

1.0.0

## Apache Impala

Date published: 2020-11-30

Date modified: 2024-07-26

# CLOUDERA

<https://docs.cloudera.com/>

# Legal Notice

© Cloudera Inc. 2024. 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

**Apache Impala Overview..... 4**  
**Components of Impala..... 4**  
**Catalog Service High Availability..... 7**

## Apache Impala Overview

The Apache Impala provides high-performance, low-latency SQL queries on data stored in popular Apache Hadoop file formats.

The Impala solution is composed of the following components.

### **Impala**

The Impala service coordinates and executes queries received from clients. Queries are distributed among Impala nodes, and these nodes then act as workers, executing parallel query fragments.

### **Hive Metastore**

Stores information about the data available to Impala. For example, the metastore lets Impala know what databases are available and what the structure of those databases is. As you create, drop, and alter schema objects, load data into tables, and so on through Impala SQL statements, the relevant metadata changes are automatically broadcast to all Impala nodes by the dedicated catalog service.

### **Clients**

Entities including Hue, ODBC clients, JDBC clients, Business Intelligence applications, and the Impala Shell can all interact with Impala. These interfaces are typically used to issue queries or complete administrative tasks such as connecting to Impala.

### **Storage for data to be queried**

Queries executed using Impala are handled as follows:

1. User applications send SQL queries to Impala through ODBC or JDBC, which provide standardized querying interfaces. The user application may connect to any impalad in the cluster. This impalad becomes the coordinator for the query.
2. Impala parses the query and analyzes it to determine what tasks need to be performed by impalad instances across the cluster. Execution is planned for optimal efficiency.
3. Storage services are accessed by local impalad instances to provide data.
4. Each impalad returns data to the coordinating impalad, which sends these results to the client.

## Components of Impala

The Impala service is a distributed, massively parallel processing (MPP) database engine. It consists of different daemon processes called as components. This topic describes the different roles these components play for a selected Virtual Warehouse.

When you create an Impala Virtual Warehouse, it is automatically optimized for your workload by the Cloudera Data Warehouse (CDW) service. Due to the containerized and compute-isolated architecture of CDW, as well as intelligence to assign different default configurations, you do not have to customize your environment to optimize performance or to avoid resource usage spikes and out-of-memory conditions. However, if you must adjust some settings to suit your needs after creating an Impala Virtual Warehouse, you can add particular component configuration details in one of the free-form fields on the SIZING AND SCALING tab or under one of the components available under the CONFIGURATIONS tab for a selected Virtual Warehouse.

The screenshot displays the 'Virtual Warehouses' management interface, specifically the 'Sizing and Scaling' tab. The interface includes a navigation bar with tabs for 'SIZING AND SCALING' (highlighted), 'CONFIGURATIONS', 'DIAGNOSTIC BUNDLE', and 'EVENTS TIMELINE'. Below the navigation bar is a 'WEB UI' section. The main content area is titled 'Cluster Shape' and contains several configuration options:

- User Groups:** A dropdown menu labeled 'Select Groups' with a help icon.
- Disable AutoSuspend:** A toggle switch currently turned off.
- Allow Shutdown Of Coordinator:** A toggle switch currently turned off.
- Trigger Shutdown Delay (in seconds):** A slider set to 300, with a range from 0 to 7000.
- AutoSuspend Timeout (in seconds):** A slider set to 300, with a range from 0 to 7000.
- Nodes:** A slider set to 40, with a range from 2 to 200 (labeled 'Min:2, Max:40').
- Scale Up Delay (in seconds):** A slider set to 20, with a range from 0 to 1000.
- Scale Down Delay (in seconds):** A slider set to 20, with a range from 0 to 1000.
- Enable Legacy Multithreading:** A toggle switch currently turned off, with a 'Learn more' link.

Before making any configuration changes, you can review the default configuration details in the components available under the CONFIGURATIONS tab in a Virtual Warehouse.

Impala service in CDW consists of the following different daemon processes.

- Impala catalogd
- Impala coordinator
- Impala executor
- Impala statestored
- Impala autoscaler
- Impala proxy

## Virtual Warehouses

Configuration files: flagfile Search Keys +

KEY	VALUE
admission_control_service_num_svc_threads	0
admission_control_service_queue_mem_limit	50MB
admission_status_wait_time_ms	100
admission_thread_pool_size	5
fair_scheduler_allocation_path	/opt/impala/conf/fair-scheduler.xml
llama_site_path	/opt/impala/conf/llama-site.xml
logtostderr	true
max_admission_queue_size	50
metrics_webserver_port	25031
redirect_stdout_stderr	false

< 1 2 >

### Components available for setting the configuration

This section provides reference information on supported configuration properties under the listed components. For the full list of configuration properties, see [Impala Properties in Cloudera Runtime](#). Based on the information provided under the components, choose the appropriate component/role to tune a configuration if you must.



**Note:** Some of the configuration properties under these components can be tuned to your needs and some of the properties are not editable. In addition, the available properties may differ by CDW Impala Runtime version.

#### Impala catalogd

The Catalog Server relays the metadata changes from Impala SQL statements to all the Impala daemons in a cluster. The catalog service avoids the need to issue `REFRESH` and `INVALIDATE METADATA` statements when the metadata changes are performed by statements issued through Impala.

#### Impala coordinator

A few of the key functions that an Impala coordinator performs are:

- Reads and writes to data files.
- Accepts queries transmitted from the `impala-shell` command, Hue, JDBC, or ODBC.
- Parallelizes the queries and distributes work across the cluster.

It is in constant communication with StateStore, to confirm which executors are healthy and can accept new work. Based on the health information it receives it assigns tasks to the executors. It also receives broadcast messages from the Catalog Server daemon whenever a cluster creates, alters, or drops any type of object, or when an `INSERT` or `LOAD DATA` statement is processed through Impala. It also communicates to Catalog Server daemon to load table metadata.

#### Impala executor

A few of the key functions that an Impala executor performs are:

- Executes the queries and transmits query results back to the central coordinator.
- Also transmits intermediate query results.

Depending on the size and the complexity of your queries you can select the number of executor nodes that are needed to run a typical query in your workloads. For more information on the

executor groups and recommendations on sizing your virtual warehouse to handle queries that must be run in your workloads concurrently, see [Impala auto-scaling on public clouds](#).

### **Impala statestored**

The Impala StateStore checks on the health of all Impala daemons in a cluster, and continuously relays its findings to each of those daemons. If an Impala daemon goes offline due to hardware failure, network error, software issue, or other reason, the StateStore informs all the other Impala daemons so that future queries can avoid making requests to the unreachable Impala daemon.

### **Components available for diagnostic purpose only**

The following components are read-only, and available only to view for diagnostic purposes. The properties listed under them are not tunable.

### **Impala autoscaler**

One of the core Impala Virtual Warehouse components is Impala autoscaler. Impala autoscaler is in constant communication with coordinators and executors to determine when more or fewer compute resources are needed, given the incoming query load. When the autoscaler detects an imbalance in resources, it sends a request to the Kubernetes framework to increase or decrease the number of executor groups in the Virtual Warehouse, thereby right-sizing the amount of resources available for queries. This ensures that workload demand is met without wasting cloud resources.

### **Impala proxy**

Impala Proxy is a small footprint reverse proxy that will forward every http client request to the Coordinator endpoint.

When you create an Impala warehouse with the 'allow coordinator shutdown' option a proxy 'impala proxy' is created to act as a connection endpoint to the impala clients. This option when enabled allows Impala coordinators to automatically shut down during idle periods. If the coordinator has shut down because of idle period, and if there is a request from a client when the coordinator is not running, impala proxy triggers the coordinator to start. So the proxy acts as a layer between the clients and the coordinator so that the clients connected to the VW do not time out when the coordinator starts up.

## **Catalog Service High Availability**

By default, the Impala Virtual Warehouse runs with Catalog service in Active-Passive HA mode which runs two Impala catalog instances. If you enable HA mode for Impala Virtual Warehouse, the statestore allows one of the catalog instances to become active, and the other paired catalog instance to become a standby.

With any new query requests, the Impala coordinator sends metadata requests to the catalog service and sends metadata updates to the catalog which in turn propagates metadata updates to HMS. With a pair of primary/standby Catalog instances, the standby instance will be promoted as the primary instance to continue executing queries when the primary instance goes down. This High Availability (HA) mode of catalog service in CDW reduces the outage duration of the Impala cluster when the primary catalog service fails.

### **Catalog Failure Detection**

The Statestore instance continuously sends a heartbeat to its registered clients, including the primary and standby Catalog instances, to determine if they are healthy. If the Statestore finds the primary Catalog instance is not healthy, but the standby Catalog instance is healthy, the Statestore promotes the standby Catalog instance as the primary instance and notifies all coordinators and catalogs of this change. Coordinators will switch over to the new primary Catalog instance.