Cloudera Runtime 7.3.3

Configuring Apache Kudu

Date published: 2020-07-28
Date modified: 2026-12-13



https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2026. 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

Configure Kudu processes 4
Experimental flags 4

Configuring the Kudu master 5

Configuring tablet servers 5

Rack awareness (Location awareness) 6

Directory configurations 7
Changing directory configuration 7

Configure Kudu processes

You can pass command-line flags when you start a Kudu process to configure its behavior or read those options from configuration files by passing them using one or more --flagfile=<file> options.

You can include the --flagfile option within your configuration file to include other files. Learn more about gflags by reading its documentation.

You can place options for masters and tablet servers in the same configuration file, and each will ignore options that do not apply.

Flags can be prefixed with either one or two - characters. This documentation standardizes on two: --example_flag.

Only the most common configuration options are documented in this topic. For a more exhaustive list of configuration options, see the Kudu Configuration Reference. To see all configuration flags for a given executable, run it with the --help option.

Experimental flags

Some configuration flags are marked 'unsafe' and 'experimental'. Such flags are disabled by default.
You can access these flags by enabling the additional flags, `--unlock_unsafe_flags` and `--unlock_experimental_flags`.

Note that these flags might be removed or modified without a deprecation period or any prior notice in future Kudu releases. Cloudera does not support using unsafe and experimental flags. As a rule of thumb, Cloudera will not support any configuration flags not explicitly documented in the Kudu Configuration Reference Guide.

Some particularly impactful unsafe Kudu configuration options are:

- --use-hybrid-clock

  Setting this flag to false is highly discouraged in any production-grade deployment, as it can introduce extensive latency and behavior such as not operating in COMMIT_WAIT consistency mode, failing to handle scan operations in READ_AT_SNAPSHOT and READ_YOUR_WRITES modes, and so on.
- --allow_world_readable_credentials

  Do not set this flag to true, but rather set the ownership and file access mask properly using chown and chmod commands.
- --allow_unsafe_replication_factor

  Do not set this flag to true since even replication factors of (2 * N) is not any better than (2 * N - 1) because Kudu uses Raft consensus protocol.
- --max_cell_size_bytes

  If you need to increase it, you must consult with Cloudera to evaluate possible performance degradation and other issues depending on specific workload.
- --max_num_columns

  If you need to adjust it, you must consult with Cloudera to evaluate possible performance degradation and other issues depending on specific workload.
- --rpc_num_service_threads

  You should never set it higher than the number of CPU cores at a node.

Configuring the Kudu master

To see all available configuration options for the kudu-master executable, run it with the --help option:

```
kudu-master --help
```

Table 1: Supported configuration flags for Kudu masters

Flag	Valid options	Default	Description
--master_addresses	string	localhost	Comma-separated list of all the RPC addresses for Master consensus-configuration. If not specified, assumes a standalone Master.
--fs_data_dirs	string		List of directories where the Master will place its data blocks.
--fs_wal_dir	string		The directory where the Master will place its write-ahead logs.
--log_dir	string	/tmp	The directory to store Master log files.

For the complete list of flags for masters, see the Kudu Master Configuration Reference.

Configuring tablet servers

To see all available configuration options for the kudu-tserver executable, run it with the --help option:

```
kudu-tserver --help
```

Table 2: Supported configuration flags for Kudu tablet servers

Flag	Valid options	Default	Description
--fs_data_dirs	string		List of directories where the Tablet Server will place its data blocks.
--fs_wal_dir	string		The directory where the Tablet Server will place its write-ahead logs.
--log_dir	string	/tmp	The directory to store Tablet Server log files
--tserver_master_addrs	string	127.0.0.1:7051	Comma separated addresses of the masters that the tablet server should connect to. The masters do not read this flag.
--block_cache_capacity_mb	integer	512	Maximum amount of memory allocated to the Kudu Tablet Server's block cache.

Flag	Valid options	Default	Description
--memory_limit_hard_bytes	integer	4294967296	Maximum amount of memory a Tablet Server can consume before it starts rejecting all incoming writes.

For the complete list of flags for tablet servers, see the Kudu Tablet Server Configuration Reference.

Rack awareness (Location awareness)

Kudu's rack awareness feature provides protection from correlated failures. The rack awareness feature has two elements: location assignment and placement policy.

Kudu supports a rack awareness feature. Kudu's ordinary re-replication methods ensure the availability of the cluster in the event of a single node failure. However, clusters can be vulnerable to correlated failures of multiple nodes. For example, all of the physical hosts on the same rack in a datacenter may become unavailable simultaneously if the top-of-rack switch fails. Kudu's rack awareness feature provides protection from certain kinds of correlated failures, such as the failure of a single rack in a datacenter.

Location assignment

The first element of Kudu's rack awareness feature is location assignment. When a tablet server registers with a master, the master assigns it a location. A location is a /-separated string that begins with a / and where each /-separated component consists of characters from the set [a-zA-Z0-9_-.].

For example, /dc-0/rack-09 is a valid location, while rack-04 and /rack=1 are not valid locations. Thus location strings resemble absolute UNIX file paths where characters in directory and file names are restricted to the set [a-zA-Z0-9_-.].

Kudu does not use the hierarchical structure of locations. Location assignment is done by a user-provided command, whose path should be specified using the --location_mapping_cmd master flag. The command should take a single argument, the IP address or hostname of a tablet server, and return the location for the tablet server. Make sure that all Kudu masters are using the same location mapping command.

Placement policy

The second element of Kudu's rack awareness feature is the placement policy: Do not place a majority of replicas of a tablet on tablet servers in the same location.

The leader master, when placing newly created replicas on tablet servers and when re-replicating existing tablets, will attempt to place the replicas in a way that complies with the placement policy.

For example, in a cluster with five tablet servers A, B, C, D, and E, with respective locations /L0, /L0, /L1, /L1, /L2, to comply with the placement policy a new 3x replicated tablet could have its replicas placed on A, C, and E, but not on A, B, and C, because then the tablet would have 2/3 replicas in location /L0.

As another example, if a tablet has replicas on tablet servers A, C, and E, and then C fails, the replacement replica must be placed on D in order to comply with the placement policy.

In the case where it is impossible to place replicas in a way that complies with the placement policy, Kudu will violate the policy and place a replica anyway. For example, using the setup described in the previous paragraph, if a tablet has replicas on tablet servers A, C, and E, and then E fails, Kudu will re-replicate the tablet onto one of B or D, violating the placement policy, rather than leaving the tablet under-replicated indefinitely.

The kudu cluster rebalance tool can reestablish the placement policy if it is possible to do so. The kudu cluster rebalance tool can also be used to reimpose the placement policy on a cluster if the cluster has just been configured to use the rack awareness feature and existing replicas need to be moved to comply with the placement policy. See *Run a tablet rebalancing tool on a rack-aware cluster* for more information.

Related Information

Running a tablet rebalancing tool on a rack-aware cluster

Directory configurations

Every Kudu node requires the specification of directory flags.

The --fs_wal_dir configuration indicates where Kudu will place its write-ahead logs.

The --fs_metadata_dir configuration indicates where Kudu will place metadata for each tablet. It is recommended, although not necessary, that these directories be placed on a high-performance drives with high bandwidth and low latency, e.g. solid-state drives. If --fs_metadata_dir is not specified, metadata will be placed in the directory specified by --fs_wal_dir.

Since a Kudu node cannot tolerate the loss of its write-ahead log or metadata directories, you might want to mirror the drives containing these directories in order to make recovering from a drive failure easier. However, mirroring may increase the latency of Kudu writes.

The --fs_data_dirs configuration indicates where Kudu will write its data blocks. This is a comma-separated list of directories; if multiple values are specified, data will be striped across the directories. If not specified, data blocks will be placed in the directory specified by --fs_wal_dir.

Note: While a single data directory backed by a RAID-0 array will outperform a single data directory backed by a single storage device, it is better to let Kudu manage its own striping over multiple devices rather than delegating the striping to a RAID-0 array.

Additionally, --fs_wal_dir and --fs_metadata_dir may be the same as one of the directories listed in --fs_data_dirs, but must not be sub-directories of any of them.

Warning: Each directory specified by a configuration flag on a given machine should be used by at most one Kudu process. If multiple Kudu processes on the same machine are configured to use the same directory, Kudu may refuse to start up.

Warning: Once --fs_data_dirs is set, extra tooling is required to change it.

Note: The --fs_wal_dir and --fs_metadata_dir configurations can be changed, provided the contents of the directories are also moved to match the flags.

Changing directory configuration

For higher read parallelism and larger volumes of storage per server, you can configure servers to store data in multiple directories on different devices using the --fs_data_dirs Gflag configuration.

About this task

You can add or remove data directories to an existing master or tablet server by updating the --fs_data_dirs Gflag configuration and restarting the server. Data is striped across data directories, and when a new data directory is added, new data will be striped across the union of the old and new directories.

Note:

Removing a data directory from --fs_data_dirs may result in failed tablet replicas in cases where there were data blocks in the directory that was removed. Use ksck to ensure the cluster can fully recover from the directory removal before moving onto another server.

In versions of Kudu below 1.12, Kudu requires that the kudu fs update_dirs tool be run before restarting with a different set of data directories. Such versions will fail to start if not run.

If on a Kudu version below 1.12, once a server is started, users must go through the below steps to change the directory configuration:

Note: Unless the --force flag is specified, Kudu will not allow for the removal of a directory across which tablets are configured to spread data. If --force is specified, all tablets configured to use that directory will fail upon starting up and be replicated elsewhere.

Note: If the metadata directory overlaps with a data directory, as was the default prior to Kudu 1.7, or if a non-default metadata directory is configured, the --fs_metadata_dir configuration must be specified when running the kudu fs update_dirs tool.

Note: Only new tablet replicas, i.e. brand new tablets' replicas and replicas that are copied to the server for high availability, will use the new directory. Existing tablet replicas on the server will not be rebalanced across the new directory.

Attention: All of the command line steps below should be executed as the Kudu UNIX user, typically kudu.

Procedure

1. Use ksck to ensure the cluster is healthy, and establish a maintenance window to bring the tablet server offline.
2. The tool can only run while the server is offline, so establish a maintenance window to update the server. The tool itself runs quickly, so this offline window should be brief, and as such, only the server to update needs to be offline.

   However, if the server is offline for too long (see the follower_unavailable_considered_failed_sec flag), the tablet replicas on it may be evicted from their Raft groups. To avoid this, it may be desirable to bring the entire cluster offline while performing the update.
3. Run the tool with the desired directory configuration flags. For example, if a cluster was set up with --fs_wal_dir =/wals, ##fs_metadata_dir=/meta, and ##fs_data_dirs=/data/1,/data/2,/data/3, and /data/3 is to be removed (e.g. due to a disk error), run the command:

   ```
   $ sudo -u kudu kudu fs update_dirs --force --fs_wal_dir=/wals --fs_metad
   ata_dir=/meta --fs_data_dirs=/data/1,/data/2
   ```
4. Modify the value of the --fs_data_dirs flag for the updated sever. If using Cloudera Manager, make sure to only update the configurations of the updated server, rather than of the entire Kudu service.
5. Once complete, the server process can be started. When Kudu is installed using system packages, service is typically used:

   ```
   $ sudo service kudu-tserver start
   ```
6. Use ksck to ensure Kudu returns to a healthy state before resuming normal operation.