OPEN SOURCE · APACHE 2.0

Streaming Storage for

Apache Fluss (Incubating) is an open-source, lakehouse-native streaming storage. It collapses the message broker, online KV store, stream-processing state backend, and lakehouse cold store into a single coherent foundation, making the Lakehouse truly real-time.

Get Started →

View on GitHub

Flink SQL Spark SQL

```
-- Register Apache Fluss as a Flink catalog
CREATE CATALOG fluss_catalog WITH (
  'type'              = 'fluss',
  'bootstrap.servers' = 'coordinator-server:9123'
);
USE CATALOG fluss_catalog;

-- Create a primary-key table
CREATE TABLE pk_table (
  shop_id    BIGINT,
  user_id    BIGINT,
  num_orders INT,
  PRIMARY KEY (shop_id, user_id) NOT ENFORCED
) WITH ('bucket.num' = '4');

INSERT INTO pk_table VALUES (1234, 1234, 1);
SELECT * FROM pk_table WHERE shop_id = 1234;
```

ARCHITECTURE

Unlocking the Streamhouse Architecture

THE MULTIPLE-SYSTEMS TAX

Five systems, four integrations, continuous engineering tax.

A conventional real-time AI stack stitches together a message broker, a stream processor, an online store, an offline store, and a synchronization layer. Every boundary is an integration point where data silently diverges. Apache Fluss collapses that stack into one substrate.

BEFORE · FRAGMENTED STACK

01 **Message broker**
Kafka, for event transport.

02 **Stream processor**
Flink or Spark, for derived features and aggregations.

03 **Online store**
Redis or DynamoDB, for sub-millisecond lookup.

04 **Offline store**
Iceberg or Parquet on S3, for training and history.

05 **Sync layer**
Bespoke pipelines and freshness monitors that drift silently.

5 Systems · 4 Sync Boundaries · Continuous Engineering Tax

AFTER · UNIFIED SUBSTRATE

Apache Fluss

One columnar streaming store designed for the real-time AI data plane.

- ✓ **Streaming Log** · Durable, replayable, offset-ordered streams
- ✓ **PK Lookup** · Sub-millisecond key/value serving
- ✓ **Streamhouse** · Real-time data layer for Lakehouse architecture
- ✓ **State Store** · Externalized state for joins and aggregations
- ✓ **Multi-Modal** · Lance integration for vectors and ML context
- ✓ **Audit Trail** · Change data feed, replayable by design

SIX CAPABILITY PILLARS

The benefits, grounded in the architecture.

Each pillar is a direct consequence of a specific architectural mechanism. Together they collapse the fragmented real-time stack into a single coherent foundation.

01

Unified Architecture

One system for messaging, applications, analytics, and AI.

Replaces the message queue, key-value store, and OLAP engine with a single platform serving transport, lookups, and queries from the same data.

ARCHITECTURAL BASIS

Dual representation of PK Tables (Log Store & KV Store).

02

Stream & Lakehouse Unification

One copy of data across real-time and batch layers.

Hot and cold tiers share the same schema and are queryable as one substrate, so streaming and historical reads hit one source of truth.

ARCHITECTURAL BASIS

Tiering Service and Union Read across Iceberg, Paimon, and Lance.

03

04

Compute / Storage Separation

Lean, elastic, stateless compute with fast recovery.

Stateless compute recovers in seconds and runs up to 85% cheaper than Kafka-based topologies. State lives on the Fluss leader, not Flink slots.

ARCHITECTURAL BASIS

Stateless compute model with leader-resident state and KV snapshots.

Columnar Streaming Analytics

Pruning that compounds.

Server-side projection, predicate pushdown, and partition pruning on Arrow-format streams compound into order-of-magnitude I/O and network savings.

ARCHITECTURAL BASIS

ARROW log format and the compound pruning stack on the TabletServer.

05

Feature & Context Stores

Multi-modal data on one substrate, ready for ML and AI.

Row, columnar, and vector data on one store. Online features, RAG context, and analytics collapse into one PK Table accessed through different views.

ARCHITECTURAL BASIS

Unified substrate spanning structured features and vector context.

06

Ecosystem Openness

Open formats. No vendor lock-in.

Readable by Flink, Spark, Trino, StarRocks, and DuckDB. Native hot tier plus Iceberg, Paimon, and Lance for the cold tier, open formats end to end.

ARCHITECTURAL BASIS

Open lake formats throughout, governed at the Apache Software Foundation.

APACHE FLUSS VS APACHE KAFKA

Where Streams Meet The Lakehouse

Kafka is the streaming transport. Fluss is the streaming storage. If your need is large-scale stream processing with Flink, real-time analytics, AI/ML, or a sub-second lakehouse, Fluss is the shared streaming storage substrate behind all of them. Read the full breakdown to see which fits your stack.

See the full comparison →

COMMUNITY

Built in the open, governed by the ASF.

Apache Fluss is developed openly by a global community of contributors. Join the discussion, file an issue, or send a patch.

Apache 2.0
Open-source license

ASF
Apache Software Foundation governance

GitHub

Source code, issues, and pull requests.

Open repository →

Slack

Real-time chat with users and committers.

Join the workspace →

Contribute

Welcome guide, mailing lists, and how to send your first patch.

Get started →

PRODUCT

Documentation

Quickstart

Roadmap

Downloads

Blog

COMMUNITY

GitHub

Slack

Welcome

Contribute

RESOURCES

Talks

Videos

Issues

Releases

APACHE