

## Apache Hive Overview

Date published: 2020-11-30

Date modified: 2025-10-22

# CLOUdera

# Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>Apache Hive features in Cloudera Data Warehouse.....</b>	<b>4</b>
<b>Apache Tez and Hive LLAP.....</b>	<b>5</b>
<b>Apache Hive 3 in architectural overview.....</b>	<b>5</b>
<b>Apache Hive content roadmap.....</b>	<b>6</b>

# Apache Hive features in Cloudera Data Warehouse

Major changes to Apache Hive 2.x improve Apache Hive 3.x transactions and security. Knowing the major differences between these versions is critical for SQL users, including those who use Apache Spark and Apache Impala.

Hive is a data warehouse system for summarizing, querying, and analyzing huge, disparate data sets.

## ACID transaction processing

Hive 3 tables are ACID (Atomicity, Consistency, Isolation, and Durability)-compliant. Hive 3 write and read operations improve the performance of transactional tables. Atomic operations include simple writes and inserts, writes to multiple partitions, and multiple inserts in a single SELECT statement. A read operation is not affected by changes that occur during the operation. You can insert or delete data, and it remains consistent throughout software and hardware crashes. Creation and maintenance of Hive tables is simplified because there is no longer any need to bucket tables.

## Materialized views

Because multiple queries frequently need the same intermediate roll up or joined table, you can avoid costly, repetitious query portion sharing, by precomputing and caching intermediate tables into views.

## Query results cache

Hive filters and caches similar or identical queries. Hive does not recompute the data that has not changed. Caching repetitive queries can reduce the load substantially when hundreds or thousands of users of BI tools and web services query Hive.

## Scheduled Queries

Using SQL statements, you can schedule Hive queries to run on a recurring basis, monitor query progress, temporarily ignore a query schedule, and limit the number running in parallel. You can use scheduled queries to start compaction and periodically rebuild materialized views, for example.

## Security improvements

Apache Ranger secures Hive data by default. To meet demands for concurrency improvements, ACID support, render security, and other features, Hive tightly controls the location of the warehouse on a file system, or object store, and memory resources.

With Apache Ranger and Apache Hive ACID support, your organization will be ready to support and implement GDPR (General Data Protection Regulation).

## Workload management at the query level

You can configure who uses query resources, how much can be used, and how fast Hive responds to resource requests. Workload management can improve parallel query execution, cluster sharing for queries, and query performance. Although the names are similar, Hive workload management queries are unrelated to Cloudera Workload XM for reporting and viewing millions of queries and hundreds of databases.

## Connection Pooling

Hive supports HikariCP JDBC connection pooling.

## Unsupported features

Cloudera does not support the following features that were available in HDP and CDH platforms:

- Multiple insert overwrite queries that read data from a source table.
- CREATE TABLE that specifies a managed table location

Do not use the LOCATION clause to create a managed table. Hive assigns a default location in the warehouse to managed tables.

- CREATE INDEX

Hive builds and stores indexes in ORC or Parquet within the main table, instead of a different table, automatically. Set `hive.optimize.index.filter` to enable use (not recommended--use materialized views instead). Existing indexes are preserved and migrated in Parquet or ORC to Cloudera during upgrade.

## Apache Tez and Hive LLAP

Cloudera Data Warehouse supports low-latency analytical processing (LLAP) of Hive queries.

Hive uses Apache Tez to execute queries internally. Apache Tez provides the following execution modes:

- Container mode

Every time you run a Hive query, Tez requests a container from YARN.

- LLAP mode

Every time you run a Hive query, Tez asks the LLAP daemon for a free thread, and starts running a fragment.

Apache Tez provides the framework to run a job that creates a graph with vertexes and tasks. SQL semantics for deciding the query physical plan, which identifies how to execute the query in a distributed fashion, is based on Apache Tez. The entire execution plan is created under this framework. SQL syntax in Hive is the same irrespective of execution engine (mode) used in Hive.

Apache Tez does not have to start from the ground up, requesting a container and waiting for a JVM, to run a fragment in LLAP mode. LLAP mode provides dedicated capacity. Caching is automated. The primary difference between LLAP mode and container mode, is that in LLAP mode the LLAP executors are used to run query fragments.

Cloudera Data Warehouse, the Hive execution mode is LLAP. In Cloudera Data Hub on Cloudera on cloud and Cloudera Base on premises, the Hive execution mode is container, and LLAP mode is not supported. When Apache Tez runs Hive in container mode, it has traditionally been called Hive on Tez.

## Apache Hive 3 in architectural overview

Understanding Apache Hive 3 major design features, such as default ACID transaction processing, can help you use Hive to address the growing needs of enterprise data warehouse systems.

### Data storage and access control

One of the major architectural changes to support Hive 3 design gives Hive much more control over metadata memory resources and the file system, or object store. The following architectural changes from Hive 2 to Hive 3 provide improved security:

- Tightly controlled file system and computer memory resources, replacing flexible boundaries: Definitive boundaries increase predictability. Greater file system control improves security.
- Optimized workloads in shared files and containers

Hive 3 is optimized for object stores in the following ways:

- Hive uses ACID to determine which files to read rather than relying on the storage system.
- In Hive 3, file movement is reduced from that in Hive 2.
- Hive caches metadata and data aggressively to reduce file system operations

The major authorization model for Hive is Ranger. Hive enforces access controls specified in Ranger. This model offers stronger security than other security schemes and more flexibility in managing policies.

This model permits only Hive to access the Hive warehouse.

### Transaction processing

You can deploy new Hive application types by taking advantage of the following transaction processing characteristics:

- Mature versions of ACID transaction processing:

ACID tables are the default table type.

ACID enabled by default causes no performance or operational overload.

- Simplified application development, operations with strong transactional guarantees, and simple semantics for SQL commands

You do not need to bucket ACID tables.

- Materialized view rewrites
- Automatic query cache
- Advanced optimizations

### Hive client changes

You can use the thin client Beeline for querying Hive from a client. You can run Hive administrative commands from the client. Beeline uses a JDBC connection to Hive to run commands. Hive parses, compiles, and runs operations. Beeline supports many of the command-line options that Hive CLI supported. Beeline does not support `hive -e set key=value` to configure the Hive Metastore.

You enter supported Hive CLI commands by invoking Beeline using the `hive` keyword, command option, and command. For example, `hive -e set`. Using Beeline instead of the thick client Hive CLI, which is no longer supported, has several advantages, including low overhead. Beeline does not use the entire Hive code base. A small number of daemons required to run queries simplifies monitoring and debugging.

Hive enforces allowlist and denylist settings that you can change using SET commands. Using the denylist, you can restrict memory configuration changes to prevent instability. Different Hive instances with different allowlists and denylists to establish different levels of stability.

### Apache Hive Metastore sharing

Hive, Impala, and other components can share a remote Hive metastore.

### Query execution of batch and interactive workloads

You can connect to Hive using a JDBC command-line tool, such as Beeline, or using an JDBC/ODBC driver with a BI tool, such as Tableau. You configure the settings file for each instance to perform either batch or interactive processing.

### Related Information

[Hive Warehouse Connector for accessing Apache Spark data](#)

[Blog: Enabling high-speed Spark direct reader for Apache Hive ACID tables](#)

## Apache Hive content roadmap

The content roadmap provides links to the available content resources for Apache Hive.

**Table 1: Apache Hive Content roadmap**

Task	Resources	Source	Description
Understanding	<a href="#">Presentations and Papers about Hive</a>	Apache wiki	Contains meeting notes, presentations, and whitepapers from the Apache community.
Getting Started	<a href="#">Hive Tutorial</a>	Apache wiki	Provides a basic overview of Apache Hive and contains some examples on working with tables, loading data, and querying and inserting data.
Developing	<a href="#">Materialized Views</a>	Apache wiki	Covers accelerating query processing in Cloudera Data Warehouse by pre-computing summaries using materialized views.
	<a href="#">JdbcStorageHandler</a>	Apache wiki	Describes how to read from a JDBC data source in Hive.
	<a href="#">Hive transactions</a>	Apache wiki	Describes ACID operations in Hive.
	<a href="#">Hive Streaming API</a>	Apache wiki	Explains how to use an API for pumping data continuously into Hive using clients such as NiFi and Flume.
	<a href="#">Hive Operators and Functions</a>	Apache wiki	Describes the Language Manual UDF.
	<a href="#">Beeline: HiveServer2 Client</a>	Apache wiki	Describes how to use the Beeline client.
Reference	<a href="#">SQL Language Manual</a>	Apache wiki	Language reference documentation available in the Apache wiki.
Contributing	<a href="#">Hive Developer FAQ</a>	Apache wiki	Resources available if you want to contribute to the Apache community.
	<a href="#">How to Contribute</a>	Apache wiki	
	<a href="#">Hive Developer Guide</a>	Apache wiki	
	<a href="#">Plug-in Developer Kit</a>	Apache wiki	
	<a href="#">Unit Test Parallel Execution</a>	Apache wiki	
	<a href="#">Hive Architecture Overview</a>	Apache wiki	
	<a href="#">Hive Design Docs</a>	Apache wiki	
	<a href="#">Project Bylaws</a>	Apache wiki	