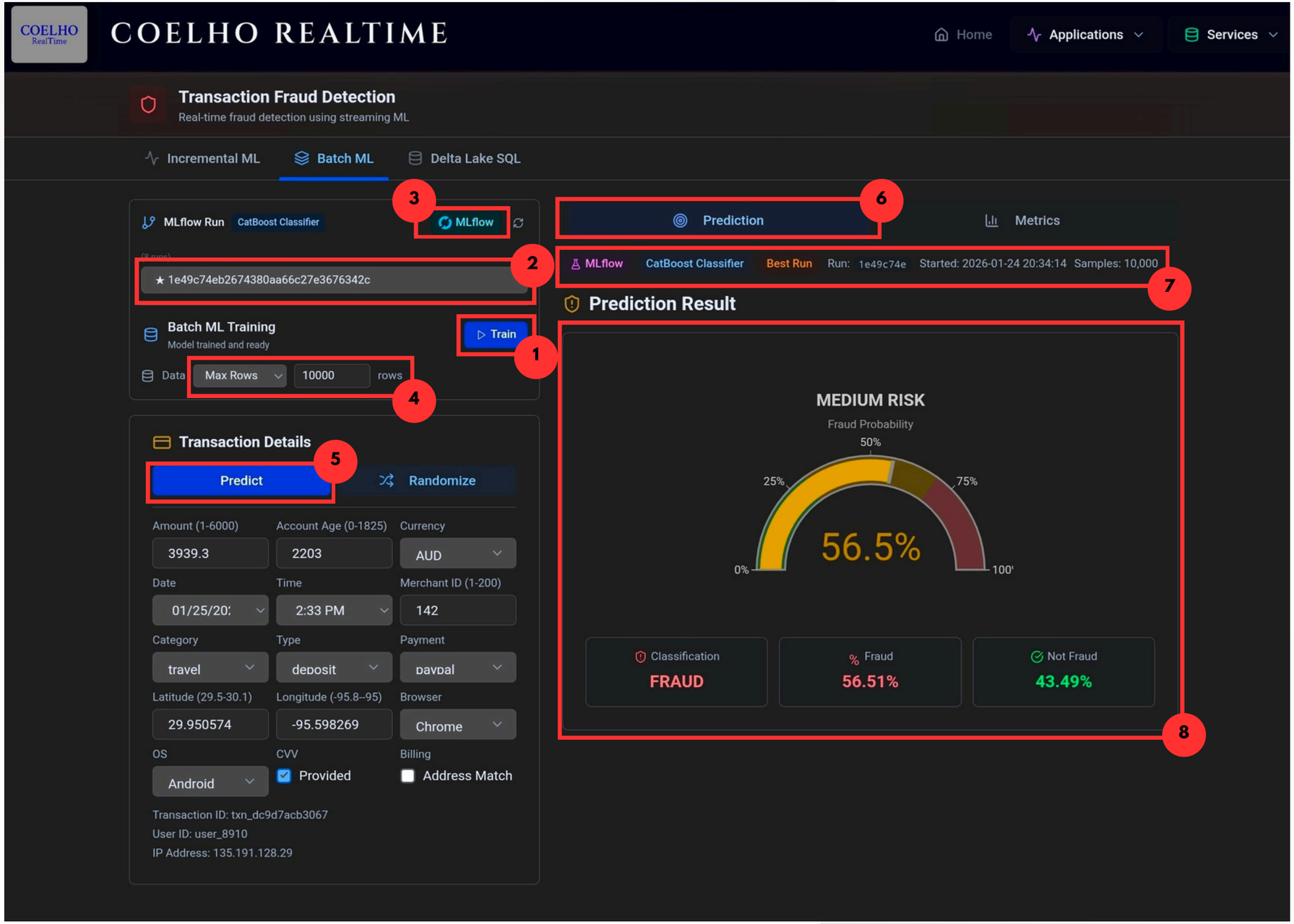
Presents the real-time metrics and incremental learning status of the TFD model.

Metrics (River library)

- Recall
- Precision
- F1
- F-Beta
- Accuracy
- Balanced Accuracy
- Matthews Correlation Coefficient
- Geometric Mean
- Cohen Kappa
- Jaccard
- ROC-AUC
- Rolling ROC-AUC
- Log Loss
- Confusion Matrix
- Classification Report

COELHO REALTIME

TRANSACTION FRAUD DETECTION - BATCH MACHINE LEARNING



1 Train button

Initiates the batch machine learning training process for the TFD model.

2 MLflow run ID - ML model

Dropdown displaying the MLflow run IDs associated with batch-trained TFD model the user can pick after training new Batch ML models.

3 MLflow run ID page

Navigates to or provides detailed information within MLflow about a particular experiment run.

4 Batch ML Training options

Provides the user the options of training new Batch ML models in real-time per maximum number of rows or per percentage of rows from the original TFD dataset.

5 Prediction button

Triggers a prediction using the currently selected or trained batch ML model.

6 Prediction tab

Displays the interface or section where batch prediction results for the TFD model are presented.

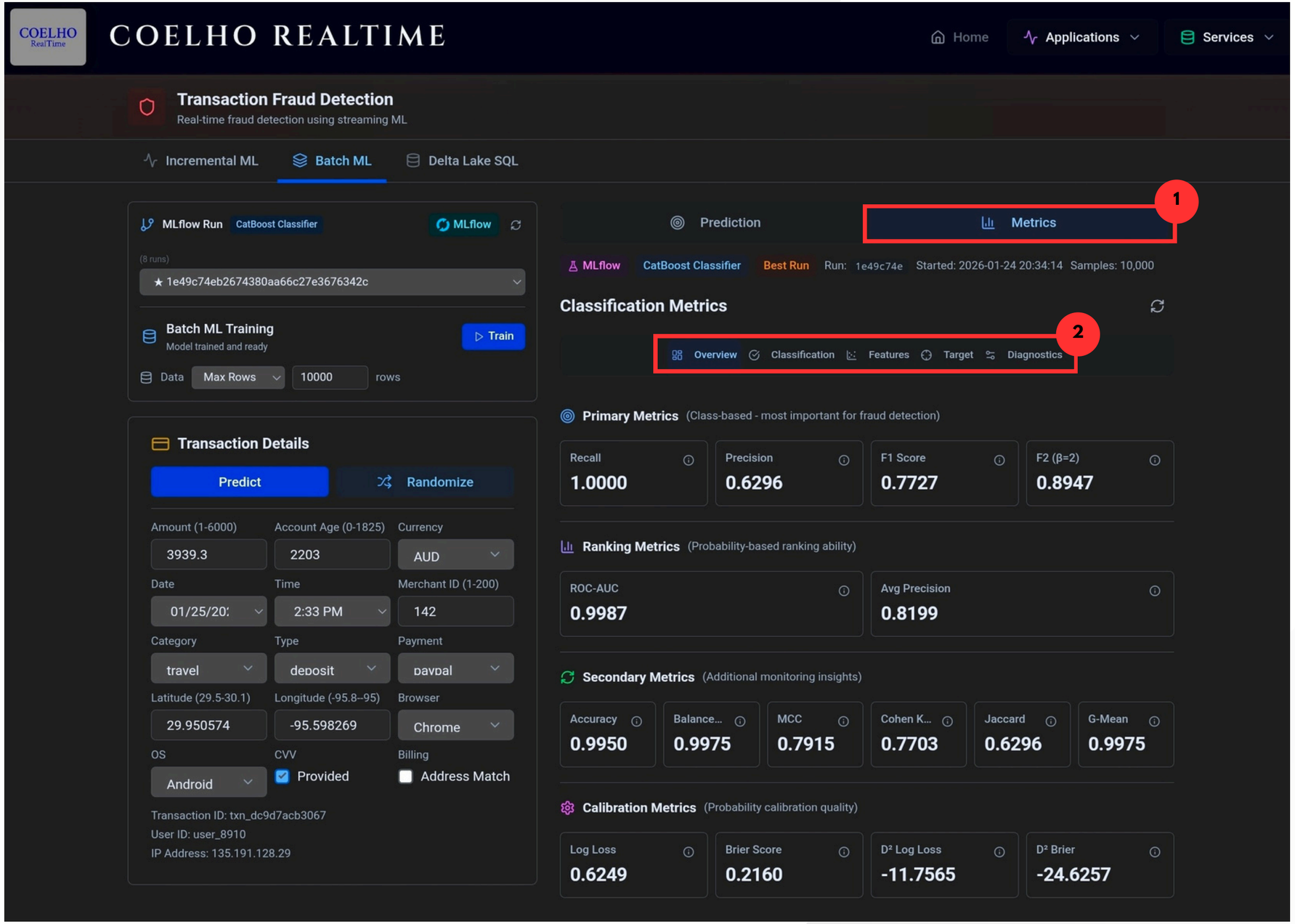
7 MLflow experiment run informations

Offers detailed status information, run ID, and other metadata for an MLflow experiment run.

8 Prediction results

Presents the output of the TFD model's real-time fraud predictions.

TRANSACTION FRAUD DETECTION - BATCH MACHINE LEARNING



1 Metrics tab

Presents various metrics of the TFD model. Includes some tabs with a lot of visual metrics from Scikit-Learn, Scikit-Plot and YellowBrick libraries.

2 Classification metrics tabs

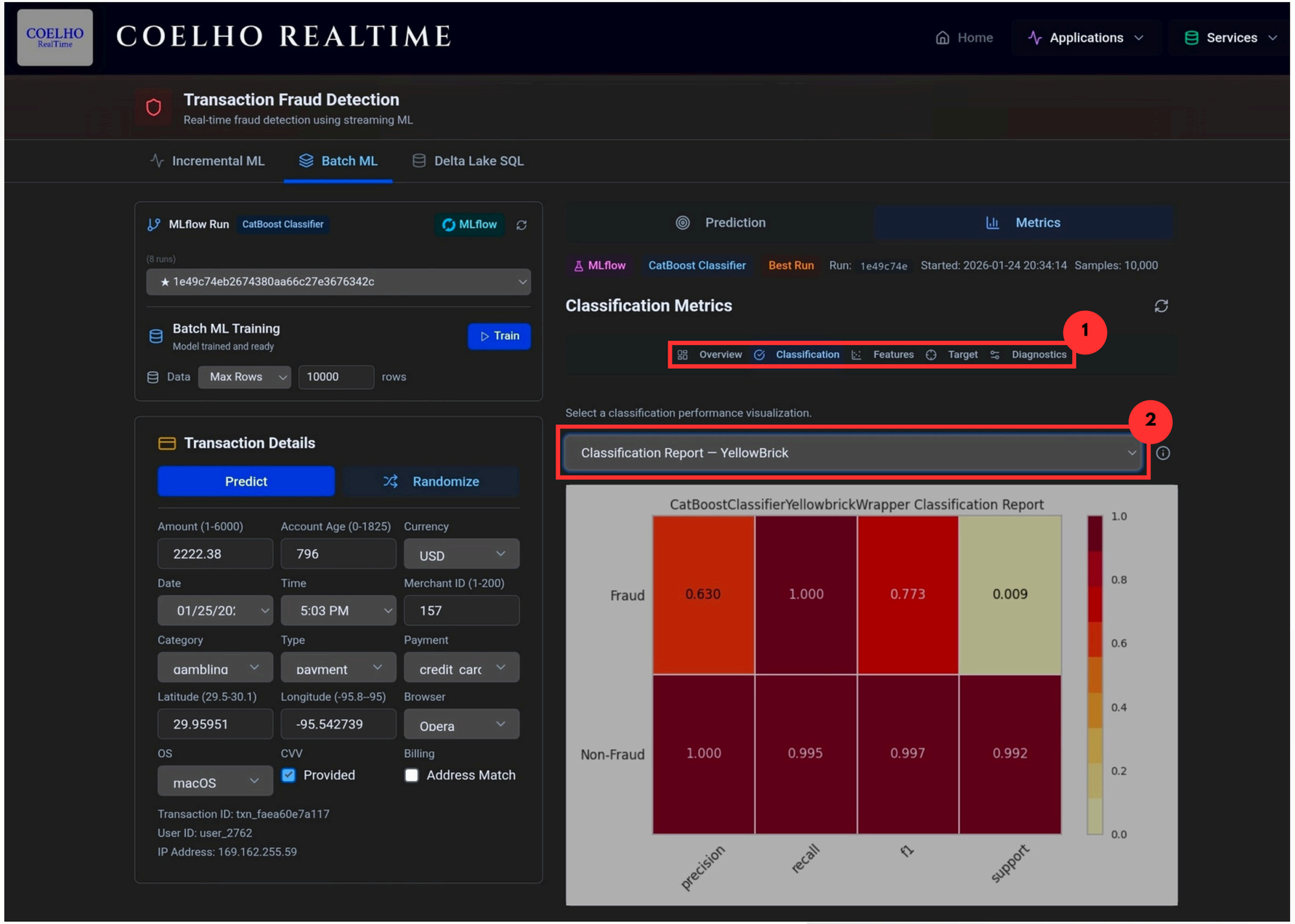
- **Overview:** Displays key MLflow metrics, including primary, ranking, secondary, and calibration scores.
- **Classification:** Provides various visualizations to assess classification model performance.
- **Features:** Offers visual tools for analyzing input features and their properties.
- **Target:** Presents visualizations related to the target variable's distribution and characteristics.
- **Diagnostics:** Contains visualizations for model selection, debugging, and advanced diagnostics.

Metrics (Scikit-Learn library)

- Recall
- Precision
- F1 Score
- F2 (β=2)
- ROC-AUC
- Average Precision
- Accuracy
- Balanced Accuracy
- Matthews Correlation Coefficient
- Cohen Kappa
- Jaccard
- Geometric Mean
- Log Loss
- Brier Score
- D^2 Log Loss
- D^2 Brier

COELHO REALTIME

TRANSACTION FRAUD DETECTION - BATCH MACHINE LEARNING



1 Classification Metrics tabs

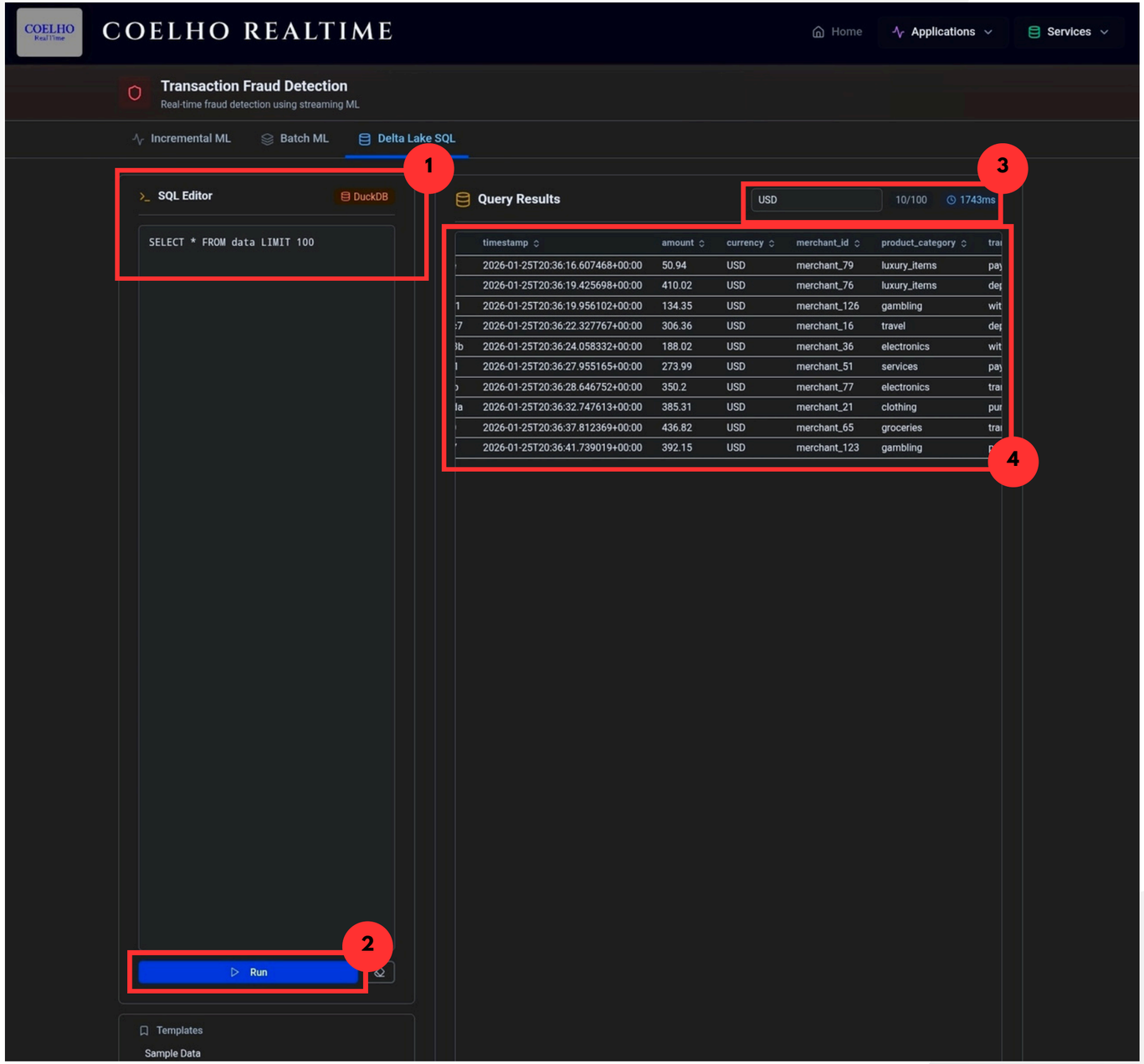
Overview, Classification, Features, Target, Diagnostics

2 Classification metrics tabs options

- **Classification**
  - Confusion Matrix
  - Classification Report
  - ROC AUC Curve
  - Precision-Recall Curve
  - Class Prediction Error
  - Discrimination Threshold
  - Calibration Curve
  - DET Curve
  - ROC Curve
  - KS Statistic
  - Cumulative Gain
  - Lift Curve
- **Features**
  - Rank 1D
  - Rank 2D
  - PCA Decomposition
  - Manifold
  - Parallel Coordinates
  - RadViz
  - Joint Plot
  - Partial Dependence
  - Decision Boundary
  - PCA 2D Projection
  - PCA Component Variance
  - Feature Importances
- **Target**
  - Class Balance
  - Feature Correlation (Mutual Info)
  - Feature Correlation (Pearson)
  - Balanced Binning Reference
- **Diagnostics**
  - Feature Importances
  - Cross-Validation Scores
  - Validation Curve
  - Learning Curve
  - Recursive Feature Elimination
  - Dropping Curve

COELHO REALTIME

TRANSACTION FRAUD DETECTION - DELTA LAKE SQL



1 SQL Editor

Interface for writing and editing SQL queries against the TFD Delta Lake.

2 Query run button

Executes the SQL query in the editor against the TFD Delta Lake.

3 Search box, rows and latency

Displays search, allowing user to filter specific values over the entire result, total rows returned, and query execution time.

4 Query results with filter

Presents the results of the SQL query with values filtered on the filter box.

COELHO REALTIME

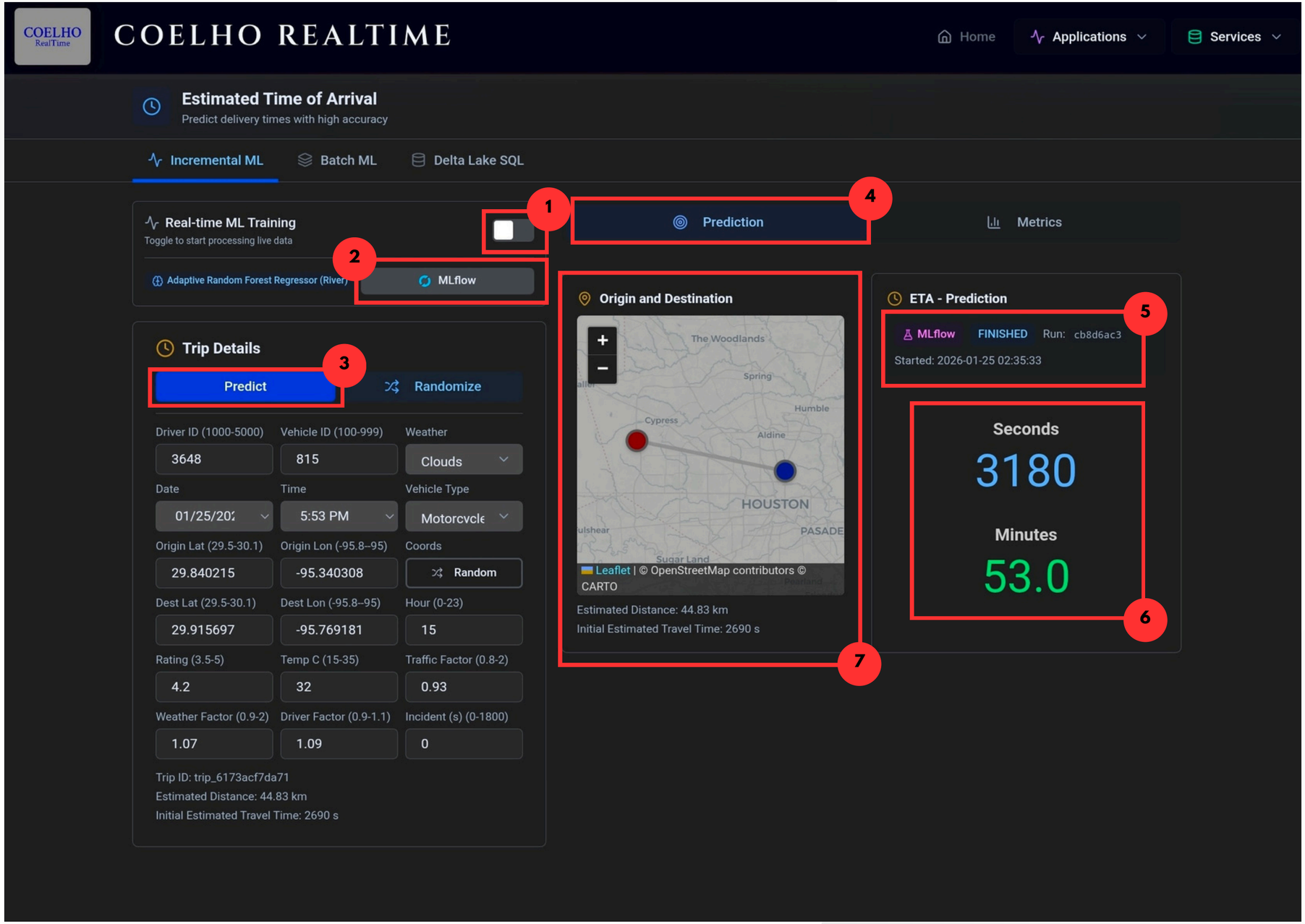
ESTIMATED TIME OF ARRIVAL

Incremental Machine Learning
Batch Machine Learning
Delta Lake SQL

- **Incremental ML model:** Adaptive Random Forest Regressor (River)
- **Batch ML model:** CatBoost Regressor

COELHO REALTIME

ESTIMATED TIME OF ARRIVAL - INCREMENTAL MACHINE LEARNING



1 Real-time ML training toggle

Activates or deactivates the continuous, real-time adaptation of the ETA model, enabling it to learn incrementally from incoming data streams.

2 MLflow model page - experiment run

Provides a summary of a specific incremental learning experiment for ETA on MLflow service page, detailing logged parameters, metrics, and associated artifacts within MLflow.

3 Prediction button

Initiates an immediate ETA prediction using the ETA model's current, incrementally updated state, demonstrating low-latency inference.

4 Prediction tab

Displays real-time estimated time of arrival (ETA) predictions from the incremental machine learning model, offering immediate and continuously updated insights.

5 MLflow experiment run informations

Offers status info (LIVE | FINISHED), MLflow run ID, and the datetime the last real-time ML model training was executed.

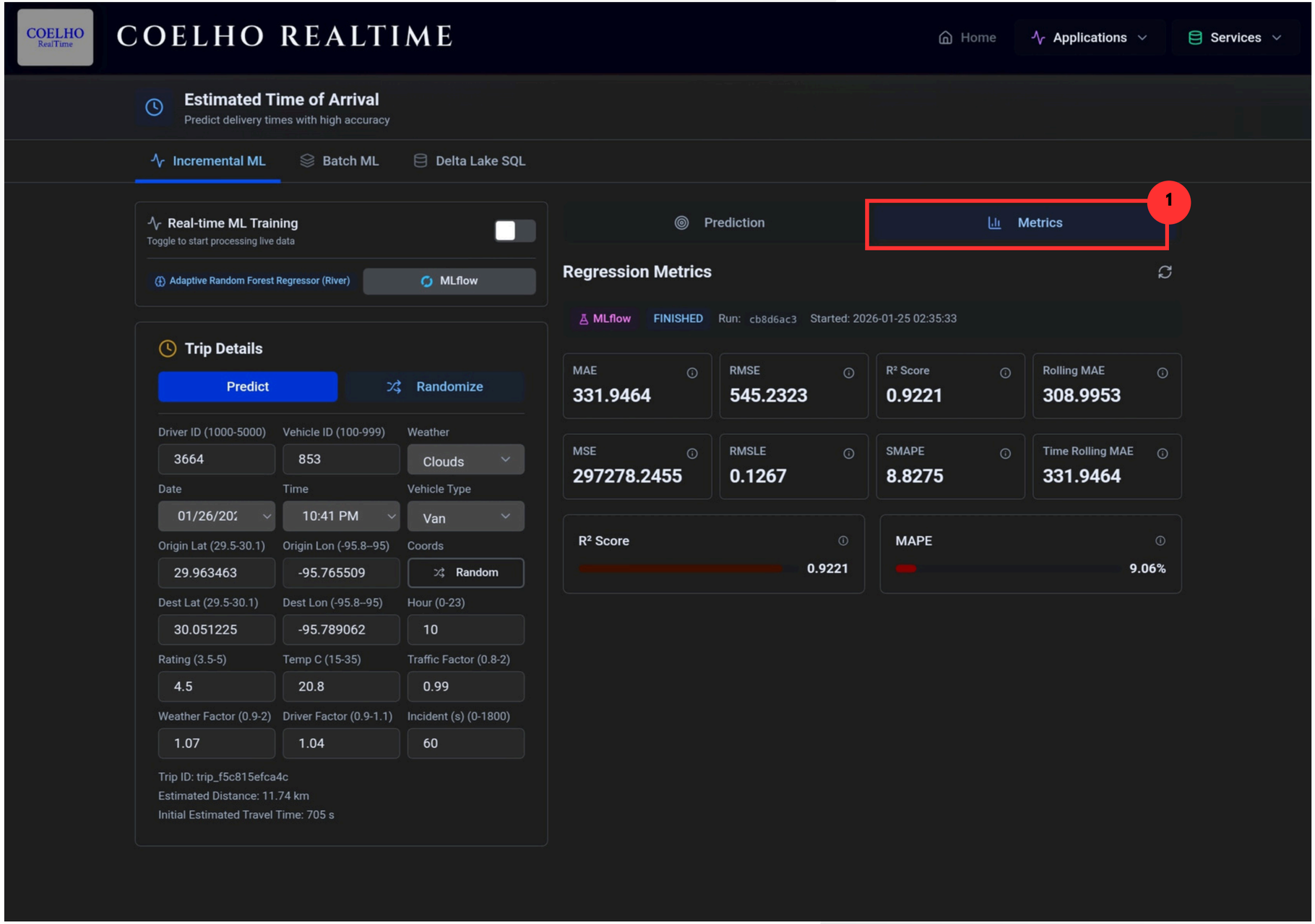
6 Prediction results

Presents the output of the ETA model's real-time predictions.

7 Map with Origin and Destination

Illustrates the real-time or predicted path of an entity, clearly marking its starting point (Origin) and its projected endpoint (Destination) to support dynamic ETA calculations.

ESTIMATED TIME OF ARRIVAL - INCREMENTAL MACHINE LEARNING



1 Metrics tab

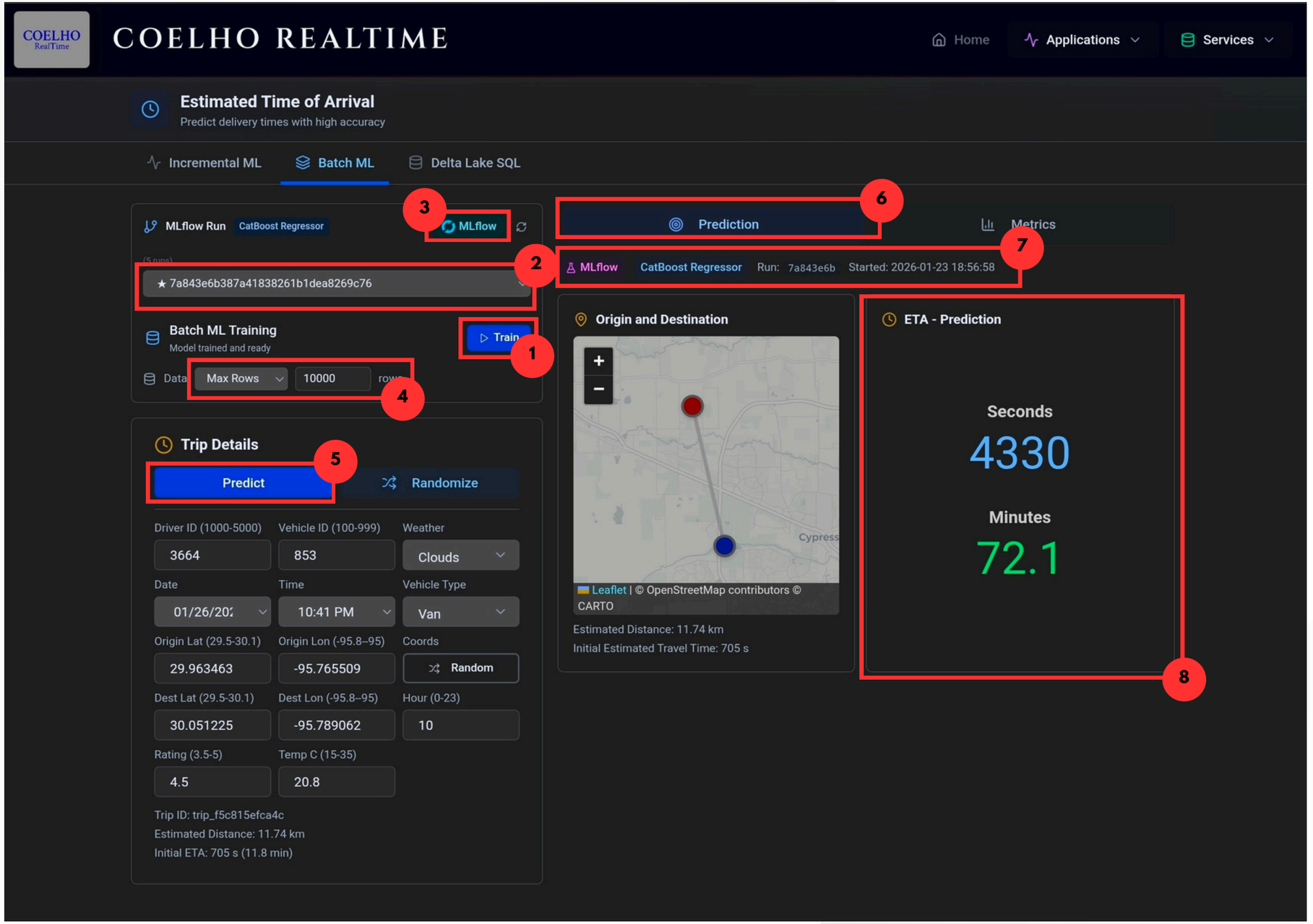
Presents the real-time metrics and incremental learning status of the ETA model.

Metrics (River library)

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- Mean Squared Error (MSE)
- R-Squared (R^2)
- Symmetric MAPE (SMAPE)
- Root Mean Squared Log Error (RMSLE)
- Rolling MAE
- Rolling RMSE
- Time Rolling MAE

COELHO REALTIME

ESTIMATED TIME OF ARRIVAL - BATCH MACHINE LEARNING



1 Train button

Initiates the batch machine learning training process for the ETA model.

2 MLflow run ID - ML model

Dropdown displaying the MLflow run IDs associated with batch-trained ETA model the user can pick after training new Batch ML models.

3 MLflow run ID page

Navigates to or provides detailed information within MLflow about a particular experiment run.

4 Batch ML Training options

Provides the user the options of training new Batch ML models in real-time per maximum number of rows or per percentage of rows from the original ETA dataset.

5 Prediction button

Triggers a prediction using the currently selected or trained batch ML model.

6 Prediction tab

Displays the interface or section where batch prediction results for the ETA model are presented.

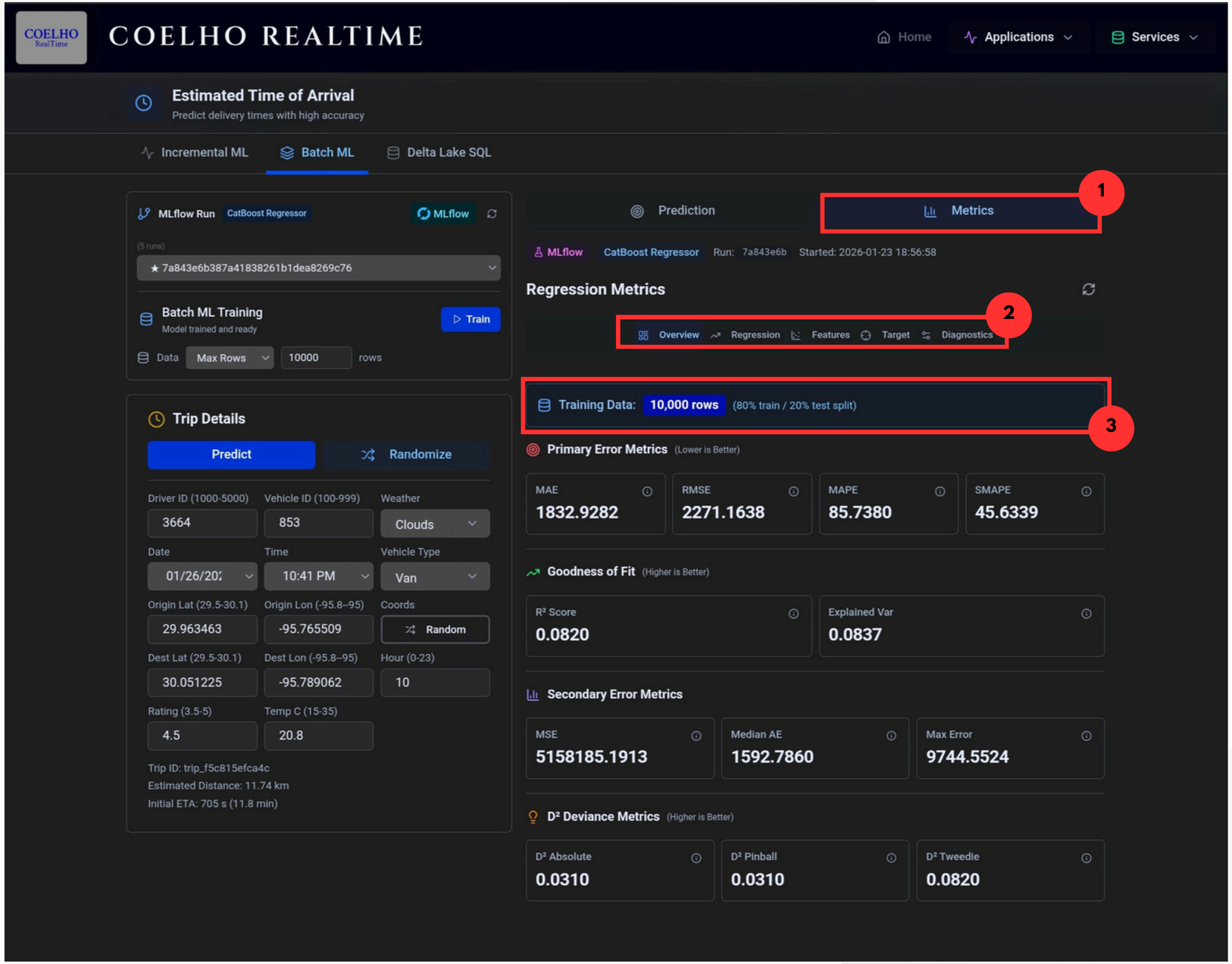
7 MLflow experiment run informations

Offers detailed status information, run ID, and other metadata for an MLflow experiment run.

8 Prediction results

Presents the output of the ETA model's real-time predictions.

ESTIMATED TIME OF ARRIVAL - BATCH MACHINE LEARNING



1 Metrics tab

Presents various metrics of the ETA model. Includes some tabs with a lot of visual metrics from Scikit-Learn, Scikit-Plot and YellowBrick libraries.

2 Regression metrics tabs

- **Overview:** Displays key MLflow metrics, including primary, ranking, secondary, and calibration scores.
- **Regression:** Provides various visualizations to assess regression model performance.
- **Features:** Offers visual tools for analyzing input features and their properties.
- **Target:** Presents visualizations related to the target variable's distribution and characteristics.
- **Diagnostics:** Contains visualizations for model selection, debugging, and advanced diagnostics.

3 Training data info

Displays the total amount of rows used to train the Batch ML model

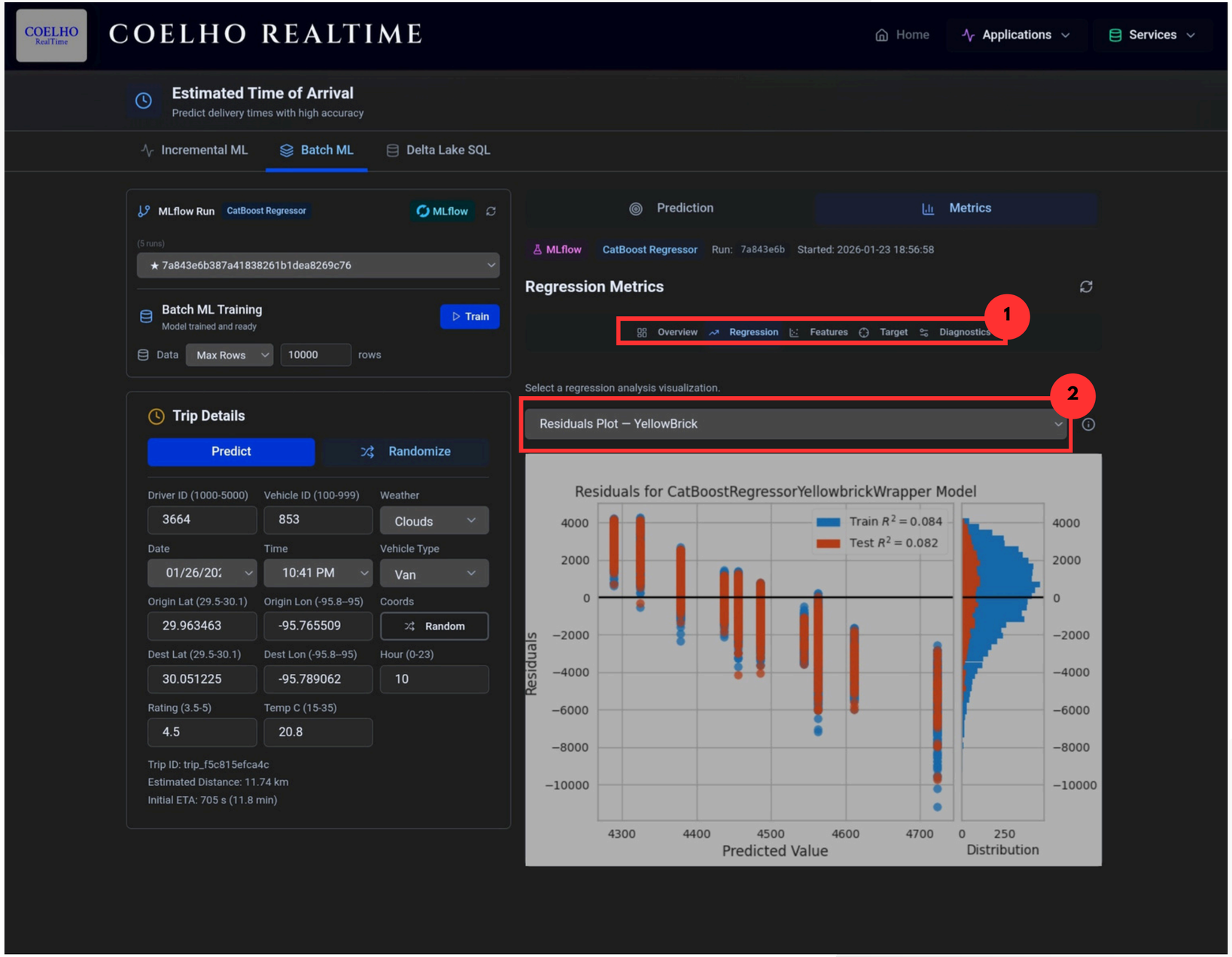
---

Metrics (Scikit-Learn library)

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Percentage Error (MAPE)
- Symmetric Mean Absolute Percentage Error (SMAPE)
- R2 Score
- Explained Variance Score
- Mean Squared Error (MSE)
- Median Absolute Error (Median AE)
- Max Error
- D2 Absolute Error Score
- D2 Pinball Score
- D2 Tweedie Score

COELHO REALTIME

ESTIMATED TIME OF ARRIVAL - BATCH MACHINE LEARNING



1 Regression Metrics tabs

Overview, Regression, Features, Target, Diagnostics

2 Regression metrics tabs options

- **Regression**
  - Residuals Plot
  - Prediction Error
  - Prediction Error (Actual vs Predicted)
  - Prediction Error (Residuals)
- **Features**
  - Rank 1D
  - Rank 2D
  - PCA Decomposition
  - Manifold
  - Joint Plot
  - Partial Dependence
  - PCA 2D Projection
  - PCA Component Variance
- **Target**
  - Feature Correlation (Mutual Info)
  - Feature Correlation (Pearson)
  - Balanced Binning Reference
- **Diagnostics**
  - Feature Importances
  - Cross-Validation Scores
  - Validation Curve
  - Learning Curve
  - Recursive Feature Elimination
  - Dropping Curve

COELHO REALTIME

ESTIMATED TIME OF ARRIVAL - DELTA LAKE SQL

1 SQL Editor

Interface for writing and editing SQL queries against the ETA Delta Lake.

2 Query run button

Executes the SQL query in the editor against the ETA Delta Lake.

3 Search box, rows and latency

Displays search, allowing user to filter specific values over the entire result, total rows returned, and query execution time.

4 Query results with filter

Presents the results of the SQL query with values filtered on the filter box.

COELHO REALTIME

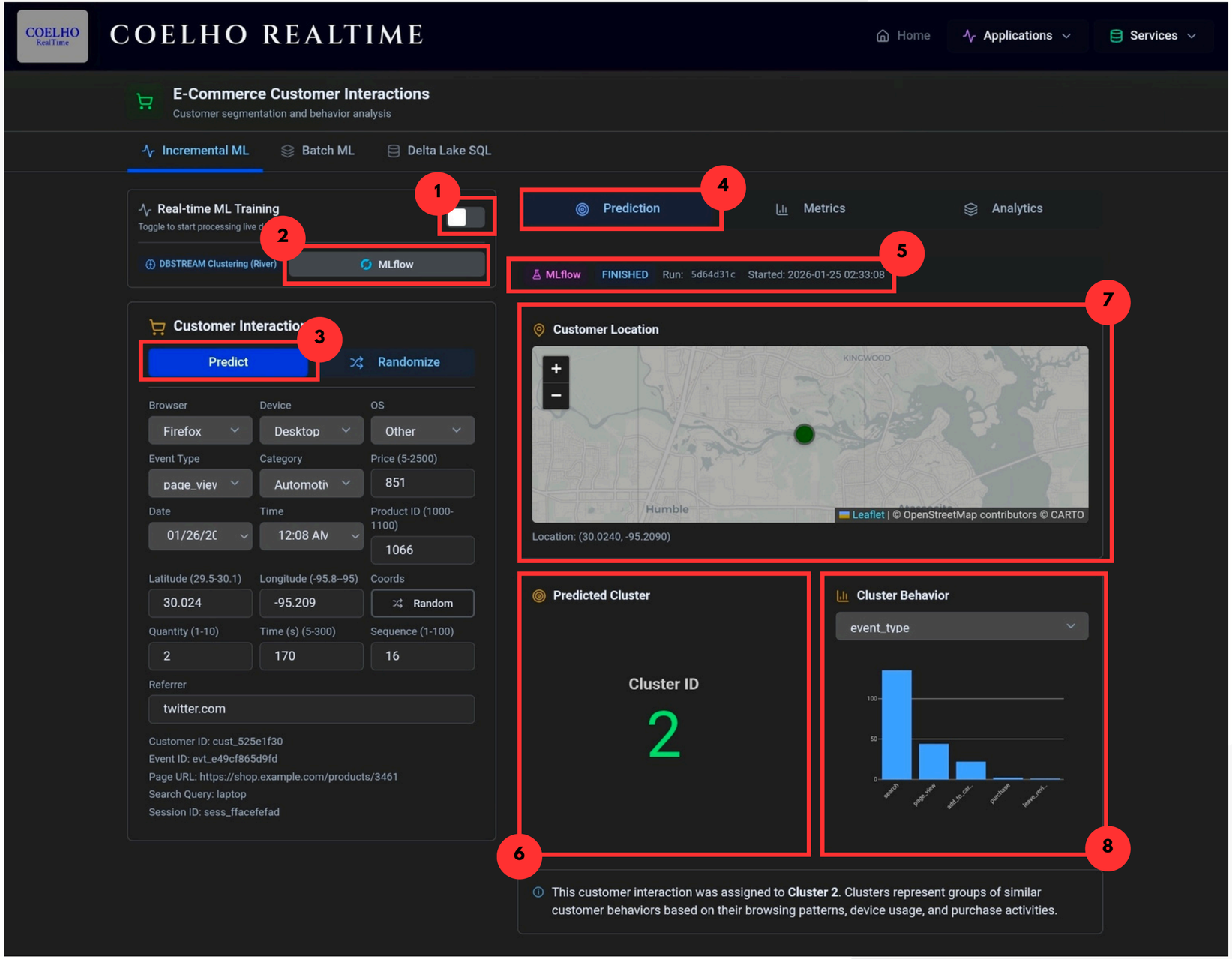
E-COMMERCE CUSTOMER INTERACTIONS

Incremental Machine Learning
Batch Machine Learning
Delta Lake SQL

- **Incremental ML model:** DBSTREAM (River)
- **Batch ML model:** KMEANS (Scikit-learn)

COELHO REALTIME

E-COMMERCE CUSTOMER INTERACTIONS - INCREMENTAL MACHINE LEARNING



1 Real-time ML training toggle

Activates or deactivates the continuous, real-time adaptation of the ECCI model, enabling it to learn incrementally from incoming data streams.

2 MLflow model page - experiment run

Provides a summary of a specific incremental learning experiment for ECCI on MLflow service page, detailing logged parameters, metrics, and associated artifacts within MLflow.

3 Prediction button

Initiates an immediate ECCI prediction using the ECCI model's current, incrementally updated state, demonstrating low-latency inference.

4 Prediction tab

Displays real-time ECCI predictions from the incremental machine learning model, offering immediate and continuously updated insights into customer interactions.

5 MLflow experiment run informations

Offers status info (LIVE | FINISHED), MLflow run ID, and the datetime the last real-time ML model training was executed.

6 Prediction results

Presents the output of the ECCI model's real-time predictions.

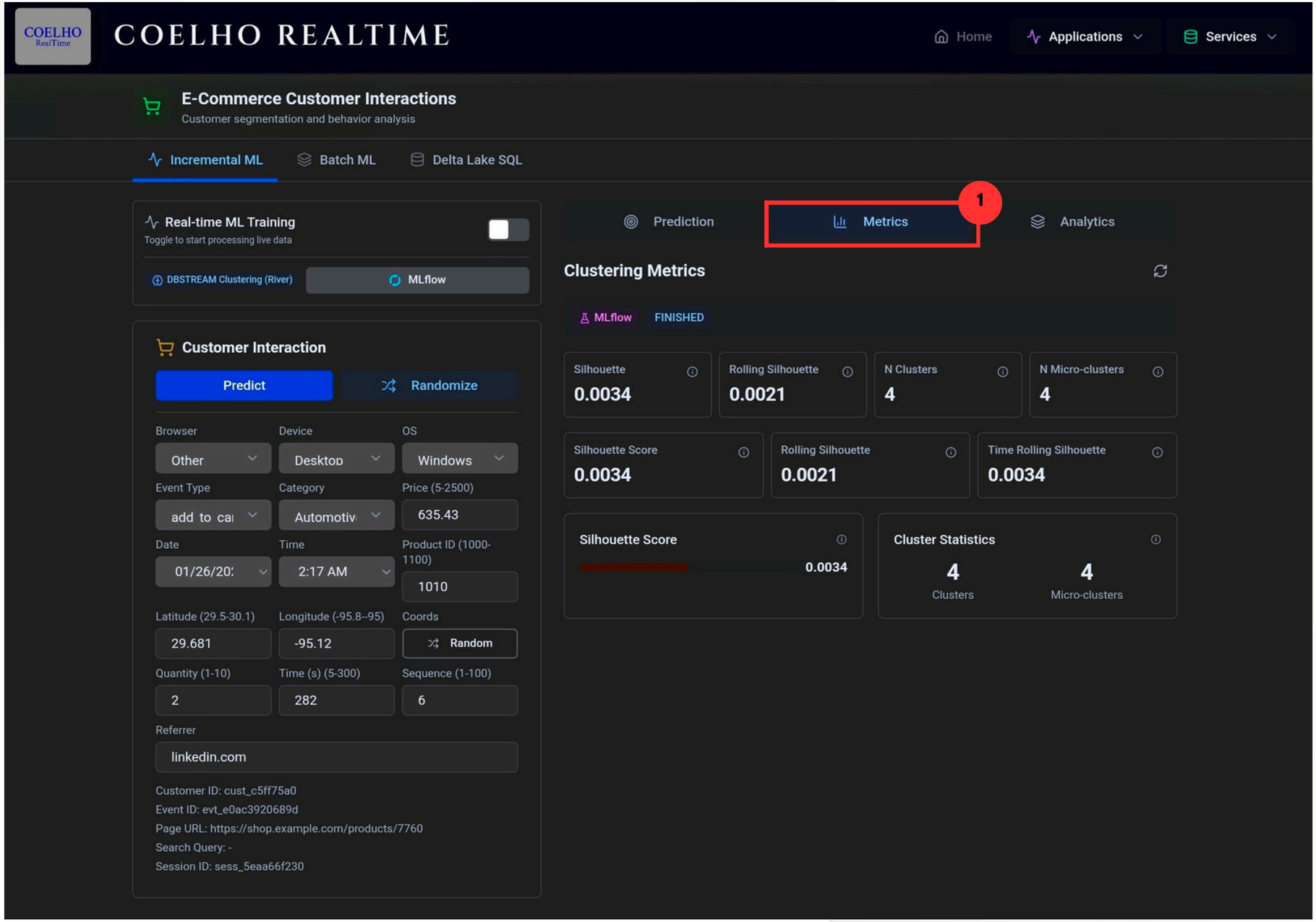
7 Map with customer location

Illustrates customer locations and their interaction journeys on a map, providing spatial insights into e-commerce customer behavior.

8 Cluster behavior analysis

Illustrates the distribution of a selected customer interaction feature (e.g., event type, product category) within the predicted cluster for the sample user, revealing the characteristic behaviors of that customer segment.

E-COMMERCE CUSTOMER INTERACTIONS - INCREMENTAL MACHINE LEARNING



1 Metrics tab

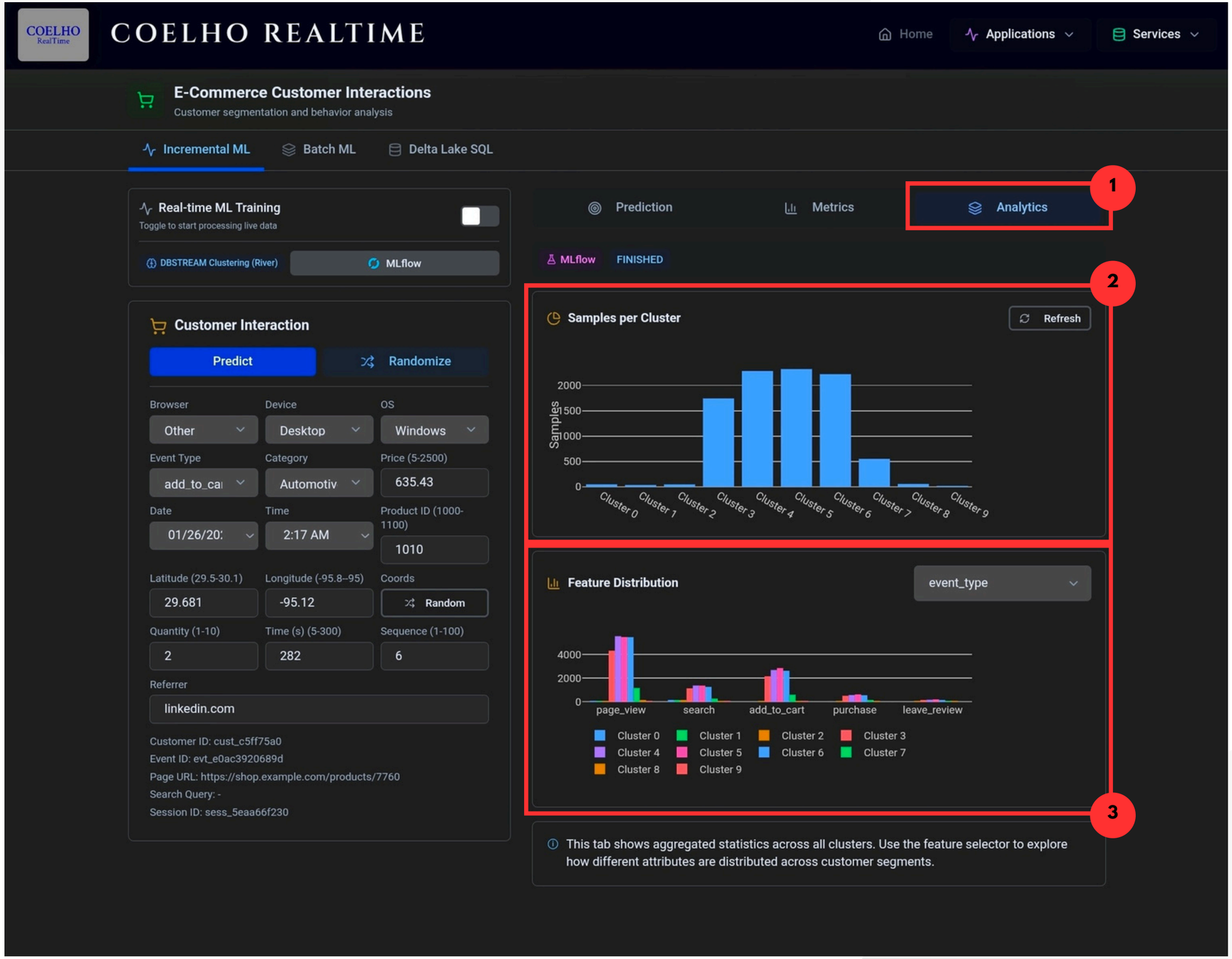
Presents the real-time metrics and incremental learning status of the ECCI model.

Metrics (River library)

- Silhouette
- Silhouette Score
- Rolling Silhouette
- Time Rolling Silhouette
- Number of Clusters
- Number of Micro-clusters

COELHO REALTIME

E-COMMERCE CUSTOMER INTERACTIONS - INCREMENTAL MACHINE LEARNING



1 Analytics tab

Provides an in-depth view of aggregated statistics across all identified customer clusters, allowing for comprehensive analysis of customer segments and their evolving characteristics.

2 Samples per cluster

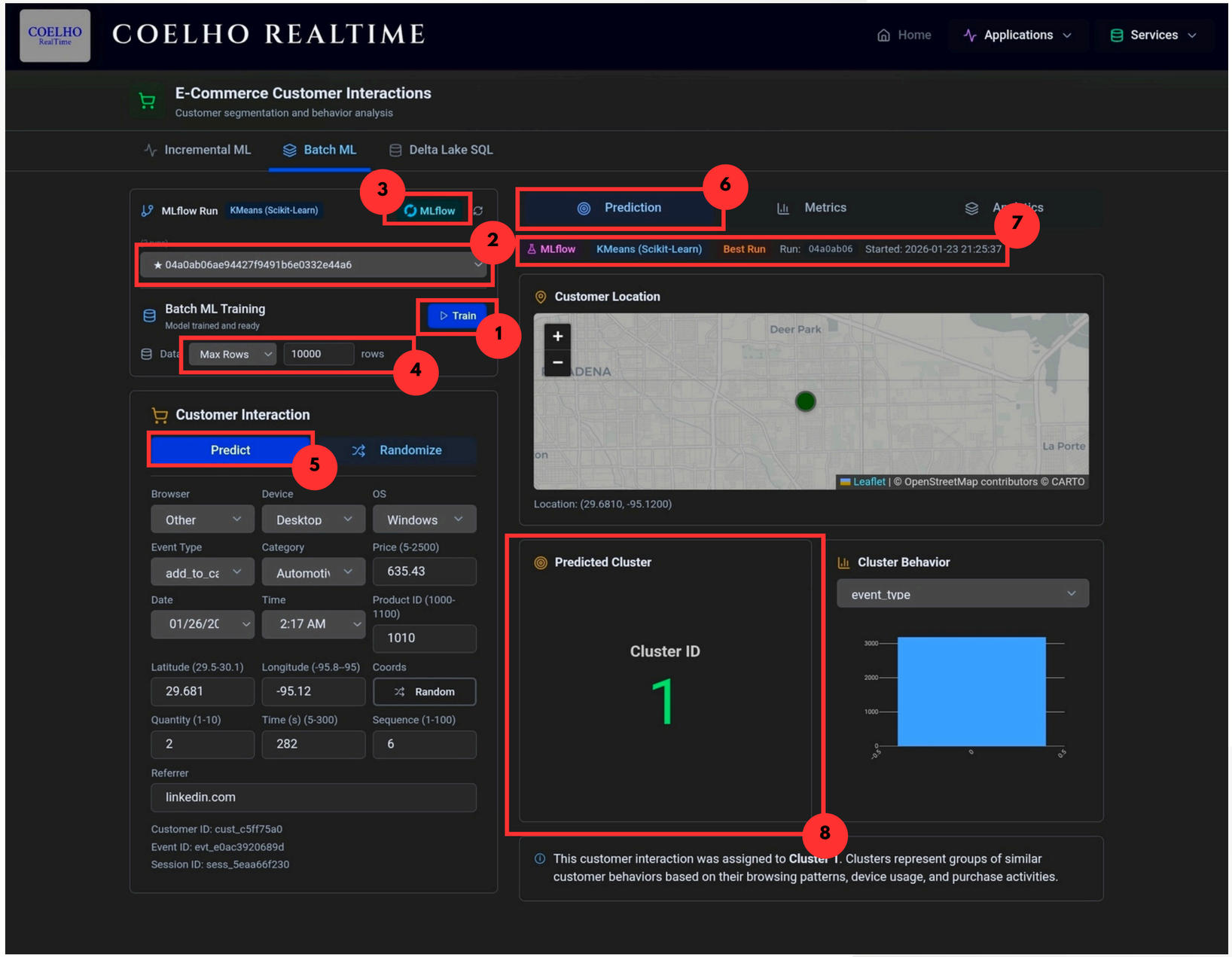
Visualizes the real-time distribution of customer interaction samples across different clusters, showing the relative size and density of each identified customer segment.

3 Feature Distribution

Displays the distribution of values for a selected feature (e.g., event type, product category) across all customer clusters, highlighting the distinguishing attributes and behavioral patterns of each segment.

COELHO REALTIME

E-COMMERCE CUSTOMER INTERACTIONS - BATCH MACHINE LEARNING



1 Train button

Initiates the batch machine learning training process for the ECCI model.

2 MLflow run ID - ML model

Dropdown displaying the MLflow run IDs associated with batch-trained ECCI model the user can pick after training new Batch ML models.

3 MLflow run ID page

Navigates to or provides detailed information within MLflow about a particular experiment run.

4 Batch ML Training options

Provides the user the options of training new Batch ML models in real-time per maximum number of rows or per percentage of rows from the original ECCI dataset.

5 Prediction button

Triggers a prediction using the currently selected or trained batch ML model.

6 Prediction tab

Displays the interface or section where batch prediction results for the ECCI model are presented.

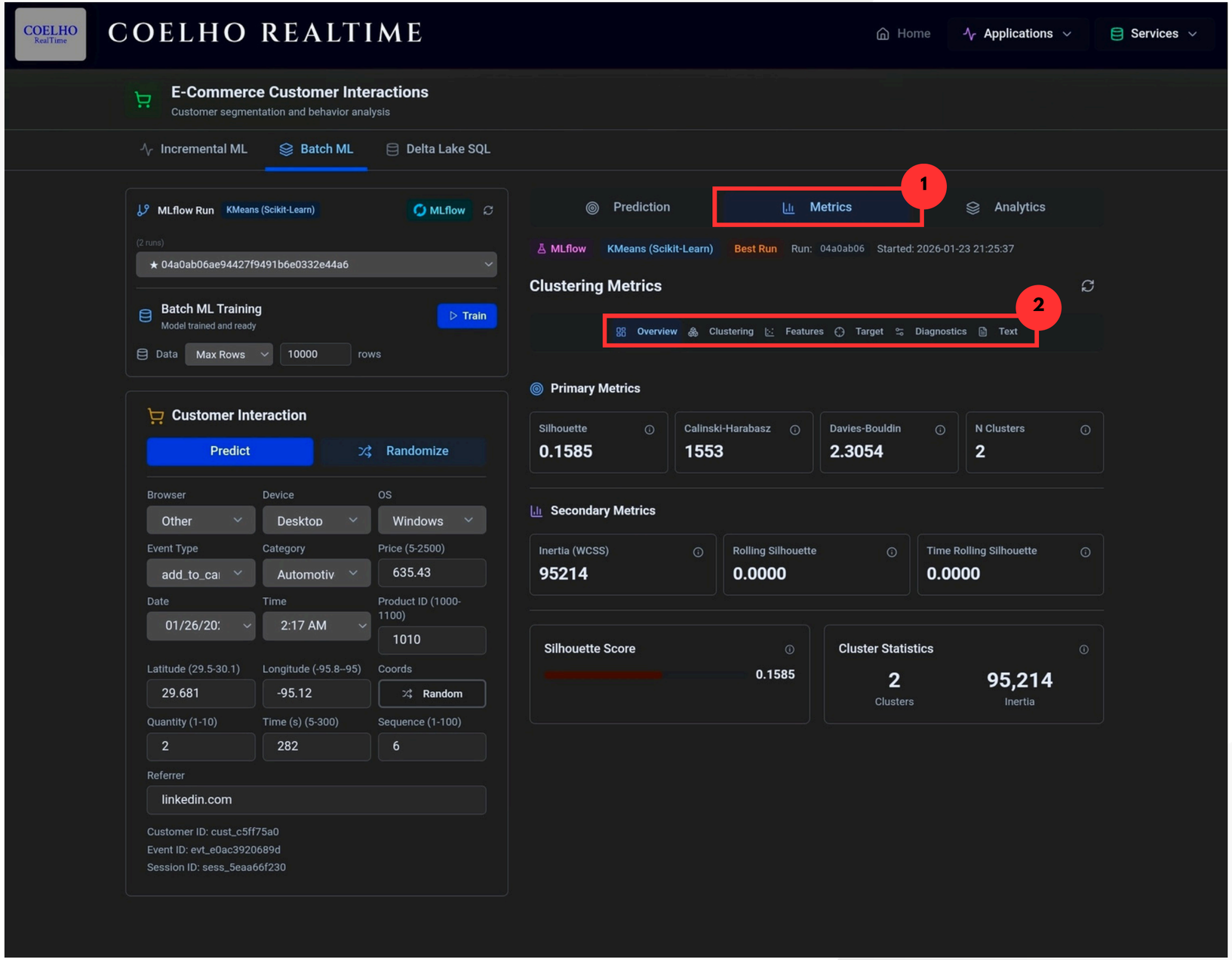
7 MLflow experiment run informations

Offers detailed status information, run ID, and other metadata for an MLflow experiment run.

8 Prediction results

Presents the output of the ECCI model's batch machine learning predictions.

E-COMMERCE CUSTOMER INTERACTIONS - BATCH MACHINE LEARNING



1 Metrics tab

Presents various metrics of the ECCI model. Includes some tabs with a lot of visual metrics from Scikit-Learn, Scikit-Plot and YellowBrick libraries.

2 Clustering metrics tabs

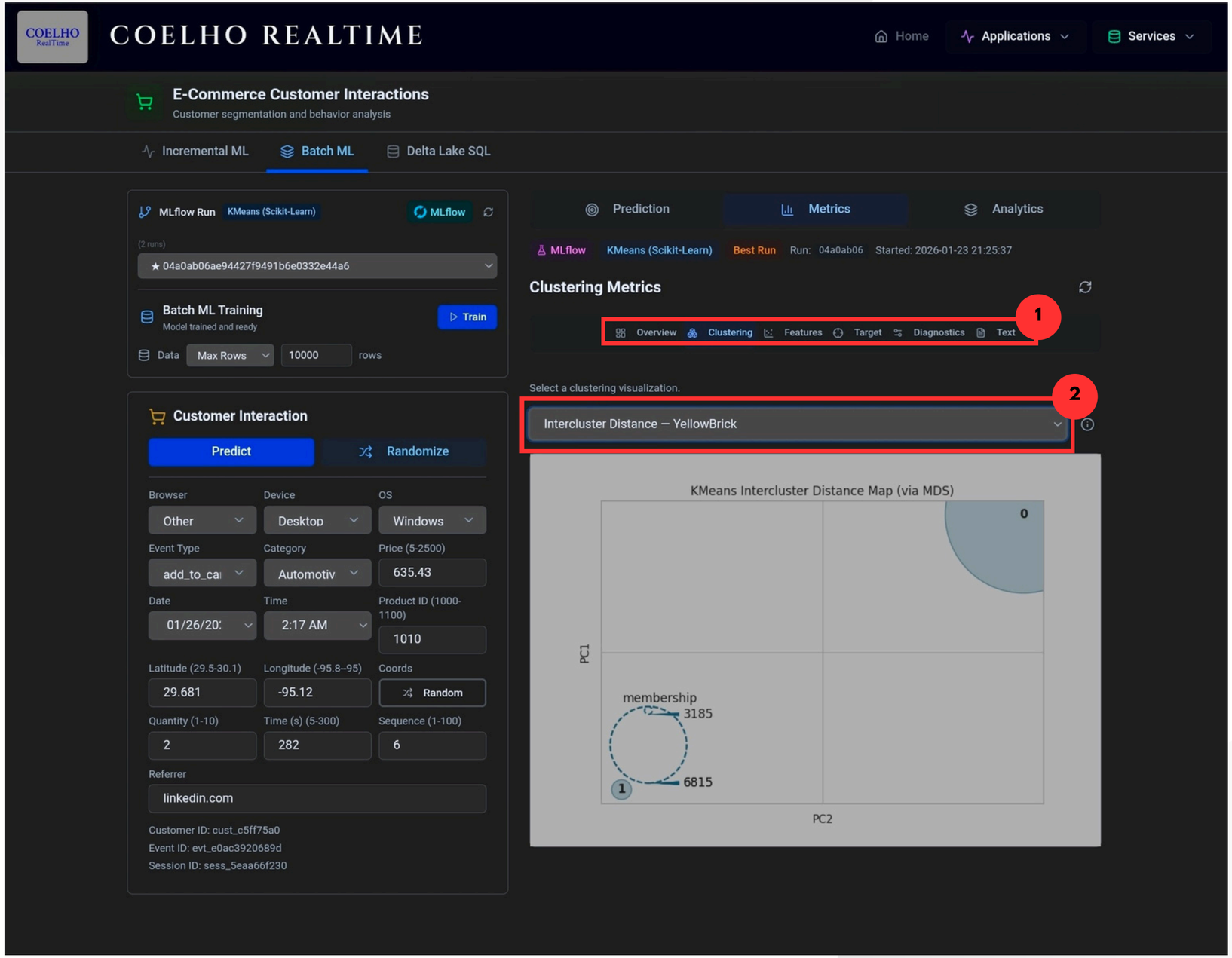
- **Overview:** Displays key MLflow metrics, including primary, ranking, secondary, and calibration scores.
- **Clustering:** Provides various visualizations to assess clustering model performance.
- **Features:** Offers visual tools for analyzing input features and their properties.
- **Target:** Presents visualizations related to the target variable's distribution and characteristics.
- **Diagnostics:** Contains visualizations for model selection, debugging, and advanced diagnostics.
- **Text:** Visualizations and metrics for Textual Features, specifically for the search_query field, including analyses such as word frequency distributions, search query clustering (t-SNE/UMAP), word dispersion, word correlation, and part-of-speech tagging, all aimed at understanding customer search behavior and intent.

Metrics (Scikit-Learn library)

- Silhouette
- Calinski-Harabasz
- Davies-Bouldin
- N Clusters
- Inertia (WCSS)
- Rolling Silhouette
- Time Rolling Silhouette

COELHO REALTIME

E-COMMERCE CUSTOMER INTERACTIONS - BATCH MACHINE LEARNING



1 Clustering Metrics tabs

Overview, Clustering, Features, Target, Diagnostics, Text

2 Clustering metrics tabs options

- **Clustering**
  - K-Elbow
  - Silhouette
  - Intercluster Distance
  - Elbow Curve
- **Features**
  - Rank 1D
  - Rank 2D
  - PCA Decomposition
  - Manifold
  - Parallel Coordinates
  - RadViz
  - Joint Plot
  - Partial Dependence
  - Decision Boundary
  - PCA 2D Projection
  - PCA Component Variance
  - Feature Importances
- **Target**
  - Cluster Distribution
  - Feature Correlation (Mutual Info)
  - Feature Correlation (Pearson)
  - Balanced Binning Reference
- **Diagnostics**
  - Feature Importances
  - Cross-Validation Scores
  - Validation Curve
  - Learning Curve
  - Recursive Feature Elimination
  - Dropping Curve
- **Text**
  - Frequency Distribution
  - t-SNE Visualization
  - UMAP Visualization
  - Dispersion Plot
  - Word Correlation
  - POS Tag Distribution

COELHO REALTIME

E-COMMERCE CUSTOMER INTERACTIONS - DELTA LAKE SQL

1 SQL Editor

Interface for writing and editing SQL queries against the ECCI Delta Lake.

2 Query run button

Executes the SQL query in the editor against the ECCI Delta Lake.

3 Search box, rows and latency

Displays search, allowing user to filter specific values over the entire result, total rows returned, and query execution time.

4 Query results with filter

Presents the results of the SQL query with values filtered on the filter box.

COELHO REALTIME

THE MICROSERVICES ENGINE

A closer look at the microservices powering the COELHO RealTime platform — from real-time observability with Prometheus and Grafana, to experiment tracking with MLflow, event streaming with Apache Kafka, and the complete infrastructure that keeps every component running seamlessly on Kubernetes.

FastAPI

Spark

MLflow

MinIO


Prometheus

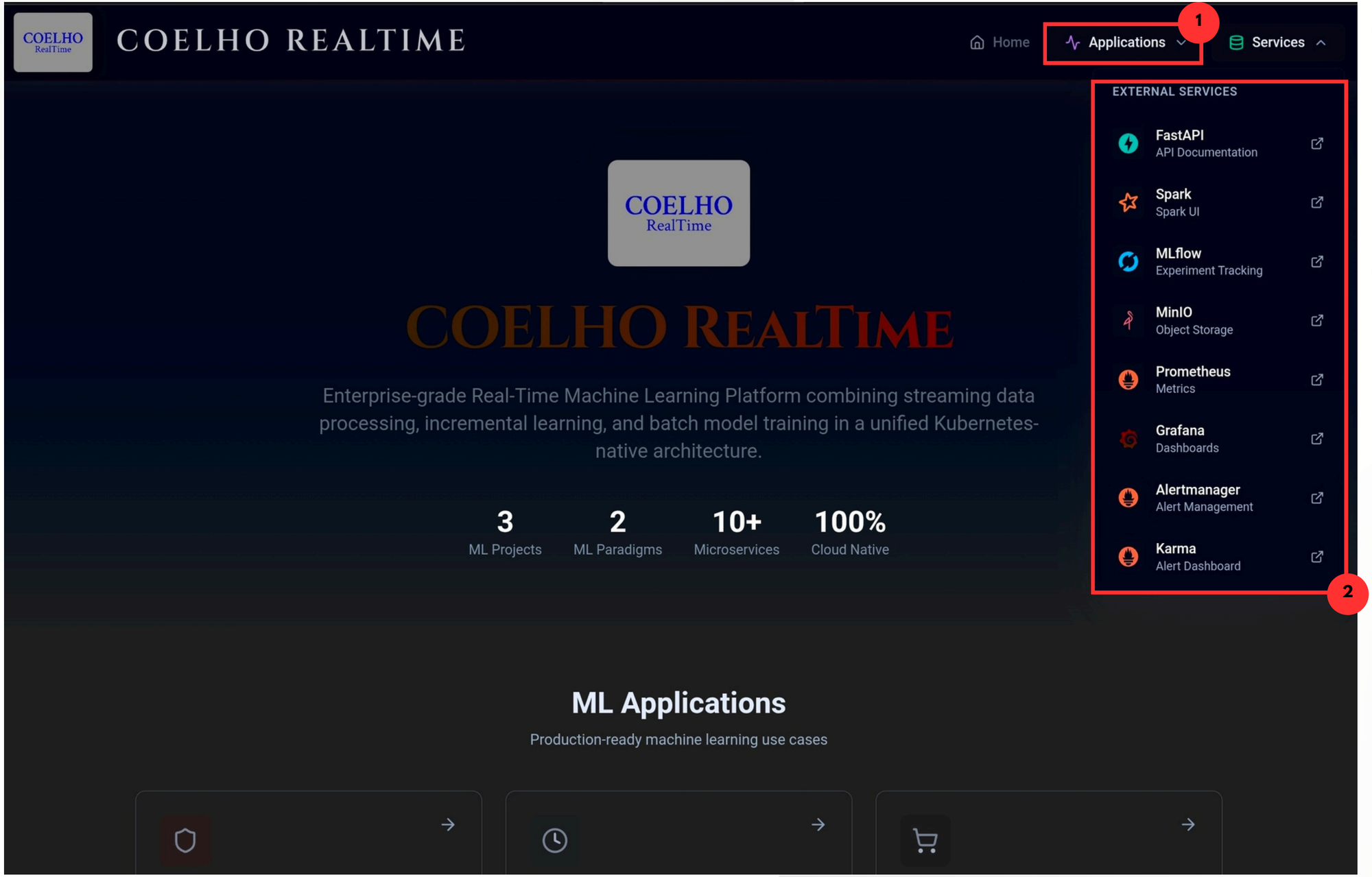
Grafana


Alertmanager


Karma

COELHO
REALTIME

MICROSERVICES - HOW TO ACCESS



1 Services dropdown on navbar

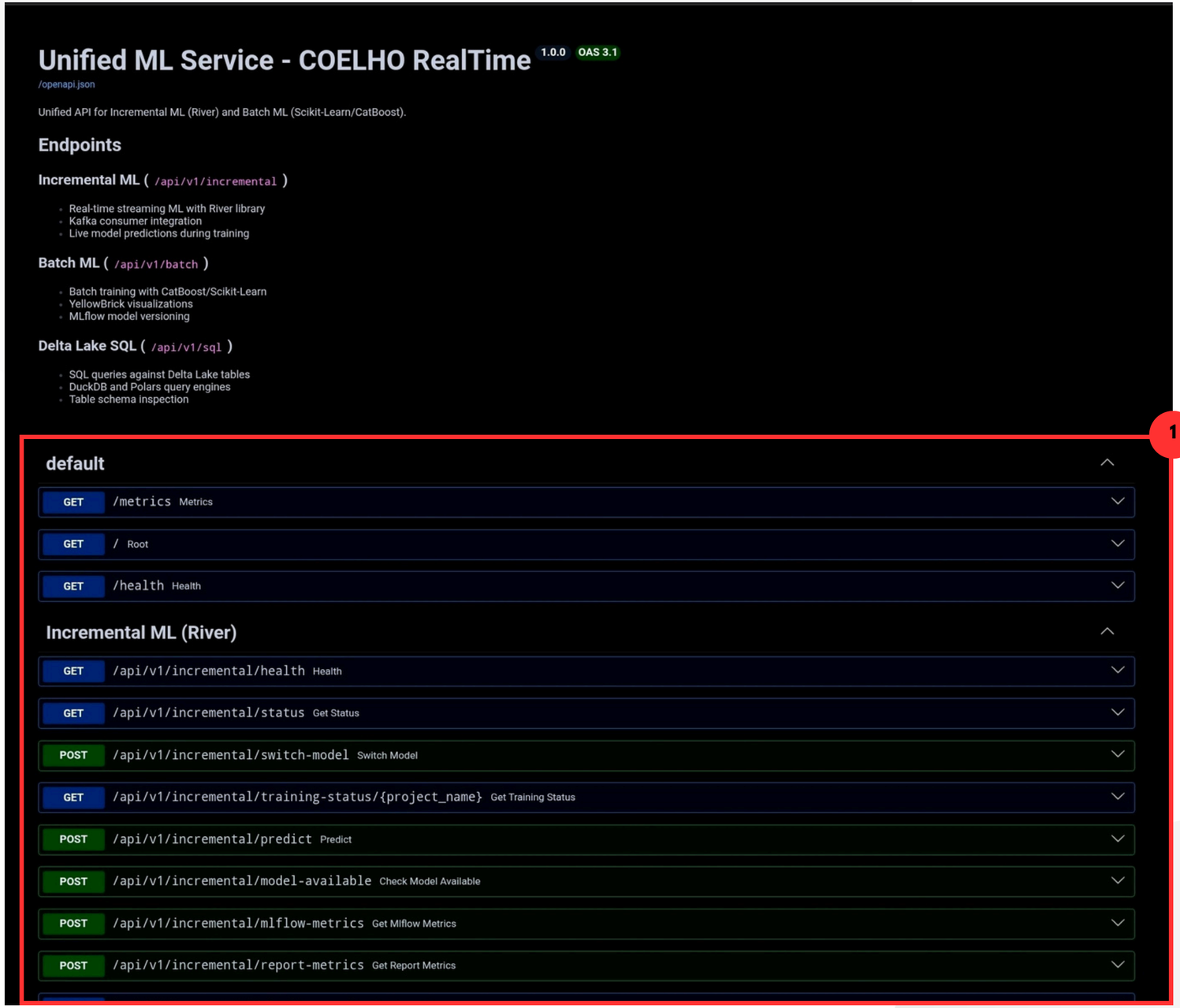
The Services dropdown on the navbar provides quick access to all external microservices running on the platform. Clicking it reveals a menu listing 7 services — FastAPI, Spark, MLflow, MinIO, Prometheus, Grafana, and Alertmanager — each linking directly to its web UI in a new browser tab.

2 Microservices options

- **FastAPI** — localhost:8000/docs
- **Spark** — localhost:4040
- **MLflow** — localhost:5001
- **MinIO** — localhost:9001
- **Prometheus** — localhost:9090
- **Grafana** — localhost:3000
- **Alertmanager** — localhost:9094
- **Karma:** localhost:8088

COELHO REALTIME

MICROSERVICES - FASTAPI



1 FastAPI — Important REST API Endpoints

Incremental ML (/api/v1/incremental) — 16 endpoints

- [GET] /health — Health check
- [GET] /status — Current training status
- [POST] /switch-model — Start/stop incremental training
- [POST] /predict — Real-time prediction (live Redis or MLflow model)
- [POST] /mlflow-metrics — Experiment metrics with caching
- [POST] /page-init — Combined page initialization
- [POST] /sample — Random dataset sample via DuckDB

Batch ML (/api/v1/batch) — 20 endpoints

- [GET] /health — Health check
- [GET] /status — Training progress with live updates
- [POST] /switch-model — Start/stop batch training
- [POST] /predict — Prediction using MLflow models
- [POST] /mlflow-runs — List all experiment runs
- [POST] /init — Page initialization with all data
- [POST] /visualization/metric/yellowbrick — YellowBrick visualizations
- [POST] /visualization/metric/sklearn — Scikit-Learn visualizations
- [POST] /visualization/metric/scikitplot — Scikit-plot visualizations

Delta Lake SQL (/api/v1/sql) — 3 endpoints

- [POST] /query — Execute SQL against Delta Lake (SELECT only)
- [POST] /schema — Table schema and metadata
- [POST] /total-rows — Total row count

MICROSERVICES - SPARK

Spark Master at spark://coelho-realtime-spark-master-0.coelho-realtime-spark-headless.coelho-realtime.svc.cluster.local:7077

URL: spark://coelho-realtime-spark-master-0.coelho-realtime-spark-headless.coelho-realtime.svc.cluster.local:7077
Workers: 1 Alive, 0 Dead, 0 Decommissioned, 0 Unknown
Cores in use: 8 Total, 6 Used
Memory in use: 3.0 GiB Total, 1536.0 MiB Used
Resources in use:
Applications: 3 Running, 24 Completed
Drivers: 0 Running (0 Waiting), 0 Completed (0 Killed, 0 Failed, 0 Error, 0 Relaunching)
Status: ALIVE (Environment, Log)

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20260126235202-10.42.0.106-45349	10.42.0.106:45349	ALIVE	8 (6 Used)	3.0 GiB (1536.0 MiB Used)

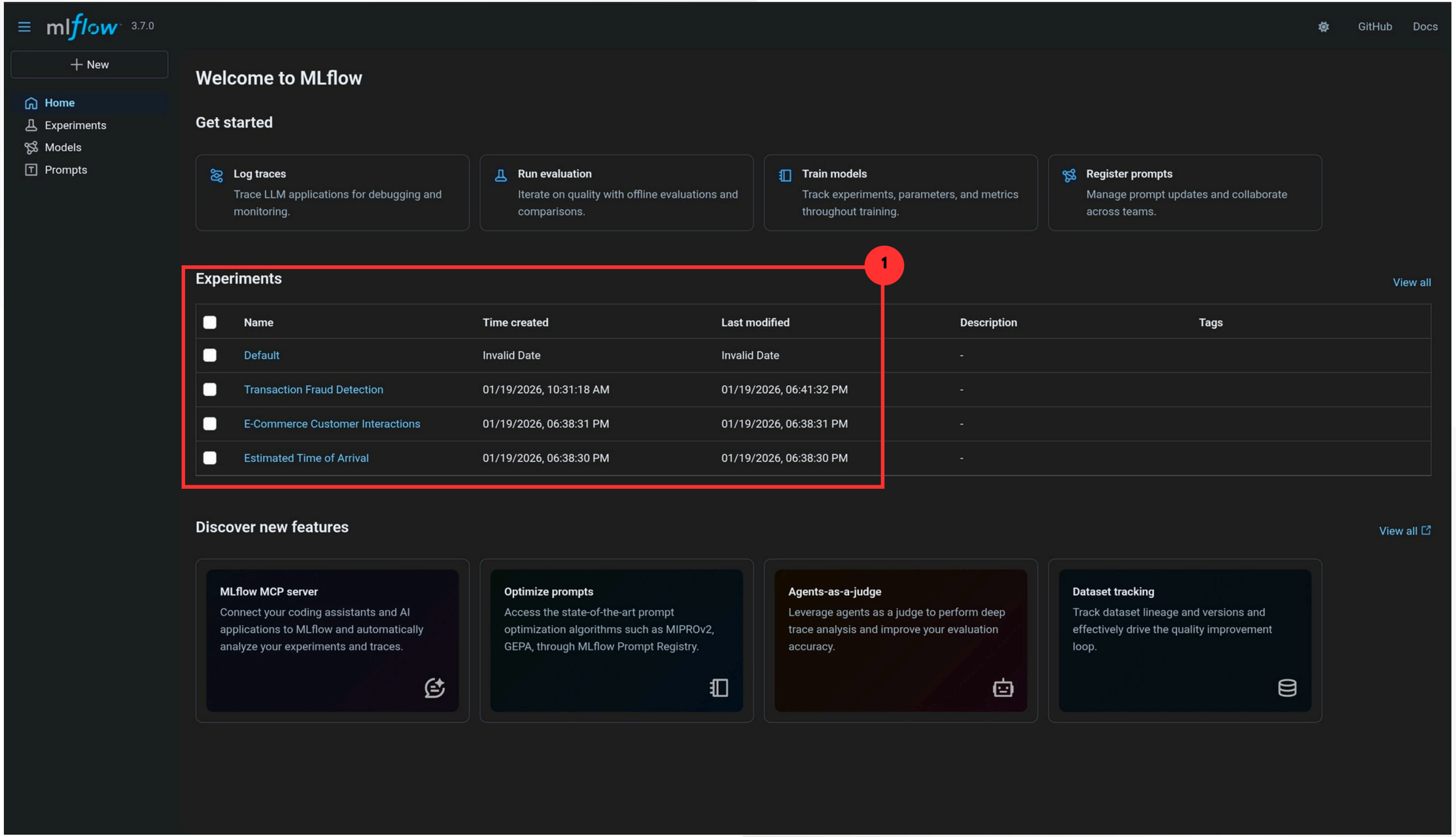
Running Applications (3)

Application ID		Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20260127023438-0026	(kill)	KafkaToDelta-ECommerceCustomerInteractions	2	512.0 MiB		2026/01/27 02:34:38	root	RUNNING	8.5 min
app-20260127023438-0025	(kill)	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/27 02:34:38	root	RUNNING	8.5 min
app-20260127023438-0024	(kill)	KafkaToDelta-TransactionFraudDetection	2	512.0 MiB		2026/01/27 02:34:38	root	RUNNING	8.5 min

Completed Applications (24)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20260127021321-0021	KafkaToDelta-ECommerceCustomerInteractions	2	512.0 MiB		2026/01/27 02:13:21	root	FINISHED	20 min
app-20260127021322-0022	KafkaToDelta-TransactionFraudDetection	2	512.0 MiB		2026/01/27 02:13:22	root	FINISHED	20 min
app-20260127021322-0023	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/27 02:13:22	root	FINISHED	20 min
app-20260127015433-0020	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/27 01:54:33	root	FINISHED	18 min
app-20260127015433-0018	KafkaToDelta-TransactionFraudDetection	2	512.0 MiB		2026/01/27 01:54:33	root	FINISHED	18 min
app-20260127015433-0019	KafkaToDelta-ECommerceCustomerInteractions	2	512.0 MiB		2026/01/27 01:54:33	root	FINISHED	18 min
app-20260127013721-0016	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/27 01:37:21	root	FINISHED	16 min
app-20260127013721-0015	KafkaToDelta-TransactionFraudDetection	2	512.0 MiB		2026/01/27 01:37:21	root	FINISHED	16 min
app-20260127013722-0017	KafkaToDelta-ECommerceCustomerInteractions	2	512.0 MiB		2026/01/27 01:37:22	root	FINISHED	16 min
app-20260127011310-0013	KafkaToDelta-ECommerceCustomerInteractions	2	512.0 MiB		2026/01/27 01:13:10	root	FINISHED	23 min
app-20260127011310-0012	KafkaToDelta-TransactionFraudDetection	2	512.0 MiB		2026/01/27 01:13:10	root	FINISHED	23 min
app-20260127011310-0014	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/27 01:13:10	root	FINISHED	23 min
app-20260127005700-0009	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/27 00:57:00	root	FINISHED	15 min
app-20260127005701-0011	KafkaToDelta-TransactionFraudDetection	2	512.0 MiB		2026/01/27 00:57:01	root	FINISHED	15 min
app-20260127005701-0010	KafkaToDelta-ECommerceCustomerInteractions	2	512.0 MiB		2026/01/27 00:57:01	root	FINISHED	15 min
app-20260127003704-0007	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/27 00:37:04	root	FINISHED	19 min
app-20260127003704-0006	KafkaToDelta-ECommerceCustomerInteractions	2	512.0 MiB		2026/01/27 00:37:04	root	FINISHED	19 min
app-20260127003704-0008	KafkaToDelta-TransactionFraudDetection	2	512.0 MiB		2026/01/27 00:37:04	root	FINISHED	19 min
app-20260127001818-0004	KafkaToDelta-ECommerceCustomerInteractions	2	512.0 MiB		2026/01/27 00:18:18	root	FINISHED	18 min
app-20260127001817-0003	KafkaToDelta-TransactionFraudDetection	2	512.0 MiB		2026/01/27 00:18:17	root	FINISHED	18 min
app-20260127001818-0005	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/27 00:18:18	root	FINISHED	18 min
app-20260126235417-0002	KafkaToDelta-EstimatedTimeOfArrival	2	512.0 MiB		2026/01/26 23:54:17	root	FINISHED	23 min

MICROSERVICES - MLFLOW



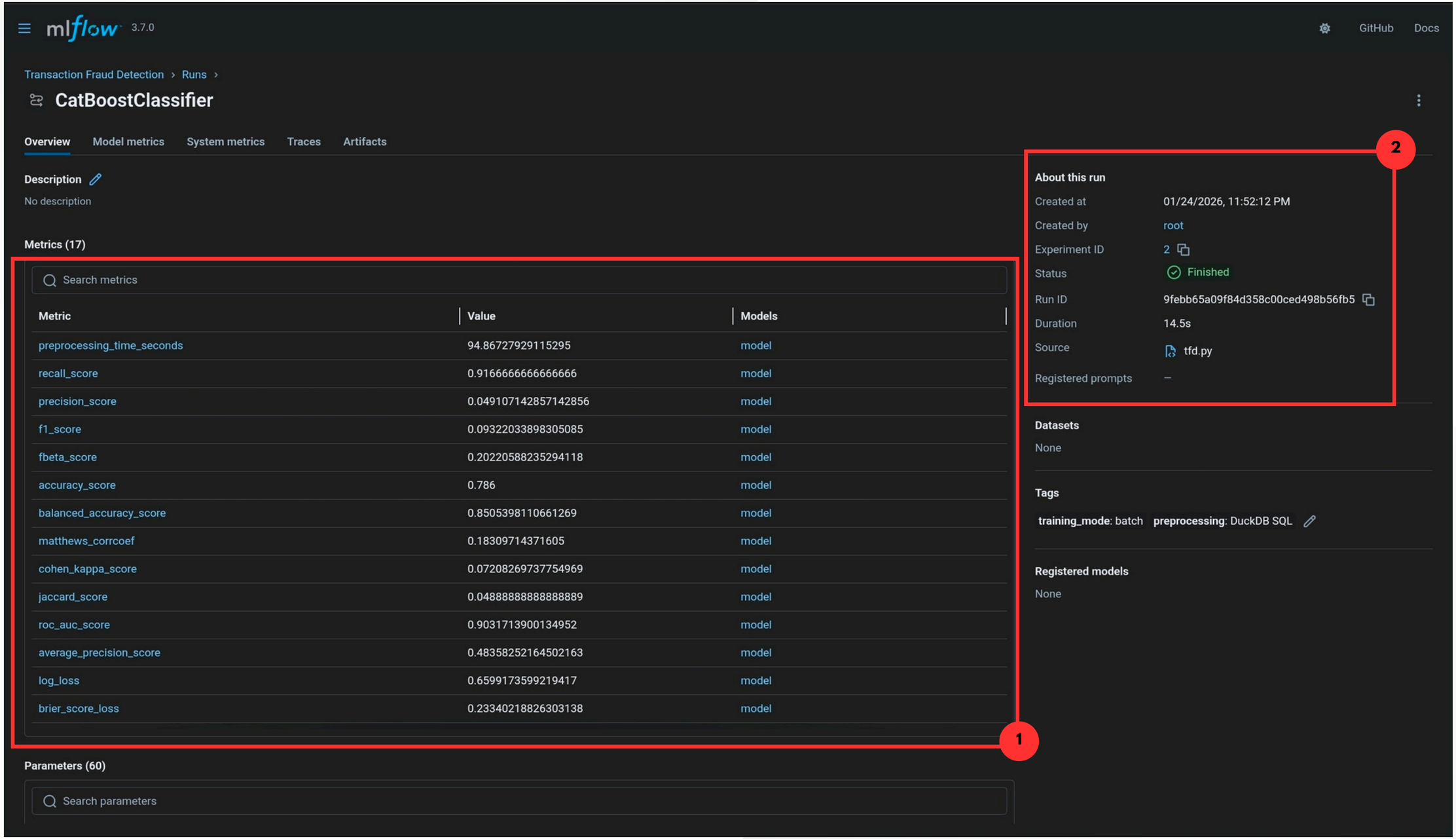
1

MLflow experiments

Each ML project maps to a dedicated MLflow experiment: Transaction Fraud Detection (binary classification), Estimated Time of Arrival (regression), and E-Commerce Customer Interactions (clustering).

Both Incremental (River) and Batch (Scikit-Learn/CatBoost) training runs are logged under their respective experiment, tracking parameters, metrics, and artifacts across model versions.

MICROSERVICES - MLFLOW



1

MLflow experiment run metrics

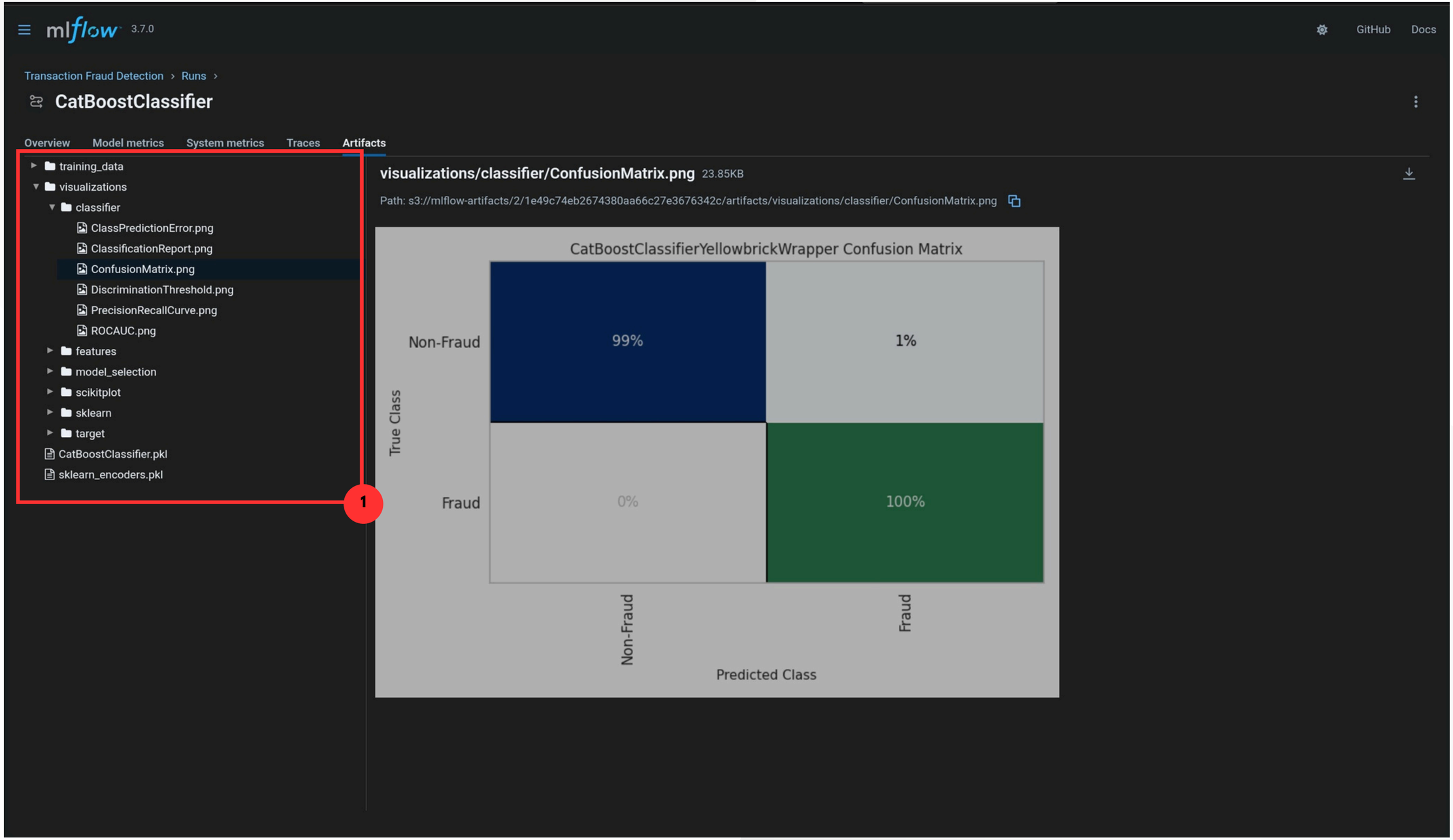
Metrics are logged in real time during training and persist after completion.
For classification (TFD): F-beta, accuracy, precision, recall, F1 score.
For regression (ETA): RMSE, MAE, R^2.
For clustering (ECCI): Silhouette score, inertia, number of clusters.

These metrics enable comparison across runs and determine which model is selected for predictions.

2

MLflow experiment run info

MICROSERVICES - MLFLOW



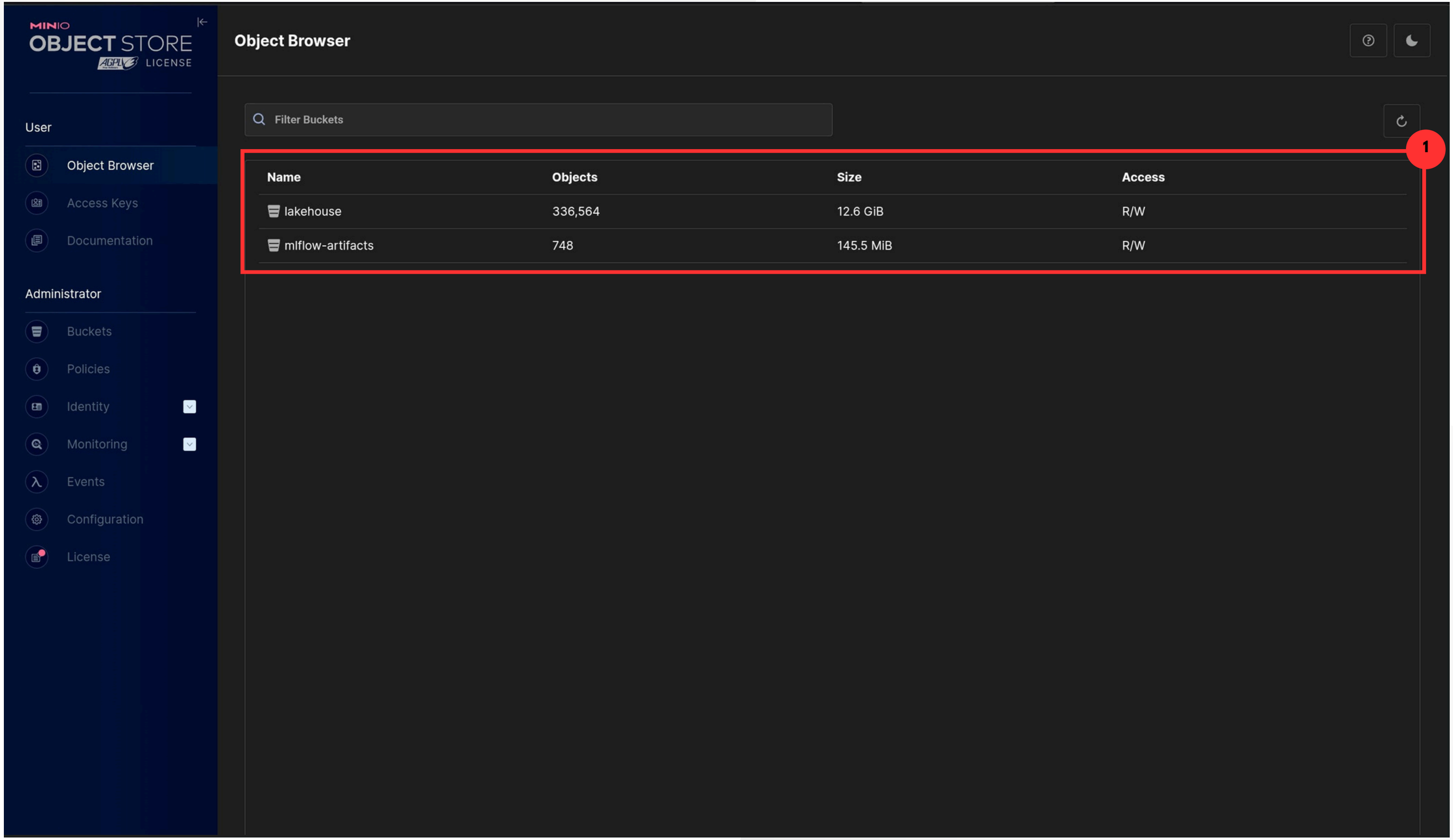
1

MLflow experiment run artifacts

Each run stores the trained model (pickle), feature encoders, training data (Parquet), and visualizations generated by YellowBrick, Scikit-Learn, and Scikit-plot (e.g., confusion matrix, ROC curve, residuals plot, silhouette analysis).

Artifacts are saved to MinIO (S3-compatible storage) and loaded on demand for predictions, model comparison, and visualization caching.

MICROSERVICES - MINIO



1

MinIO buckets

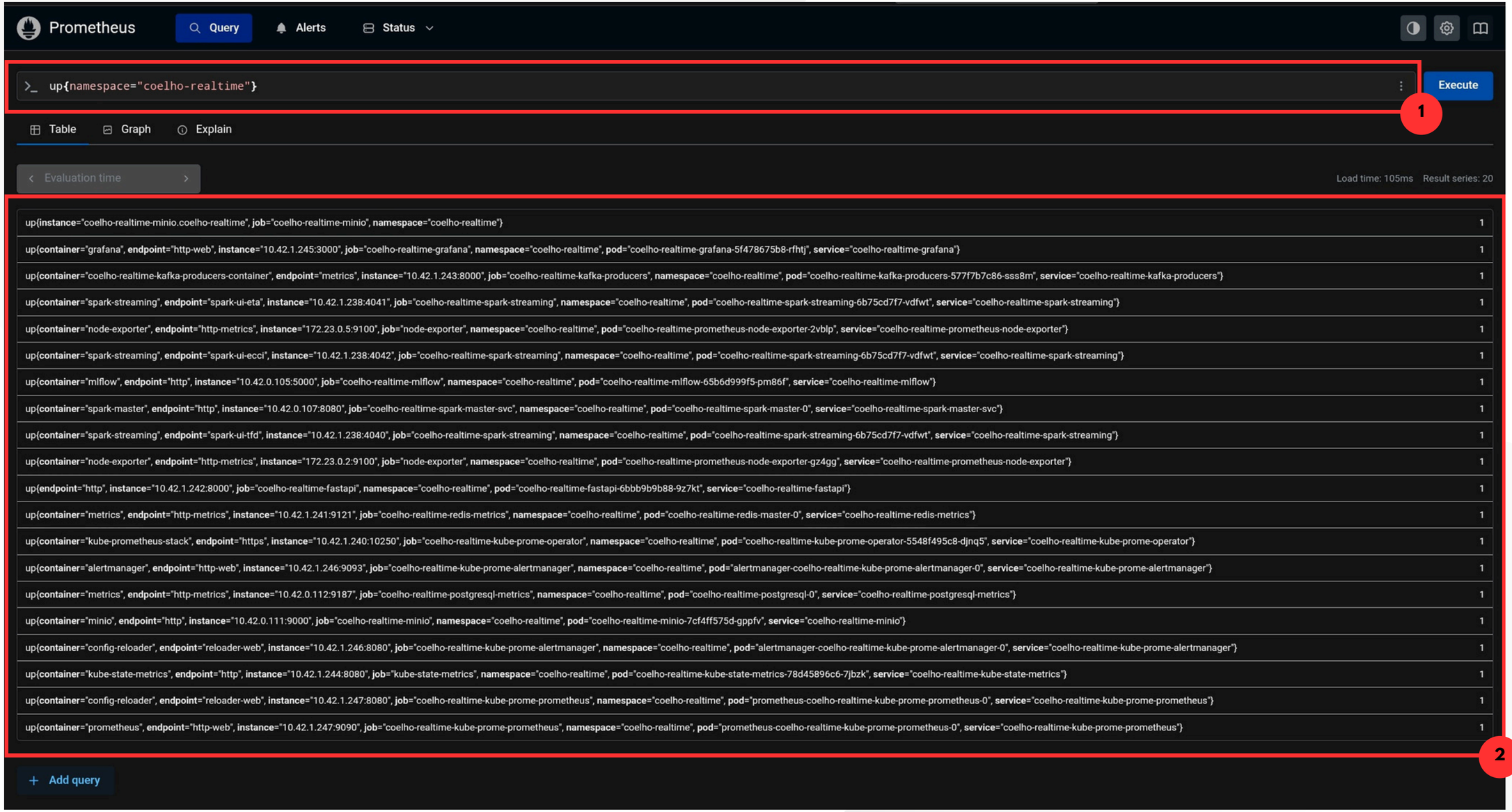
MinIO serves as the unified storage backend for the platform.

The mlflow-artifacts bucket stores all MLflow experiment data — trained models, encoders, training datasets, and visualizations.

The lakehouse bucket holds the Delta Lake tables consumed by Apache Spark and DuckDB, organized by project (TFD, ETA, ECCI).

Both buckets are accessed via the S3 API, making the setup portable to any S3-compatible cloud provider.

MICROSERVICES - PROMETHEUS



1

PromQL query

Prometheus Query Language expression executed against the time-series database.

The up metric is a built-in indicator that returns 1 if a target is reachable and 0 if it is down, serving as the foundation for all service health alerts in the platform.

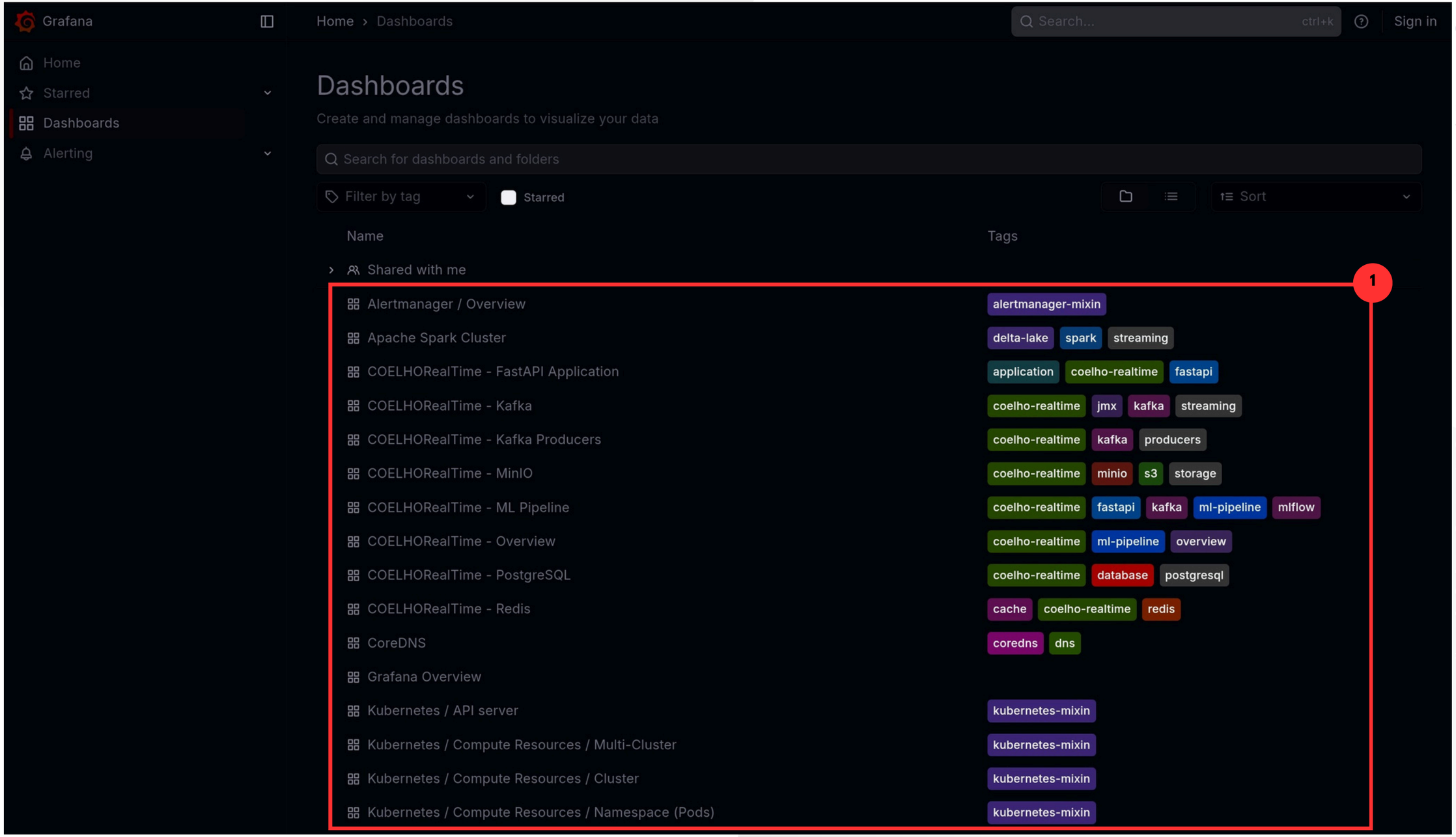
2

PromQL query results

The query returns 20 monitored targets across the platform: FastAPI, Kafka Producers, Spark (Master + 3 Streaming jobs), MLflow, MinIO, Redis, PostgreSQL, Grafana, Alertmanager, Prometheus, Kube State Metrics, and Node Exporters — all reporting up = 1 (healthy).

These results feed the alerting rules that trigger notifications when any service goes down.

MICROSERVICES - GRAFANA

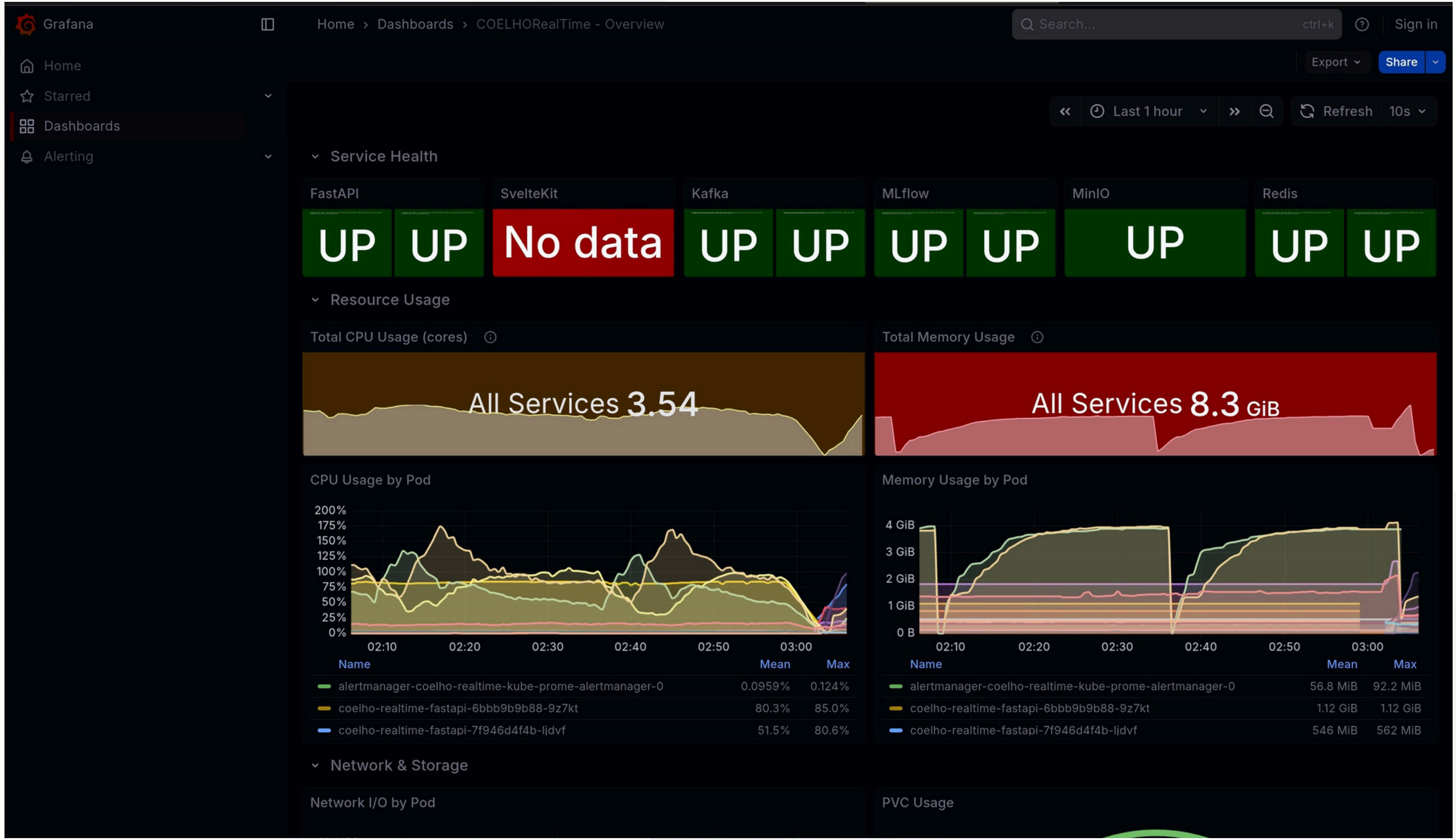


1

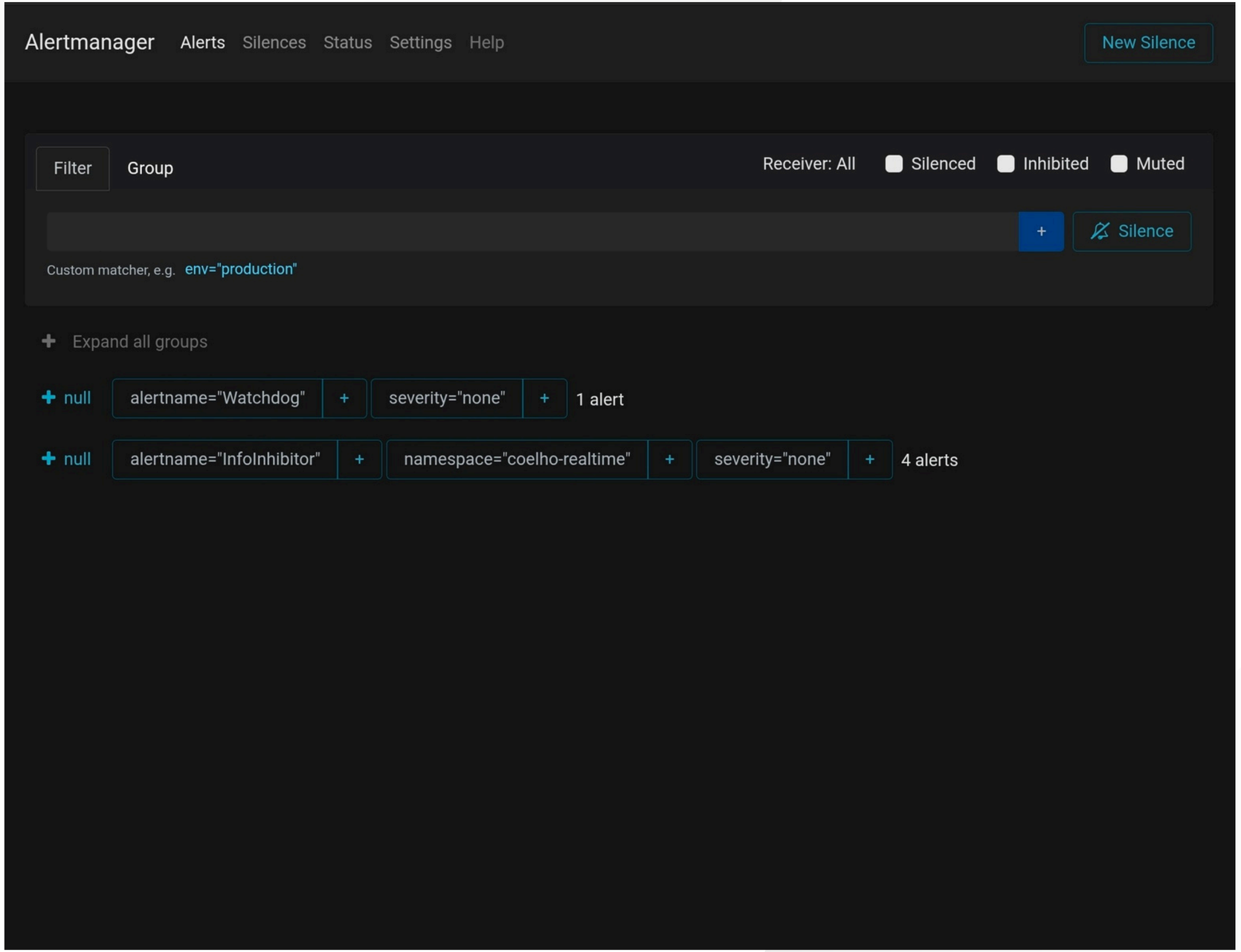
Grafana dashboards

The platform ships with 10 pre-built dashboards, automatically provisioned via Kubernetes ConfigMaps: COELHO RealTime Overview (platform-wide health), ML Pipeline (training and prediction metrics), FastAPI Detailed (API latency, throughput, errors), Kafka Producers (message rates, send latency), Kafka (broker health, partitions), Spark (master and workers), Spark Streaming (batch processing, records ingested), PostgreSQL (connections, deadlocks), Redis (memory, commands), and MinIO (storage, requests). All dashboards query Prometheus as their data source and update in real time.

MICROSERVICES - GRAFANA



MICROSERVICES - ALERTMANAGER



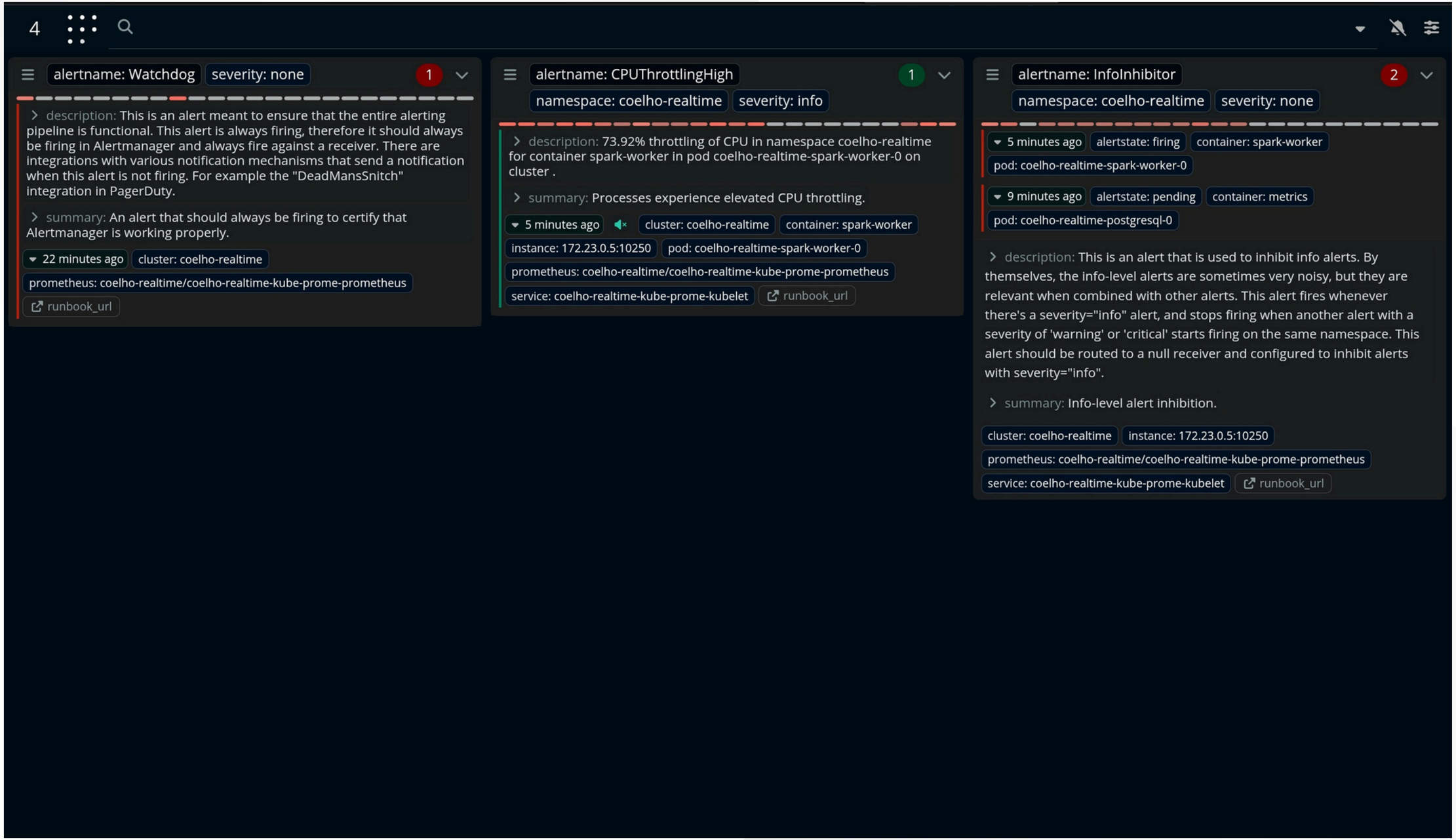
Alertmanager receives firing alerts from Prometheus and handles routing, grouping, deduplication, and silencing.

Alerts are grouped by alertname, namespace, and severity, with critical alerts dispatched within 10 seconds and warnings batched at 10-minute intervals.

Inhibition rules prevent cascading noise — if Kafka goes down, all downstream Kafka alerts are suppressed; if PostgreSQL is unavailable, MLflow alerts are silenced.

The routing tree supports Slack, PagerDuty, Discord, email, and webhook receivers, all pre-configured and ready to enable.

MICROSERVICES - KARMA



Karma is a lightweight Alertmanager dashboard that provides real-time visibility into all firing, suppressed, and silenced alerts.

It connects to Alertmanager and offers label-based filtering, color-coded severity grouping, autocomplete search, and one-click silence creation — features that go beyond Alertmanager's built-in UI.

Karma enables fast alert triage during incidents by aggregating and deduplicating alerts across sources into a single overview.

COELHO REALTIME IS AN END-TO-END, CLOUD-NATIVE MACHINE LEARNING PLATFORM THAT UNIFIES REAL-TIME STREAMING, INCREMENTAL AND BATCH TRAINING, AND FULL OBSERVABILITY INTO A SINGLE KUBERNETES-NATIVE ARCHITECTURE — FROM KAFKA INGESTION TO MODEL SERVING, FULLY AUTOMATED AND PRODUCTION-READY.

LET'S WORK TOGETHER

https://rafaelcoelho1409.github.io/

rafaelcoelho1409@hotmail.com
coelhorafael@proton.me