Cloudera Runtime 7.2.11

Managing Apache ZooKeeper Security

Date published: 2019-09-23
Date modified: 2021-09-09



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

ZooKeeper Authentication....4
Configure ZooKeeper server for Kerberos authentication....4
Configure ZooKeeper client shell for Kerberos authentication....4
Verify the ZooKeeper authentication....4
Enable server-server mutual authentication....5
Use Digest Authentication Provider in ZooKeeper....6

Configure ZooKeeper TLS/SSL using Cloudera Manager....6

ZooKeeper ACLs Best Practices....7
ZooKeeper ACLs Best Practices: Atlas....7
ZooKeeper ACLs Best Practices: Cruise Control....8
ZooKeeper ACLs Best Practices: HBase....8
ZooKeeper ACLs Best Practices: HDFS....9
ZooKeeper ACLs Best Practices: Kafka....9
ZooKeeper ACLs Best Practices: Oozie....10
ZooKeeper ACLs Best Practices: Ranger....11
ZooKeeper ACLs best practices: Search....12
ZooKeeper ACLs Best Practices: YARN....14
ZooKeeper ACLs Best Practices: ZooKeeper....14

ZooKeeper Authentication

ZooKeeper supports mutual server-to-server (quorum peer) authentication using SASL (Simple Authentication and Security Layer), which provides a layer around Kerberos authentication.

Server to server authentication among ZooKeeper servers in an ensemble mitigates the risk of spoofing by a rogue server on an unsecured network. For more information about quorum peer authentication and how the feature leverages ZooKeeper's SASL support, see the Cloudera Engineering Blog post, Hardening Apache ZooKeeper Security.

Configuring ZooKeeper to use Kerberos for client-server or server-server authentication requires that your organization's Kerberos instance (MIT Kerberos, Microsoft Active Directory) be up and running, and reachable by the ZooKeeper server or client during the configuration processes detailed below.

Before enabling mutual authentication, the ZooKeeper servers in the cluster must be configured to authenticate using Kerberos.

Cloudera recommends that you ensure your ZooKeeper ensemble is working properly, before you attempt to integrate Kerberos authentication.

Configure ZooKeeper server for Kerberos authentication

You can configure the ZooKeeper server for Kerberos authentication in Cloudera Manager.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for kerberos.
4. Find the Enable Kerberos Authentication property.
5. Select the ZooKeeper services that you want to configure for Kerberos authentication.
6. Click Save Changes.

Configure ZooKeeper client shell for Kerberos authentication

The zookeeper-client shell script is automatically configured for Kerberos authentication if Kerberos is enabled in Cloudera Manager. You do not need to do any manual configurations.

Verify the ZooKeeper authentication

After enabling Kerberos authentication and restarting the ZooKeeper cluster, you can verify that the ZooKeeper authentication is working correctly.

Procedure

1. Start the ZooKeeper client, passing to it the name of a ZooKeeper server:

   ```
   zookeeper-client -server FQDN.EXAMPLE.COM:port
   ```

2. From the ZooKeeper CLI, create a protected znode using your ZooKeeper client principal.

   ```
   create /znode1 znode1data sasl:zkcli:cdwra
   ```

Cloudera Manager by default maps the Kerberos principal to its short name by setting two parameters in ZooKeeper's service configuration file zoo.cfg:

```
kerberos.removeHostFromPrincipal=true
kerberos.removeRealmFromPrincipal=true
```

With the help of above settings, for example, the client principal zkcli/myhost@EXAMPLE.COM will be authenticated in ZooKeeper as zkcli.

3. Verify the znode created and the ACL is set correctly:

```
getAcl/znode1
```

The getAcl command returns the znode's scheme and permission values.

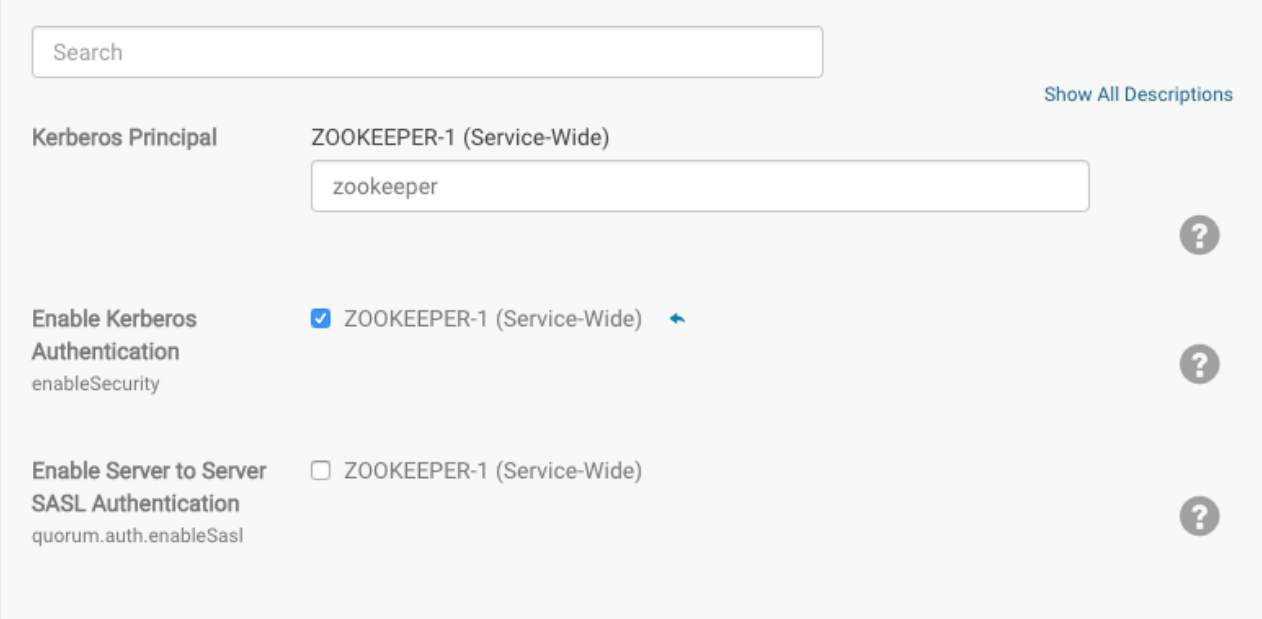
4. Verify that the znode's scheme and permissions values are as expected.

Enable server-server mutual authentication

You can enable mutual authentication between multiple ZooKeeper Servers.

About this task

Support for mutual authentication between ZooKeeper Servers can be enabled through the Cloudera Manager Admin Console. For secured networks, server-to-server authentication is considered an optional security enhancement, so the capability is disabled by default:



Server-to-server SASL authentication requires all servers in the ZooKeeper ensemble to authenticate using Kerberos.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for sasl.
4. Find the Enable Server to Server SASL Authentication and select it.
5. Click Save Changes.
6. Click the Actions button.

7. Select Restart.

Use Digest Authentication Provider in ZooKeeper

Cloudera recommends to override the default SHA1 algorithm and use a more secure one when using the Digest Authentication Provider in ZooKeeper.

About this task

When using the Digest Authentication Provider in ZooKeeper, for example for specifying SuperUser, it is recommended to override the default SHA1 algorithm and use a more secure one, such as SHA3-384 or SHA3-512. This is mandatory for FIPS compliant clusters.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Find the Java Configuration Options for ZooKeeper Server.
4. Add the following configuration:

   ```
   -Dzookeeper.DigestAuthenticationProvider.digestAlg=[***ALGORITHM***]
   ```

   For example:

   ```
   -Dzookeeper.DigestAuthenticationProvider.digestAlg=SHA3-512
   ```

Configure ZooKeeper TLS/SSL using Cloudera Manager

TLS/SSL encryption between the ZooKeeper client and the ZooKeeper server and within the ZooKeeper Quorum is supported.

About this task

The ZooKeeper TLS/SSL feature has the following limitations:

- In each ZooKeeper server process the same ZooKeeper Server KeyStore/TrustStore configuration is used for both QuorumSSL and ClientSSL.
- HTTPS for the ZooKeeper REST Admin Server is still not supported. Even if you enable SSL for ZooKeeper, the AdminServer will still use HTTP only.



Note: From 7.3.2 SP1 and higher versions, you can enable the secure ZooKeeper Admin Server by checking the Enable Secure ZooKeeper Admin Server checkbox.

TLS/SSL encryption is automatically enabled when AutoTLS is enabled. As a result it is enabled by default in Data Hub cluster templates.

You can disable, enable and configure ZooKeeper TLS/SSL manually using Cloudera Manager:

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for SSL.
4. Find the Enable TLS/SSL for ZooKeeper property and select it to enable TLS/SSL for ZooKeeper.

When TLS/SSL for ZooKeeper is enabled, two ZooKeeper features get enabled:

- QuorumSSL: the ZooKeeper servers are talking to each other using secure connection.
- ClientSSL: the ZooKeeper clients are using secure connection when talking to the ZooKeeper server.

5. Find the Secure Client Port property and change the port number if necessary.

   When ClientSSL is enabled, a new secure port is opened on the ZooKeeper server which handles the SSL connections. Its default value is 2182.

   The old port (default: 2181) will still be available for unsecured connections.
6. Click Save Changes.
7. Restart.

What to do next

Enable and configure ClientSSL by other components that support this feature.

The following components support ZooKeeper TLS/SSL:

- HBase Indexer (KS Indexer)
- Kafka
- Oozie
- Solr

ZooKeeper ACLs Best Practices

As more and more components begin to rely on ZooKeeper within a Hadoop cluster, there are various permissions that need to be maintained to ensure the integrity and security of the znodes. These permissions are different from component to component. You must follow the required steps for tightening the ZooKeeper ACLs or permissions when provisioning a secure cluster as a best practices guideline.

Some components only use ZooKeeper when they are running in their component specific HA mode. Others have separate secure and unsecure ACLs defined and switch between which to enforce based on the component knowledge of whether the cluster is secured or not.

In general, ACLs are pretty open and assume an unsecure cluster by default. These permissions need to be hardened for secure clusters in order to avoid inappropriate access or modification of this critical platform state.

Unaffected Components

The following components require no action:

- Calcite
- Knox
- MapReduce
- Spark
- Tez
- Zeppelin

ZooKeeper ACLs Best Practices: Atlas

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Atlas when provisioning a secure cluster.

- ZooKeeper Usage:

- /apache_atlas - Root zookeeper node which is configured for curator, under which nodes for leader election are created.
- /apache_atlas/active_server_info - Znode used in HA environments for storing active server information.
- /apache_atlas/setup_in_progress - Transient Znode used to ensure some setup steps are executed only from one instance. This gets deleted after use and should normally not be seen.
- /leader_elector_path and /setup_lock - Transient Znodes.
- /apache_atlas is the root node which is configuration driven. The default value is /apache_atlas but can be overwritten by setting the value for this property atlas.server.ha.zookeeper.zkroot

- Default ACLs:
  - All znodes have world:anyone:cdrwa by default.
- Security Best Practice ACLs/Permissions and Required Steps:
  - No user intervention is required for creating/using the Znodes. They are all managed internally by Atlas. Atlas exposes two configuration properties that define the auth and ACL - to use while creating these Znodes.

    Cloudera Manager should configure these correctly for a secure cluster. The recommended configuration is atlas.server.ha.zookeeper.auth=sasl:atlas@<domain.com> andatlas.server.ha.zookeeper.acl=sasl:atlas@<domain.com>, where <domain.com> should be replaced with the right value of the Atlas service user principal. (Assuming atlas is the service user name). When set this way, the ACLs for all Znodes will be atlas.server.ha.zookeeper.acl=sasl:atlas@<domain.com>:cdrwa.

    Note: Configuring permissions from Cloudera Manager is not allowed.

    .

ZooKeeper ACLs Best Practices: Cruise Control

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Cruise Control when provisioning a secure cluster.

Cruise Control currently logs into Zookeeper with Kafka credentials. This configuration should not be changed as it will render Cruise Control unable to operate.

A Custom ZooKeeper node, /CruiseControlBrokerList, is created under the root node of Kafka to store the failed broker information in case of Kafka broker failure.

For more information about configuring ZooKeeper ACLs for Kafka, see the ZooKeeper ACLs Best Practices: Kafka section.

Related Information

ZooKeeper ACLs Best Practices: Kafka

ZooKeeper ACLs Best Practices: HBase

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for HBase when provisioning a secure cluster.

- ZooKeeper Usage:
  - /hbase - Default ZNode for unsecured and secured clusters

    Note: The ZNodes do not contain any sensitive information, only metadata.
- Default ACLs:
  - In unsecured setup /hbase - world:anyone:cdrwa

        - All children ZNodes are also world cdrwa
    - Open for global read, write protected: world:anyone:r, sasl:hbase:cdrwa
        - /hbase
        - /hbase/master
        - /hbase/meta-region-server
        - /hbase/hbaseid
        - /hbase/table
        - /hbase/rs
    - No global read, r/w protected: sasl:hbase:cdrwa:
        - /hbase/acl
        - /hbase/namespace
        - /hbase/backup-masters
        - /hbase/online-snapshot
        - /hbase/draining
        - /hbase/replication
        - /hbase/region-in-transition
        - /hbase/splitWAL
        - /hbase/table-lock
        - /hbase/recovering-regions
        - /hbase/running
        - /hbase/tokenauth
    - Security Best Practice ACLs/Permissions and Required Steps:
        - HBase code determines which ACL to enforce based on the configured security mode of the cluster/hbase. Users are not expected to perform any modification of ZooKeeper ACLs on ZNodes and users should not alter any ACLs by hand.

ZooKeeper ACLs Best Practices: HDFS

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for HDFS when provisioning a secure cluster.

- ZooKeeper Usage:
    - hadoop-ha: Default ZNode for unsecured and secured clusters.
- Default ACLs:
    - In an unsecured deployment, the default ACL is world: anyone: cdrwa
    - In a secured deployment, the default ACL is digest: hdfs-fcs: cdrwa
- Security Best Practice ACLs/Permissions and Required Steps:
    - HDFS ZNodes are protected with digest authentication by default in a secure CDP cluster. You need not modify Zookeeper ACLs on HDFS ZNodes or alter any ACLs by hand.

ZooKeeper ACLs Best Practices: Kafka

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Kafka when provisioning a secure cluster.

- ZooKeeper Usage:
    - /controller - Kafka Znode for controller leader election
    - /cluster - Contains a unique id for the Kafka cluster

- /brokers - Kafka Znode for broker metadata
- /kafka-acl - Kafka ZNode for SimpleAclAuthorizer ACL storage
- /admin - Kafka admin tool metadata
- /isr_change_notification - Track changes to In Sync Replication
- /log_dir_event_notification - Node for the broker to notify the controller about log dir events like disk failure
- /delegation_token - Znode for storing delegation tokens
- /controller_epoch - Track movement of controller
- /consumers - Kafka Consumer list
- /config - Entity configuration



Note: The ZNodes do not contain customer data or PII information.

- Default ACLs:
  - /controller - world:anyone:cdrwa
  - /cluster - world:anyone:cdrwa
  - /brokers - world:anyone:cdrwa
  - /kafka-acl - world:anyone:cdrwa
  - /admin - world:anyone:cdrwa
  - /isr_change_notification - world:anyone:cdrwa
  - /log_dir_event_notification -world:anyone:cdrwa
  - /delegation_token - world:anyone:cdrwa
  - /controller_epoch - world:anyone:cdrwa
  - /consumers - world:anyone:cdrwa
  - /config - world:anyone:cdrwa
- Security Best Practice ACLs/Permissions and Required Steps:
  - /controller - sasl:kafka:cdrwa
  - /cluster - sasl:kafka:cdrwa
  - /brokers - sasl:kafka:cdrwa
  - /kafka-acl - sasl:kafka:cdrwa
  - /admin - sasl:kafka:cdrwa
  - /isr_change_notification - sasl:kafka:cdrwa
  - /log_dir_event_notification -sasl:kafka:cdrwa
  - /delegation_token - sasl:kafka:cdrwa
  - /controller_epoch - sasl:kafka:cdrwa
  - /consumers - sasl:kafka:cdrwa
  - /config - sasl:kafka:cdrwa

In a secure Kafka cluster Cloudera recommends that the Enable Zookeeper ACL (zookeeper.set.acl) property is set to true. You can configure this property in Cloudera Manager by going to Kafka Configuration . Once the property is set to true, run the zookeeper-security-migration tool with the zookeeper.acl option set to secure. Finally, reset the ACLs on the root node to allow full access.

For more information, see *Kafka security hardening with Zookeeper ACLs*.

Related Information

Kafka security hardening with ZooKeeper ACLs

ZooKeeper ACLs Best Practices: Oozie

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Oozie when provisioning a secure cluster.

- ZooKeeper Usage:
  - Used to coordinate multiple Oozie servers.
- Default ACLs:

  In a secure cluster, Oozie restricts the access to Oozie Znodes to the oozie principals only using Kerberos backed ACLs.
  - /oozie - node that stores oozie server information in HA mode

  Default ACLs:
  - /oozie - world:anyone:cdrwa
  - /zkdtsm-oozie - node used for handling Oozie delegation tokens when the callback URL authentication is enabled

    ACLs:

    /zkdtsm-oozie - world:anyone:cdrwa
- Security Best Practice ACLs/Permissions and Required Steps:
  - If security is enabled in ZooKeeper, then Oozie connects to ZooKeeper using Kerberos, by default.

Limitations

In Cloudera Base on premises 7.3.2 and lower versions, an Oozie-Curator integration bug causes Oozie to create the /services node without an Access Control List (ACL). Although Oozie does not use this node, the missing ACL triggers validation alerts in the system. To resolve this issue, you must manually apply the ACL using the following command:

```
setAcl /services sasl:oozie:cdrwa
```

ZooKeeper ACLs Best Practices: Ranger

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Ranger when provisioning a secure cluster.

- ZooKeeper Usage:
  - Ranger does not use ZooKeeper directly. Only if Audit to Solr is enabled and Solr is configured in SolrCloud mode, Solr nodes will need to access zookeeper node /ranger_audits.

    /ranger_audits
- Default ACLs:
  - /ranger_audits - world:anyone:cdrwa
- Security Best Practice ACLs/Permissions and Required Steps:
  - Only Solr needs access to this Znode:

    /ranger_audits - sasl:solr:cdrwa
  - After enabling SolrCloud, edit the Ranger collection path permission on Znode:
    1. SSH to the cluster.

       For example, any host where the ZooKeeper server is running.
    2. Start the ZooKeeper CLI by running the following command:

       ```
       zookeeper-client -server [***ZOOKEEPER SERVER HOST***]:[***ZOOKEEPER
        CLIENT PORT***]
       ```

       The default port of zookeeper client port is 2181.

3. After it connects, run: ls /
4. Verify there is a folder for the Ranger Solr collection.
5. Run getAcl /ranger_audits and if the permission is for world,anyone: cdrwa, restrict the permission to “sas l:solr:cdrwa” using this command: setAcl /ranger_audits sasl:solr:cdrwa.
6. Repeat the above step for all clusters where SolrCloud is installed.

```
[zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 0] ls /
          [zookeeper, rmstore, ranger_audits]
          [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 1] getAcl /ran
ger_audits
          'world,'anyone
          : cdrwa
          [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 2] setAcl /ra
nger_audits sasl:solr:cdrwa
          cZxid = 0x200000037
          ctime = Wed Jun 29 10:40:24 UTC 2016
          mZxid = 0x200000037
          mtime = Wed Jun 29 10:40:24 UTC 2016
          pZxid = 0x200000056
          cversion = 7
          dataVersion = 0
          aclVersion = 1
          ephemeralOwner = 0x0
          dataLength = 0
          numChildren = 7
          [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 3] getAcl /rang
er_audits
          'sasl,'solr
          : cdrwa
          [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 4]
```

ZooKeeper ACLs best practices: Search

You must follow certain best practices for effective control of client access to ZNodes using the ZooKeeper access-control lists (ACLs) or permissions for Search when provisioning a secure cluster.

- ZooKeeper Usage
  - Solr uses ZooKeeper to store cluster configuration (document collections, shard and replica configuration / location), cluster status (live servers, overseer metadata), collection configuration (schemas, settings, etc.), and security-related information (delegation tokens). The root ZNode is configurable inCloudera Manager for each Solr service.

    Note: Since users or administrators might upload data into ZNodes, remember that they should not upload sensitive data including PII or passwords in files such as collection configurations. Search content is not stored in ZooKeeper.

    Note: For infrastructure services such as Ranger audits, a dedicated Solr service called Infra Solr is used, where the default ZNode is called /solr-infra.
- Default ACLs

  /solr
  - In a non-kerberized cluster: world:anyone:cdrwa
  - With Kerberos enabled: sasl:solr:cdrwa, world:anyone:r
- Security Best Practice ACLs/Permissions and Required Steps
  - ACLs are initialized during deployment, depending on whether Kerberos for Solr is enabled or not. Operators do not need to modify them by default.

- When creating a new Solr collection, a set of configuration files need to be uploaded to ZooKeeper. Although it is possible to directly upload files (solrctl instancedir commands), because of ACL settings only the solr user is allowed to do this. Cloudera recommends using the Config and ConfigSets APIs (solrctl config commands) instead, which do not require direct ZooKeeper communication. As solrctl config commands communicate directly with Solr where you can configure role-based authorization using Ranger, this method is considered more secure.
- If you disable or enable Kerberos authentication for an existing Solr service in Cloudera Manager, the ACLs in ZooKeeper are not adjusted automatically. In case of disabling Kerberos this can prevent the recently unkerberized Solr service from writing data to ZooKeeper.

  To update ZooKeeper ACLs, you need to perform the following steps:

1. Shut down Solr.

   If you want to disable Kerberos, shut down Solr while Kerberos is still enabled. This allows you to use a valid Solr keytab.
2. Navigate to the /var/run/cloudera-scm-agent/process directory:

   ```
   cd /var/run/cloudera-scm-agent/process
   ```
3. Locate the current or most recent [***PROCESS_DIR_ID***]-solr-SOLR_SERVER directory, with the solr.keytab and jaas.conf files. The [***PROCESS_DIR_ID***] part changes every time the Solr server is restarted.

   To identify the current or most recent directory, list the /var/run/cloudera-scm-agent/process/*solr* directories sorted by time in reverse as follows:

   ```
   ls -ltrd /var/run/cloudera-scm-agent/process/*solr*
   ```

   The entry at the bottom is the current or most recent process directory.
4. Go to the Solr directory:

   ```
   cd [***PROCESS_DIR_ID***]-solr-SOLR_SERVER
   ```
5. Copy log4j2.properties temporarily to /opt/cloudera/parcels/CDH/lib/solr/bin, to get proper logging for the zkcli.sh script.

   ```
   cp log4j2.properties /opt/cloudera/parcels/CDH/lib/solr/bin/
   ```
6. Export the JVM flags for ZooKeeper client to configure ACLs.

   a. To enable Kerberos-specific ACLs:

   ```
   export ZKCLI_JVM_FLAGS="-Djava.security.auth.login.config=jaas.conf
   -DzkACLProvider=org.apache.solr.common.cloud.SaslZkACLProvider -Droo
   t.logger=INFO,console"
   ```

   b. To enable unkerberized ACLs:

   ```
   export ZKCLI_JVM_FLAGS="-Djava.security.auth.login.config=jaas.conf
   -DzkACLProvider=org.apache.solr.common.cloud.DefaultZkACLProvider -D
   root.logger=INFO,console"
   ```

   Note: If Solr ZooKeeper TLS is enabled, append the JVM options documented in Using Search ZooKeeper CLI tools with TLS-enabled ZooKeeper to ZKCLI_JVM_FLAGS.
7. Run the following command to update ACLs:

   ```
   /opt/cloudera/parcels/CDH/lib/solr/bin/zkcli.sh -zkhost [***ZOOKEEPER SE
   RVER HOST***]:[***ZOOKEEPER CLIENT PORT***] -cmd updateacls /solr
   ```

The default [***ZOOKEEPER CLIENT PORT***] is 2181.

8. To verify that the ZooKeeper client uses jaas.conf and ACLs were updated, check the output of the following command:

```
zookeeper-client -server [***ZOOKEEPER SERVER HOST***]:[***ZOOKEEPER CLI
ENT PORT***] getAcl /solr
```

The default [***ZOOKEEPER CLIENT PORT***] is 2181.

For a non-kerberized ACL setup, the command output looks like this:

```
'world,'anyone
: cdrwa
```

For a kerberized ACL setup, the command output looks like this:

```
'sasl,'solr
: cdrwa
'world,'anyone
: r
```

9. Remove log4j2.properties from /opt/cloudera/parcels/CDH/lib/solr/bin:

```
rm /opt/cloudera/parcels/CDH/lib/solr/bin/log4j2.properties
```

ZooKeeper ACLs Best Practices: YARN

YARN related ZooKeeper ACLs are automatically created using Cloudera Manager. Review the list of default ACLs to ensure they are set as recommended for YARN.

Cloudera Manager automatically sets some ZooKeeper ACLs related YARN properties that are used by the YARN service to set up the default ZooKeeper ACLs. That means no manual configuration step is needed. However, customized principles are set by Cloudera Manager only at first launch. For any later launches Clouder Manager checks only the root znodes.

- ZooKeeper Usage:
  - /yarn-leader-election - used for RM leader election
  - /rmstore - used for storing RM application state
- Default ACLs:
  - /yarn-leader-election - sasl:*[***CUSTOMIZED PRINCIPLE UPON FIRST LAUNCH***]*:cdrwa
  - /rmstore - sasl:*[***CUSTOMIZED PRINCIPLE UPON FIRST LAUNCH***]*:cdrwa

If default ACLs are set incorrectly, perform one of the following workarounds:

- Delete the znode and restart the YARN service.
- Use the reset ZK ACLs command. This also sets the znodes below /rmstore/ZKRMStateRoot to world:anyone:cdrwa which is less secure.

ZooKeeper ACLs Best Practices: ZooKeeper

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for ZooKeeper when provisioning a secure cluster.

- ZooKeeper Usage:
  - /zookeeper - Node stores metadata of ZooKeeper itself.
  - /zookeeper/quota - Stores quota information.

    - /zookeeper/config - Stores the current configuration, if the dynamic reconfiguration feature is enabled.
- Default ACLs:
    - /zookeeper - world:anyone:cdrwa
    - /zookeeper/quota - world:anyone:cdrwa
    - /zookeeper/config - world:anyone:r
- Security Best Practice ACLs/Permissions and Required Steps:

  The following steps must be manually performed by users who are using the ZooKeeper quota or dynamic reconfig feature. Components in CDP do not use these features, so these ZNodes are acutally empty by default -- most users do not need to run the following commands:

    - setAcl /zookeeper sasl:zookeeper:rwcda
    - setAcl /zookeeper/quota sasl:zookeeper:cdrwa,world:anyone:r