

Cloudera Runtime 7.1.9

Configuring Apache Impala

Date published: 2020-11-30

Date modified: 2024-07-19

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

Configuring Impala.....	4
Configuring Impala for High Availability.....	5
Enabling Catalog and StateStore High Availability (HA).....	5
Disabling Catalog and StateStore High Availability.....	7
Failure detection for Catalog and StateStore.....	8
Configuring Load Balancer for Impala.....	9
Migrating Impala Catalog to another host.....	15

Configuring Impala

You must review and configure the mandatory and recommended settings if you installed Impala without Cloudera Manager or if you want to customize your environment after installing Impala. If you installed Impala using Cloudera Manager, some of these configurations are completed automatically.

In some cases, depending on the level of Impala, CDP, and Cloudera Manager, you might need to add particular component configuration details in one of the free-form fields on the Impala configuration pages in Cloudera Manager.

- You must enable short-circuit reads, whether or not Impala was installed through Cloudera Manager. This setting goes in the Impala configuration settings, not the Hadoop-wide settings.
- If you installed Impala in an environment that is not managed by Cloudera Manager, you must enable block location tracking, and you can optionally enable native checksumming for optimal performance.

Short-Circuit Reads

Enabling short-circuit reads allows Impala to read local data directly from the file system. This removes the need to communicate through the DataNodes, improving performance. This setting also minimizes the number of additional copies of data. Short-circuit reads requires `libhadoop.so` (the Hadoop Native Library) to be accessible to both the server and the client. You must install it from an `.rpm`, `.deb`, or `parcel` to use short-circuit local reads.



Note: If you use Cloudera Manager, you can enable short-circuit reads through a checkbox in the user interface and that setting takes effect for Impala as well.

To Enable Short-Circuit Reads

You can enable short-circuit reads through a checkbox available under Configuration tab for both Impala and HDFS.

1. In **Cloudera Manager Clusters** select the Impala service, for example, `IMPALA`.
2. On the Configuration tab, search for `dfs.client.read.shortcircuit`.
3. Accept the default (enabled), or check to enable the `dfs.client.read.shortcircuit` property to read the HDFS file blocks directly.
4. Do the above for HDFS service too by clicking **Cloudera Manager Clusters** and by selecting the HDFS service.
5. Save the changes and Restart the service.

To configure DataNodes for short-circuit reads:

1. On all Impala nodes, configure the following attributes from the HDFS service as shown:
 - a. In **Cloudera Manager Clusters** select the HDFS service, for example, `HDFS`.
 - b. On the Configuration tab, search for `dfs.domain.socket.path` and set the attribute.
 - c. Search for and set the attribute, if necessary, `dfs.client.file-block-storage-locations.timeout.millis`.
 - d. Search for and set the attribute, if necessary, `dfs.datanode.hdfs-blocks-metadata.enabled`
2. After applying these changes, restart all DataNodes.

Block Location Tracking

Enabling block location metadata allows Impala to know which disk data blocks are located on, allowing better utilization of the underlying disks. Impala will not start unless this setting is enabled.

To enable block location tracking:

1. For each DataNode, enable the following attribute `dfs.datanode.hdfs-blocks-metadata.enabled` file:
 - a. In **Cloudera Manager Clusters** select the HDFS service, for example, `HDFS`.
 - b. On the Configuration tab, search for `dfs.datanode.hdfs-blocks-metadata.enabled` and enable the attribute if not already enabled.

2. After applying these changes, restart all DataNodes.

Native Checksumming

Enabling native checksumming causes Impala to use an optimized native library for computing checksums, if that library is available.

To enable native checksumming:

If you installed CDP from packages, the native checksumming library is installed and setup correctly, and no additional steps are required.

If you installed by other means, native checksumming may not be available due to missing shared objects. Finding the message "Unable to load native-hadoop library for your platform... using builtin-java classes where applicable" in the Impala logs indicates native checksumming may be unavailable.

To enable native checksumming, you must build and install libhadoop.so (the Hadoop Native Library).

Configuring Impala for High Availability

You can learn how to ensure high availability in an Impala cluster by deploying pairs of StateStore and Catalog instances in primary/standby mode. This setup reduces single points of failure and minimizes outage durations by allowing standby instances to take over when primary instances fail.

How configuring High Availability helps

The Impala StateStore checks on the health of all Impala daemons in a cluster, and continuously relays its findings to each of the daemons. The Catalog stores metadata of databases, tables, partitions, resource usage information, configuration settings, and other objects managed by Impala. If StateStore and Catalog daemons are single instances in an Impala cluster, it creates a single point of failure. Although Impala coordinators/executors continue to execute queries if the StateStore node is down, coordinators/executors will not get state updates. This causes degradation of admission control and cluster membership updates. To mitigate this, a pair of StateStore and Catalog instances can be deployed in an Impala cluster so that Impala cluster can survive failures of StateStore or Catalog.

How High Availability works in Impala cluster

With a pair of StateStore instances in primary/standby mode, the primary StateStore instance will send the cluster's state update and propagate metadata updates. It periodically sends heartbeat to the standby StateStore instance, Catalog, coordinators and executors. The standby StateStore instance also sends heartbeats to the Catalog, and coordinators and executors. RPC connections between daemons and StateStore instances are kept alive so that broken connections usually do not result in false failure reports between nodes. The standby StateStore instance takes over the primary role when the service is needed in order to continue to operate when the primary instance goes down.

With any new query requests, the Impala coordinator sends metadata requests to catalog service and sends metadata updates to catalog which in turn propagates metadata updates to hive metastore. With a pair of primary/standby Catalog instances, the standby catalog instance is promoted as the primary instance to continue Catalog service for Impala cluster when the primary instance goes down. The active catalogd acts as the source of metadata and provides Catalog service for the Impala cluster. This High Availability (HA) mode of Catalog service reduces the outage duration of the Impala cluster when the primary catalog instance fails. To support Catalog HA, you can now add two catalogd instances in an Active-Passive HA pair to an Impala cluster by choosing the High Availability option.

Enabling Catalog and StateStore High Availability (HA)

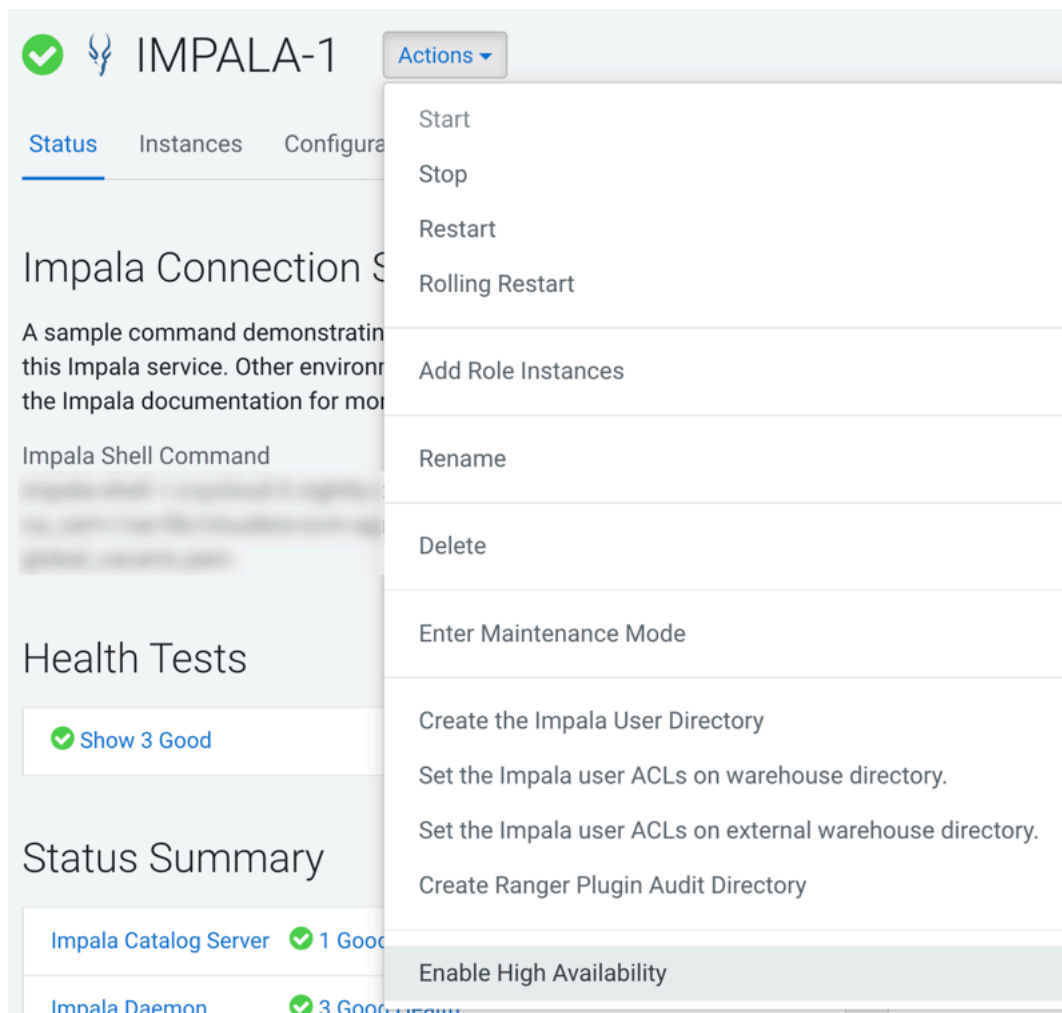
You can use Cloudera Manager to enable Catalog and StateStore HA for a cluster. When enabled, the preemptive behavior allows one of the Catalogd and StateStore pair to become active, and the other paired Catalogd and StateStore to become a standby.

Before you begin

- Impala High Availability is supported starting with 7.1.9 SP1 and Cloudera Manager 7.11.3 Cumulative Hotfix 7.
- To avoid incompatible Impala versions, High Availability should be enabled before or after rolling upgrades.
- Ensure that all nodes running Impala service are using the same CDP Parcel version.

Procedure

1. Log in to Cloudera Manager as an Administrator
2. In Cloudera Manager, select **Impala Actions Enable High Availability**.



3. On the **Assign Roles** page click **Select a host for the Catalog to act as a standby Impala Catalog server**.
4. On the **Hosts Selected** page, the existing Catalog instance is selected by default. Select a backup Catalog host. An **ICS** icon appears in the **Added Roles** column for the Catalog server.
5. On the **Assign Roles** page click **Select a host for the StateStore to act as a standby Impala StateStore server**.
6. On the **Hosts Selected** page, the existing StateStore instance is selected by default. Select a backup StateStore host. An **ISS** icon appears in the **Added Roles** column for the StateStore server.
7. Click **OK**.

8. Click **Continue** on the **Review Changes** page.

Review the summary of actions when you add a new host. When a new host is selected, Impala and its dependent services are restarted.



Important:

During a Catalogd restart, metadata operations including creating new tables, adding new partitions, and invalidating metadata do not work until the Catalog service is fully restarted.

Results

You notice two Impala Catalog servers and two Impala StateStore servers under **Impala > Instances** page.

Disabling Catalog and StateStore High Availability

You can use Cloudera Manager to disable Catalog and StateStore High Availability (HA) for a cluster.

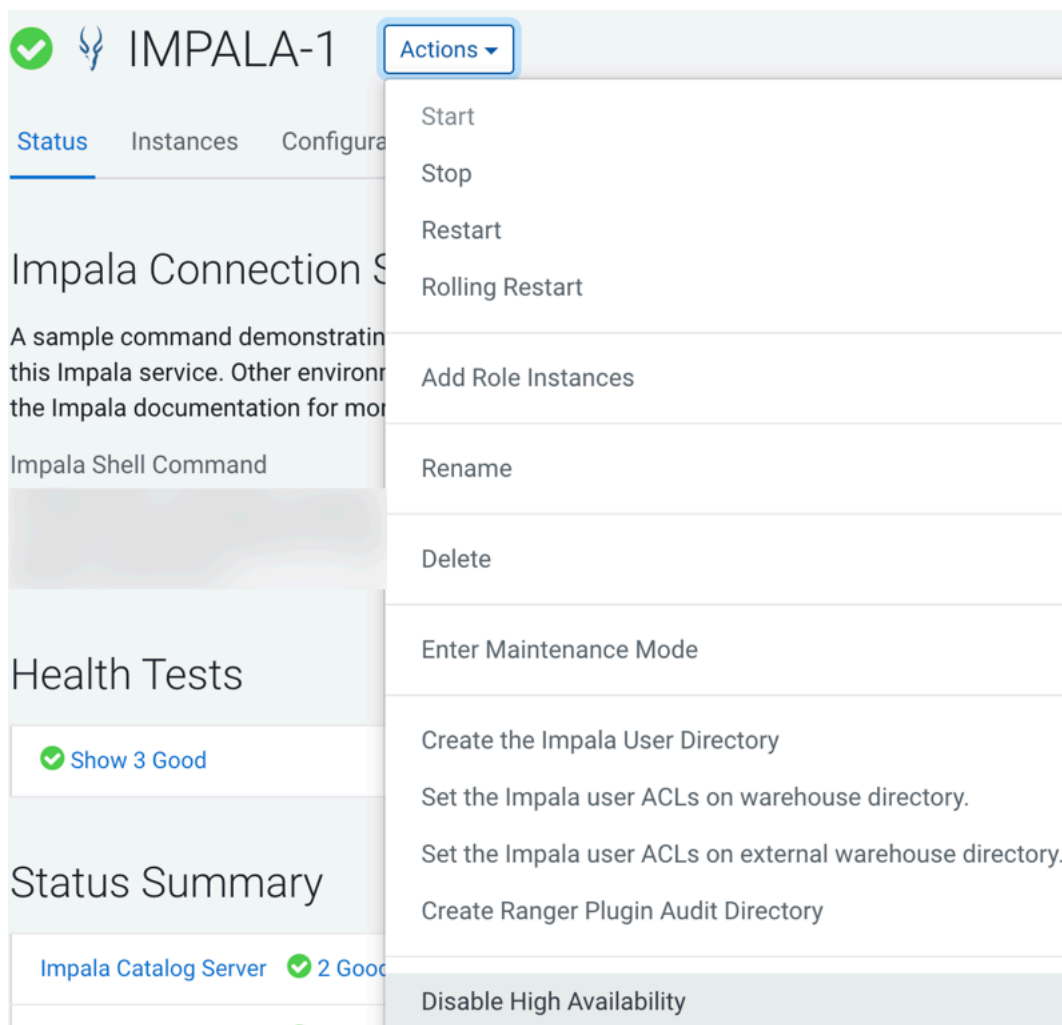
Before you begin

Ensure that all nodes running Impala service are using the same CDP Parcel version.

Procedure

1. Log in to Cloudera Manager as an Administrator.

2. In Cloudera Manager, select **Impala Actions Enable High Availability**.



3. On the **Getting started** page, select the hosts you want to remain active when reverting Impala Catalog and Impala StateStore to a single node configuration. You will need to select one host for the Impala Catalog and one host for the Impala StateStore.
4. Click **Continue**.
All the dependent services are restarted.

Results

You will notice one Impala Catalog server and one Impala StateStore server under **Impala Instances** page.

Failure detection for Catalog and StateStore

Learn how the Impala High Availability (HA) system maintains continuous operation by switching from primary to standby instances when a heartbeat check fails.

Catalog Failure Detection

The StateStore instance continuously sends heartbeat to its registered clients, including the primary and standby Catalog instances, to track Impala daemons in the cluster to determine if the daemon is healthy. If the StateStore finds the primary Catalog instance is not healthy but the standby Catalog instance is healthy, StateStore promotes the standby Catalog instance as primary instance and notifies all coordinators about this change. Coordinators switch over to the new primary Catalog instance.

When the system detects that the active catalogd is unhealthy, it initiates a failover to the standby Catalogd. During this brief transition, some nodes might not immediately recognize the new active Catalogd, causing currently running queries to fail due to lack of access to metadata. These failed queries need to be rerun after the failover is complete and the new active Catalogd is operational.

StateStore Failure Detection

The primary StateStore instance continuously sends heartbeat to its registered clients, and the standby StateStore instance. Each StateStore client registers with both active and standby statestore instances, and maintains the following information about the StateStore servers: the server IP and port, service role - primary/standby, the last time the heartbeat request was received, or number of missed heartbeats. A missing heartbeat response from the StateStore's client indicates an unhealthy daemon. There is a flag that defines `MAX_MISSED_HEARTBEAT_REQUEST_NUM` as the consecutive number of missed heartbeat requests to indicate losing communication with the StateStore server from the client's point of view so that the client marks the StateStore server as down. Standby StateStore instance collects the connection states between the clients (Catalog, coordinators and executors) and primary StateStore instance in its heartbeat messages to the clients. If the standby StateStore instance misses `MAX_MISSED_HEARTBEAT_REQUEST_NUM` of heartbeat requests from the primary StateStore instance and majority of clients lose connections with the primary statestore, it takes over the primary role.

Configuring Load Balancer for Impala

For most clusters that have multiple users and production availability requirements, you might want to set up a load-balancing proxy server to relay requests to and from Impala.

When using a load balancer for Impala, applications connect to a single well-known host and port, rather than keeping track of the hosts where a specific Impala daemon is running. The load balancer also lets the Impala nodes share resources to balance out the work loads.

Set up a software package of your choice to perform these functions.

Most considerations for load balancing and high availability apply to the `impalad` daemons. The `statestored` and `catalogd` daemons do not have special requirements for high availability, because problems with those daemons do not result in data loss.

The following are the general setup steps that apply to any load-balancing proxy software:

1. Select and download a load-balancing proxy software or other load-balancing hardware appliance. It should only need to be installed and configured on a single host, typically on an edge node.
2. Configure the load balancer (typically by editing a configuration file). In particular:
 - To relay Impala requests back and forth, set up a port that the load balancer will listen on.
 - Select a load balancing algorithm.
 - For Kerberized clusters, follow the instructions in [Special Proxy Considerations for Clusters Using Kerberos](#) on page 10 below.
3. If you are using Hue or JDBC-based applications, you typically set up load balancing for both ports 21000 and 21050, because these client applications connect through port 21050 while the `impala-shell` command connects through port 21000. See [Ports used by Impala](#) for when to use port 21000, 21050, or another value depending on what type of connections you are load balancing.
4. Run the load-balancing proxy server, pointing it at the configuration file that you set up.
5. In Cloudera Manager, navigate to ImpalaConfigurationImpala Daemon Default Group.
6. In the Impala Daemons Load Balancer field, specify the address of the load balancer in the `host:port` format.

This setting lets Cloudera Manager route all appropriate Impala-related operations through the load-balancing proxy server.

7. For any scripts, jobs, or configuration settings for applications that formerly connected to a specific `impalad` to run Impala SQL statements, change the connection information (such as the `-i` option in `impala-shell`) to point to the load balancer instead.



Note: The following sections use the HAProxy software as a representative example of a load balancer that you can use with Impala. For information specifically about using Impala with the F5 BIG-IP load balancer, see [Impala HA with F5 BIG-IP](#).

Choosing the Load-Balancing Algorithm

Load-balancing software offers a number of algorithms to distribute requests. Each algorithm has its own characteristics that make it suitable in some situations but not others.

Leastconn

Connects sessions to the coordinator with the fewest connections, to balance the load evenly. Typically used for workloads consisting of many independent, short-running queries. In configurations with only a few client machines, this setting can avoid having all requests go to only a small set of coordinators.

Recommended for Impala with F5.

Source IP Persistence

Sessions from the same IP address always go to the same coordinator. A good choice for Impala workloads containing a mix of queries and DDL statements, such as CREATE TABLE and ALTER TABLE. Because the metadata changes from a DDL statement take time to propagate across the cluster, prefer to use the Source IP Persistence algorithm in this case. If you are unable to choose Source IP Persistence, run the DDL and subsequent queries that depend on the results of the DDL through the same session, for example by running `impala-shell -f script_file` to submit several statements through a single session.

Required for setting up high availability with Hue.

Round-robin

Distributes connections to all coordinator nodes. Typically not recommended for Impala.

You might need to perform benchmarks and load testing to determine which setting is optimal for your use case. Always set up using two load-balancing algorithms: Source IP Persistence for Hue and Leastconn for others.

Special Proxy Considerations for Clusters Using Kerberos

In a cluster using Kerberos, applications check host credentials to verify that the host they are connecting to is the same one that is actually processing the request.

In Impala 2.11 and lower versions, once you enable a proxy server in a Kerberized cluster, users will not be able to connect to individual impala daemons directly from `impala-shell`.

In Impala 2.12 and higher, when you enable a proxy server in a Kerberized cluster, users have an option to connect to Impala daemons directly from `impala-shell` using the `-b / --kerberos_host_fqdn impala-shell` flag. This option can be used for testing or troubleshooting purposes, but not recommended for live production environments as it defeats the purpose of a load balancer/proxy.

Example:

```
impala-shell -i impalad-1.mydomain.com -k -b loadbalancer-1.mydomain.com
```

Alternatively, with the fully qualified configurations:

```
impala-shell --impalad=impalad-1.mydomain.com:21000 --kerberos --kerberos_host_fqdn=loadbalancer-1.mydomain.com
```

To validate the load-balancing proxy server, perform these extra Kerberos setup steps:

1. This section assumes you are starting with a Kerberos-enabled cluster.

2. Choose the host you will use for the proxy server. Based on the Kerberos setup procedure, it should already have an entry `impala/proxy_host@realm` in its keytab.
3. In Cloudera Manager, navigate to ImpalaConfigurationImpala Daemon Default Group.
4. In the Impala Daemons Load Balancer field, specify the address of the load balancer in the `host:port` format.
When this field is specified and Kerberos is enabled, Cloudera Manager adds a principal for `impala/proxy_host@realm` to the keytab for all Impala daemons.
5. Restart the Impala service.

Client Connection to Proxy Server in Kerberized Clusters

When a client connect to Impala, the service principal specified by the client must match the `-principal` setting, `impala/proxy_host@realm`, of the Impala proxy server as specified in its keytab. And the client should connect to the proxy server port.

In `hue.ini`, set the following to configure Hue to automatically connect to the proxy server:

```
[impala]
server_host=proxy_host
impala_principal=impala/proxy_host
```

The following are the JDBC connection string formats when connecting through the load balancer with the load balancer's host name in the principal:

```
jdbc:hive2://proxy_host:load_balancer_port;/principal=impala/_HOST@realm
jdbc:hive2://proxy_host:load_balancer_port;/principal=impala/proxy_host@realm
```

When starting `impala-shell`, specify the service principal via the `-b` or `--kerberos_host_fqdn` flag.

Special Proxy Considerations for TLS/SSL Enabled Clusters

When TLS/SSL is enabled for Impala, the client application, whether `impala-shell`, Hue, or something else, expects the certificate common name (CN) to match the hostname that it is connected to. With no load balancing proxy server, the hostname and certificate CN are both that of the `impalad` instance. However, with a proxy server, the certificate presented by the `impalad` instance does not match the load balancing proxy server hostname. If you try to load-balance a TLS/SSL-enabled Impala installation without additional configuration, you see a certificate mismatch error when a client attempts to connect to the load balancing proxy host.

You can configure a proxy server in several ways to load balance TLS/SSL enabled Impala:

TLS/SSL Bridging

In this configuration, the proxy server presents a TLS/SSL certificate to the client, decrypts the client request, then re-encrypts the request before sending it to the backend `impalad`. The client and server certificates can be managed separately. The request or resulting payload is encrypted in transit at all times.

TLS/SSL Passthrough

In this configuration, traffic passes through to the backend `impalad` instance with no interaction from the load balancing proxy server. Traffic is still encrypted end-to-end.

The same server certificate, utilizing either wildcard or Subject Alternate Name (SAN), must be installed on each `impalad` instance.

TLS/SSL Offload

In this configuration, all traffic is decrypted on the load balancing proxy server, and traffic between the backend `impalad` instances is unencrypted. This configuration presumes that cluster hosts reside on a trusted network and only external client-facing communication need to be encrypted in-transit.

If you plan to use Auto-TLS, your load balancer must perform TLS/SSL bridging or TLS/SSL offload.

Refer to your load balancer documentation for the steps to set up Impala and the load balancer using one of the options above.

For information specifically about using Impala with the F5 BIG-IP load balancer with TLS/SSL enabled, see [Impala HA with F5 BIG-IP](#).

Example of Configuring HAProxy Load Balancer for Impala

If you are not already using a load-balancing proxy, you can experiment with HAProxy a free, open source load balancer.



Attention: HAProxy is not a CDH component, and Cloudera does not provide the support for HAProxy. Refer to HAProxy for questions and support issues for HAProxy.

This example shows how you might install and configure that load balancer on a Red Hat Enterprise Linux system.

- Install the load balancer:

```
yum install haproxy
```

- Set up the configuration file: `/etc/haproxy/haproxy.cfg`. See the following section for a sample configuration file.
- Run the load balancer (on a single host, preferably one not running `impalad`):

```
/usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg
```

- In `impala-shell`, JDBC applications, or ODBC applications, connect to the listener port of the proxy host, rather than port 21000 or 21050 on a host actually running `impalad`. The sample configuration file sets `haproxy` to listen on port 25003, therefore you would send all requests to `haproxy_host:25003`.

This is the sample `haproxy.cfg` used in this example:

```
global
    # To have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events. This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #    file. A line like the following can be added to
    #    /etc/sysconfig/syslog
    #
    #    local2.*                /var/log/haproxy.log
    #
    log      127.0.0.1 local0
    log      127.0.0.1 local1 notice
    chroot   /var/lib/haproxy
    pidfile   /var/run/haproxy.pid
    maxconn   4000
    user      haproxy
    group     haproxy
    daemon

    # turn on stats unix socket
    #stats socket /var/lib/haproxy/stats
#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#
# You might need to adjust timing values to prevent timeouts.
#
# The timeout values should be dependant on how you use the cluster
```

```
# and how long your queries run.
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option              redispatch
    retries             3
    maxconn             3000
    timeout connect     5000
    timeout client      3600s
    timeout server      3600s

#
# This sets up the admin page for HA Proxy at port 25002.
#
listen stats :25002
    balance
    mode http
    stats enable
    stats auth username:password

# This is the setup for Impala. Impala client connect to load_balancer_host:25003.
# HAProxy will balance connections among the list of servers listed below.
# The list of Impalad is listening at port 21000 for beeswax (impala-shell) or original ODBC driver.
# For JDBC or ODBC version 2.x driver, use port 21050 instead of 21000.
listen impala :25003
    mode tcp
    option tcplog
    balance leastconn

    server symbolic_name_1 impala-host-1.example.com:21000 check
    server symbolic_name_2 impala-host-2.example.com:21000 check
    server symbolic_name_3 impala-host-3.example.com:21000 check
    server symbolic_name_4 impala-host-4.example.com:21000 check

# Setup for Hue or other JDBC-enabled applications.
# In particular, Hue requires sticky sessions.
# The application connects to load_balancer_host:21051, and HAProxy balances
# connections to the associated hosts, where Impala listens for JDBC
# requests on port 21050.
listen impalajdbc :21051
    mode tcp
    option tcplog
    balance source
    server symbolic_name_5 impala-host-1.example.com:21050 check
    server symbolic_name_6 impala-host-2.example.com:21050 check
    server symbolic_name_7 impala-host-3.example.com:21050 check
    server symbolic_name_8 impala-host-4.example.com:21050 check
```



Important: Hue requires the check option at the end of each line in the above file to ensure HAProxy can detect any unreachable impalad server, and failover can be successful. Without the TCP check, you can encounter an error when the impalad daemon to which Hue tries to connect is down.



Note: If your JDBC or ODBC application connects to Impala through a load balancer such as haproxy, be cautious about reusing the connections. If the load balancer has set up connection timeout values, either check the connection frequently so that it never sits idle longer than the load balancer timeout value, or check the connection validity before using it and create a new one if the connection has been closed.

Example of Configuring HAProxy Load Balancer for Impala with Auto-TLS

1. Create unencrypted key file. Auto TLS will not create the unencrypted file so you must create it manually.

```
openssl rsa -in /var/lib/cloudera-scm-agent/agent-cert/cm-auto-host_key.pem -out /etc/haproxy/cm-auto-host_unenckey.pem
# enter password from /var/lib/cloudera-scm-agent/agent-cert/cm-auto-host_key.pw
```



Note: The created cm-auto-host_cert_chain_unenckey.pem should be owned by the haproxy user with 600 or 400 privileges.

2. Combine the created unencrypted key with CA certificate:

```
cat /var/lib/cloudera-scm-agent/agent-cert/cm-auto-host_cert_chain.pem /etc/haproxy/cm-auto-host_unenckey.pem > /etc/haproxy/cm-auto-host_cert_chain_unenckey.pem
```



Note: You must reload HAProxy after combining the unencrypted key with CA certificate.

3. Configure haproxy.cfg, most often found in /etc/haproxy/haproxy.cfg. This example shown here uses non-standard ports however you may want to configure port 21000 and 21050. And you may want to use balance source instead of balance leastconn which is needed for tools like Hue to work properly.

```
frontend impala-auto-tls-thrift
  bind :21013 ssl crt /etc/haproxy/cm-auto-host_cert_chain_unenckey.pem
  mode tcp
  stats enable
  default_backend shell_autotls_backend

backend shell_autotls_backend
  mode tcp
  balance leastconn
  timeout connect 5000ms
  timeout client 3600000ms
  timeout server 3600000ms
  # Impala Nodes
  server vel311.halxg.cloudera.com vel311.halxg.cloudera.com:21000 check
  ssl ca-file /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
  server vel312.halxg.cloudera.com vel312.halxg.cloudera.com:21000 check
  ssl ca-file /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem

frontend impala-autotls-jdbc
  bind :21056 ssl crt /etc/haproxy/cm-auto-host_cert_chain_unenckey.pem
  mode tcp
  stats enable

  default_backend jdbc_autotls_backend

backend jdbc_autotls_backend
  mode tcp
  balance leastconn
  timeout connect 5000ms
  timeout client 3600000ms
  timeout server 3600000ms
  # Impala Nodes
  server vel311.halxg.cloudera.com vel311.halxg.cloudera.com:21050 check
  ssl ca-file /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
```

```
server ve1312.halxg.cloudera.com ve1312.halxg.cloudera.com:21050 ch  
eck ssl ca-file /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_ca  
certs.pem
```

Related Information

[Ports used by Impala](#)

[Impala Shell Configuration Options](#)

[Configuring Kerberos Authentication](#)

Migrating Impala Catalog to another host

You can move the Impala Catalog Server role to a new host using Cloudera Manager.

Procedure

1. Log in to Cloudera Manager
2. Stop the **Impala** service and all dependent services, including **Oozie** and **Hue**.
3. Go to the **Impala instances** page and select the **Impala Catalog Server** role type.
4. Click on Actions for Selected and choose Delete to remove the existing role. Confirm the deletion.
5. Click Add Role Instances and assign the Impala Catalog Server role to the desired host.
6. Restart the Impala service.

Results

The Impala Catalog Server role now runs on a new host.