Apache Hudi

This wiki space hosts

- Technical documentation
- How-to blogs
- Design documents/RFCs
- Community Management
- Roadmap
  - Integrations
  - Writing
  - Querying
  - Metadata Management

If you are looking for documentation on using Apache Hudi, please visit the project site or engage with our community

Technical documentation

- Overview of design & architecture
- Migration guide to org.apache.hudi
- Tuning Guide
- FAQs

How-to blogs

1. How to manually register Hudi tables into Hive via Beeline?
2. Ingesting Database changes via Sqoop/Hudi
3. De-Duping Kafka Events With Hudi DeltaStreamer

Design documents/RFCs

RFCs are the way to propose large changes to Hudi and the RFC Process details how to go about driving one from proposal to completion. Anyone can initiate a RFC. Please note that if you are unsure of whether a feature already exists or if there is a plan already to implement a similar one, always start a discussion thread on the dev mailing list before initiating a RFC. This will help everyone get the right context and optimize everyone’s usage of time.

Below is a list of RFCs

- Completed
- Inactive
- WIP
- RFC-02: ORC Storage in Hudi
- RFC-08 Record level indexing mechanisms for Hudi datasets
- RFC - 13 : Integrate Hudi with Flink
- RFC - 14 : JDBC incremental puller
- RFC - 20 : handle failed records
- RFC - <RFC NUMBER>
- RFC - 23 : Hudi Observability metrics collection
- RFC-24: Hoodie Flink Writer Proposal
- RFC-26 Optimization For Hudi Table Query
- RFC-27 Data skipping index to improve query performance
- RFC-28 Support Z-order curve
- RFC - 29: Hash Index
- RFC - 30: Batch operation
- RFC - 31: Hive integration Improvment
- RFC-32 Kafka Connect Sink for Hudi
- RFC - 33 Hudi supports more comprehensive Schema Evolution
- RFC-34 Hudi BigQuery Integration (WIP)
- RFC-35: Make Flink MOR table writing streaming friendly
- [WIP] RFC-36: HUDI Metastore Server

Community Management

- Apache Hudi - Release Guide (Pre Graduation)
- Apache Hudi Community Bi-Weekly Sync
- Committer On-boarding Guide
- Community Support

Roadmap
Below is a tentative roadmap for 2021 (in no particular order; since that is determined by Release Management process)

**Integrations**

1. Spark SQL with Merge/Delete statements support (RFC - 25: Spark SQL Extension For Hud)
2. Trino integration with support for querying/writing Hudi table using SQL statements
3. Kinesis/Pulsar integrations with DeltaStreamer
4. Kafka Connect Sink for Hudi
5. Dremio integration
6. Interops with other table formats
7. ORC Support

**Writing**

- Indexing
  - MetadataIndex implementation that servers bloom filters/key ranges from metadata table, to speed up bloom index on cloud storage.
  - Addition of record level indexes for fast CDC (RFC-08 Record level indexing mechanisms for Hudi datasets)
  - Range index to maintain column/field value ranges, to help file skipping for query performance
  - Addition of more auxiliary indexing structures - bitmaps, ..
  - global/hash based index to faster point-in-time lookup
- Concurrency Control
  - Addition of optimistic concurrency control, with pluggable locking services.
  - Non-blocking clustering implementation w.r.t updates
  - Multi-writer support with fully non-blocking log based concurrency control.
  - Multi table transactions
- Performance
  - Integrate row writer with all Hudi writer operations
- Self Managing
  - Clustering based on historical workload trend
  - On-fly data locality during write time (HUDI-1628)
  - Auto Determination of compression ratio

**Querying**

- Performance
  - Complete integration with metadata table.
  - Realtime view performance/memory footprint reduction.
- PrestoDB
  - Incremental Query support on Presto
- Hive
  - Storage handler to leverage metadata table for partition pruning
- Spark SQL
  - Hardening incremental pull via Realtime view
  - Spark Datasource redesign around metadata table
  - Streaming ETL via Structured Streaming
- Flink
  - Support for end-end streaming ETL pipelines
  - Materialized view support via Flink/Calcite SQL
- Mutable, Columnar Cache Service
  - File group level caching to enable real-time analytics (backed by Arrow/AresDB)

**Metadata Management**

- Standalone timeline server
  - Serves interactive query planning performance: schema, DFS listings, statistics, timeline requests
  - High availability/sharding
  - Pluggable backing stores including rocksDB, Dynamo, Spanner