NETCELL EDA PLATFORM

Feature Store And Pipeline Architecture (Enterprise Ready)

NETCELL-EDA Architecture

Feature Store Architecture (Enterprise Ready)

What the Feature Store actually does

It solves three problems:

Consistency Training and inference use the *same* features.

Reusability Multiple models (risk, churn, engagement) can use the same features.

Scalability Features are stored once, reused many times.

NETCELL-EDA Architecture

Feature Store Architecture (Enterprise Ready)

1. Online Feature Store (real-time)

Used by the Inference Service.

Fast reads (ms latency)

Stores the latest features per entity

Lives in:

- Redis
- MongoDB
- Cassandra
- or Cloudera's operational DB layer

NETCELL-EDA Architecture

Feature Store Architecture (Enterprise Ready)

2. Offline Feature Store (batch)

Used for model training.

Large historical datasets

Stored in Cloudera Data Lake (Iceberg/Hive)

Supports:

- joins
- aggregations
- time-window features
- backfills
- Purpose: Train models on months/years of behavior.

NETCELL-EDA Architecture

Feature Store Architecture (Enterprise Ready)

3. Feature Registry

The “catalog” of all features.

Defines feature names, types, owners, versions

Ensures consistency across services

Prevents duplication

Purpose: Governance + discoverability.

NETCELL-EDA Architecture

Feature Store Architecture (Enterprise Ready)

A. Online Feature Store (Document DB or Redis)

Key: TenantId + ContactId

Document example:

json


```
{
  "TenantId": "tenantA",
  "ContactId": 12345,
  "Features": {
    "state_change_count_last_30_days": 4,
    "time_since_last_state_change_hours": 2.5,
    "total_logins_last_7_days": 12,
    "billing_incidents_last_90_days": 0,
    "security_incidents_last_90_days": 1,
    "avg_session_duration_minutes": 14.2,
    "days_since_last_payment": 32
  },
  "UpdatedAt": "2026-08-04T19:40:00Z"
}
```


Characteristics:

Updated on every event

Read by Inference Service

Low latency

Small footprint

NETCELL-EDA Architecture

Feature Store Architecture (Enterprise Ready)

B. Offline Feature Store (Cloudera Iceberg/Hive)

Table: contact_features_daily

Partition: TenantId, Date

Characteristics:

Used for training

Supports backfills

Supports historical analysis

Can be joined with other tables

Column	Type	Description
TenantId	string	Tenant
ContactId	int	Entity
Date	date	Feature snapshot date
state_change_count_last_30_days	int	Rolling window
total_logins_last_7_days	int	Rolling window
billing_incidents_last_90_days	int	Rolling window
security_incidents_last_90_days	int	Rolling window
avg_session_duration_minutes	float	Behavior metric
days_since_last_payment	int	Billing metric
label_churn	int	Training label
label_risk	float	Training label

NETCELL-EDA Architecture

Feature Store Architecture (Enterprise Ready)

C. Feature Registry (Metadata Service)

Table: feature_registry

FeatureName	Type	Domain	Version	Description
state_change_count_last_30_days	int	Contact	1	Rolling count
total_logins_last_7_days	int	Contact	1	Rolling count
avg_session_duration_minutes	float	Contact	1	Avg session length
days_since_last_payment	int	Billing	1	Billing recency

This registry ensures:

No duplicate features

No conflicting definitions

Versioning for evolution

Documentation for data scientists

NETCELL EDA PLATFORM

Event-Driven AI Feature Pipeline

NETCELL-EDA Architecture

Feature Pipeline

This layout is perfect for Cloudera + Kubernetes

Online store → fast microservices

Offline store → Cloudera's strength (Iceberg/Hive)

Registry → governance (Ranger/Atlas)

Pipelines → scalable (Spark/Flink/K8s jobs)

NETCELL-EDA Architecture

Feature Pipeline

Step 1 — Event arrives

Example: ContactStateChanged.

Step 2 — Processor updates profile

Adds state history, timestamps, etc.

Step 3 — Processor updates online features

Calculates:

time since last state change

rolling counts

recency metrics

Writes to Online Feature Store.

Step 4 — Daily batch job builds offline features

Reads:

raw events from Kafka → Data Lake

profiles from Document DB

billing/security tables

Computes:

rolling windows

aggregates

labels (for training)

Writes to Offline Feature Store.

Step 5 — Model training uses offline features

Trains risk/churn/engagement models.

Step 6 — Inference Service uses online features

Real-time scoring.

NETCELL-EDA Architecture

Feature Pipeline

Thank you