Cloudera Runtime 7.3.3

Apache Knox Authentication

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

Apache Knox overview.....5
Dynamically generating Knox topology files.....5
Securing access to Hadoop cluster: Apache Knox.....5
Apache Knox Gateway overview.....6
Knox Supported Services Matrix.....6
Knox Topology Management in Cloudera Manager.....8
Considerations for Knox.....9

Proxy Cloudera Manager through Apache Knox.....9

Knox database requirements.....10

Installing Apache Knox.....12
Apache Knox install role parameters.....14

Management of Knox shared providers in Cloudera Manager.....16
Configure Apache Knox authentication for PAM.....16
Configure Apache Knox authentication for AD/LDAP.....17
Use advanced LDAP authentication.....20
Knox CLI testing tools.....21
Configure Apache Knox Authentication for SAML.....22
Add a new shared provider configuration.....22
TLS Mutual Authentication.....23
Generate client certificates for TLS Mutual Authentication.....24
Management of existing Apache Knox shared providers.....24
Add a new provider in an existing provider configuration.....25
Modify a provider in an existing provider configuration.....27
Disable a provider in an existing provider configuration.....28
Remove a provider parameter in an existing provider configuration.....29
Remove a shared provider configuration.....30
Saving aliases.....32
Configuring Kerberos authentication in Apache Knox shared providers.....34
Configuring group mapping in Knox.....36

Management of services for Apache Knox through Cloudera Manager.....38
Enable proxy for a known service in Apache Knox.....38
Disable proxy for a known service in Apache Knox.....40
Add custom service to existing descriptor in Apache Knox Proxy.....41
Add a custom descriptor to Apache Knox.....42
Remove a custom descriptor from Apache Knox.....43
Migrate Knox Gateway service from one host to another.....44
Configure Knox auto-discovery for large clusters.....45
Configuring Knox IDBroker with HashiCorp Vault.....47
Extend the Knox Gateway classpath.....49

Configure Knox for IPv4-only operation....50

Management of Service Parameters for Apache Knox via Cloudera Manager....50

Add custom service parameter to descriptor....50
Modify custom service parameter in descriptor....52
Remove custom service parameter from descriptor....54

Load balancing for Apache Knox....56

Configure load balancing for Apache Hive through Knox....57
Generate and configure a signing keystore for Knox in HA....58

Knox Gateway token integration....58

Overview....59
Token configurations....61
Generate tokens....66
Manage Knox Gateway tokens....68
Knox Token API....70
Manage Knox metadata....74

Configuring Group Impersonation in Knox....75

Knox SSO SAML on FIPS-enabled clusters....76

Knox SSO Cookie Invalidation....77

Configure custom SSO cookie name for Knox....79

Adjust the lifetime of Knox SSO session tokens....80

Concurrent session verification (Tech Preview)....81

Apache Knox overview

Dynamically generating Knox topology files

Topology files can be dynamically generated from combinations of provider configurations and descriptors, which can be defined using Cloudera Manager.

In the early days of Knox, you enabled Knox proxy by editing topology files directly. The topology files consisted of the following things:

- Provider configurations: For example, authentication, federation, authorization, identity assertion, HA, and so on.
- Services: Native Knox services (for example, KNOXTOKEN) or proxied services.

You configured each of the above list in every topology file.

More recently, topology files can be dynamically generated from combinations of provider configurations and descriptors. Additionally, these provider configurations are now shared; you no longer have to specify configurations (for example, authentication provider, identity assertion provider, or authorization provider) for each topology file. You define a provider configuration which can be shared by many descriptors. The following list describes provider configurations, descriptors, and topologies:

- Provider configurations: A named set of providers, for example, authentication, federation, authentication, authorization, identity assertion, and so on. Provider configurations can be shared across descriptors or topologies.
- Descriptors: References the provider configurations to declare the policy (authentication, authorization, identity assertion, and so on) that goes along with proxying that cluster. Descriptors and topologies are 1-to-1.
- Topologies: Dynamically generated based on the descriptors you define.

However, the topologies that are managed by Cloudera Manager should be read-only. Within a Cloudera Manager managed cluster, use Cloudera Manager for creating additional topologies.

Securing access to Hadoop cluster: Apache Knox

The Apache Knox Gateway (“Knox”) is a system to extend the reach of Apache™ Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop Security. Knox also simplifies Hadoop security for users who access the cluster data and execute jobs. The Knox Gateway is designed as a reverse proxy.

Establishing user identity with strong authentication is the basis for secure access in Hadoop. Users need to reliably identify themselves and then have that identity propagated throughout the Hadoop cluster to access cluster resources.

Layers of defense for a Cloudera cluster

- Authentication: Kerberos

  Cloudera uses Kerberos for authentication. Kerberos is an industry standard used to authenticate users and resources within a Hadoop cluster. Cloudera also includes Cloudera Manager, which simplifies Kerberos setup, configuration, and maintenance.
- Perimeter Level Security: Apache Knox

  Apache Knox Gateway is used to help ensure perimeter security for Cloudera customers. With Knox, enterprises can confidently extend the Hadoop REST API to new users without Kerberos complexities, while also maintaining compliance with enterprise security policies. Knox provides a central gateway for Hadoop REST APIs that have varying degrees of authorization, authentication, SSL, and SSO capabilities to enable a single access point for Hadoop.

Cloudera recommends that you leverage the default PAM Authentication Provider for the benefits in performance and ease of administration rather than direct LDAP. For more details, see *Considerations for Knox*.

- Authorization: Ranger

  OS Security: Data Encryption and HDFS

Related Information

Considerations for Knox

Apache Knox Gateway overview

A conceptual overview of the Apache Knox Gateway, a reverse proxy.

Overview

Knox integrates with Identity Management and SSO systems used in enterprises and allows identity from these systems be used for access to Hadoop clusters.

Knox Gateway provides security for multiple Hadoop clusters, with these advantages:

- Simplifies access: Extends Hadoop's REST/HTTP services by encapsulating Kerberos to within the Cluster.
- Enhances security: Exposes Hadoop's REST/HTTP services without revealing network details, providing SSL out of the box.
- Centralized control: Enforces REST API security centrally, routing requests to multiple Hadoop clusters.
- Enterprise integration: Supports LDAP, Active Directory, SSO, SAML and other authentication systems.

Typical security flow: Firewall, routed through Knox Gateway

Knox can be used with both unsecured Hadoop clusters, and Kerberos secured clusters. In an enterprise solution that employs Kerberos secured clusters, the Apache Knox Gateway provides an enterprise security solution that:

- Integrates well with enterprise identity management solutions
- Protects the details of the Hadoop cluster deployment (hosts and ports are hidden from end users)
- Simplifies the number of services with which a client needs to interact

Knox Gateway deployment architecture

Users who access Hadoop externally do so either through Knox, via the Apache REST API, or through the Hadoop CLI tools.

Knox Supported Services Matrix

A support matrix showing which services Apache Knox supports for Proxy and SSO, for both Kerberized and Non-Kerberized clusters.

Table 1: Knox Supported Components

Component	UI Proxy (with SSO)	API Proxy
Atlas API	#	#
Atlas UI	#	#
Beacon		
Cloudera Manager API	#	#
Cloudera Manager UI	#	
Data Analytics Studio (DAS)	#	

Component	UI Proxy (with SSO)	API Proxy
Druid		
Falcon		
Flink		
HBase REST API(aka WebHBase & Stargate)		#
HBase UI	#	
HDFS UI	#	
HiveServer2 HTTP JDBC API (HS2 via HTTP)		#
HiveServer2 LLAP JDBC API		
HiveServer2 LLAP UI		
HiveServer2 UI		
Cloudera Data Explorer (Hue)	#	
Impala HTTP JDBC API		#
Impala UI	#	
JobHistory UI	#	
JobTracker		#
Kudu UI	#	
Livy API + UI	#	#
LogSearch		
NameNode	#	#
NiFi	#	#
NiFi Registry	#	#
Oozie API	#	#
Oozie UI	#	
Ozone	#	
Phoenix (aka Avatica)		#
Profiler	#	
Ranger API	#	#
Ranger UI	#	
Yarn ResourceManager API	#	#
Schema Registry API + UI	#	#
Streams Messaging Manager API	#	#
Streams Messaging Manager UI	#	
Solr	#	#
Spark3History UI	#	
SparkHistory UI	#	
SQL Stream Builder (SSB)	#	#
Storm		
Storm LogViewer		
Superset		

Component	UI Proxy (with SSO)	API Proxy
WebHCat		
WebHDFS		#
YARN UI	#	
YARN UI V2	#	
Zeppelin UI	#	
Zeppelin WS	#	

Note:

APIs, UIs, and SSO in the Apache Knox project that are not listed above are considered Community Features.

Community Features are developed and tested by the Apache Knox community but are not officially supported by Cloudera. These features are excluded for a variety of reasons, including insufficient reliability or incomplete test case coverage, declaration of non-production readiness by the community at large, and feature deviation from Cloudera best practices. Do not use these features in your production environments.

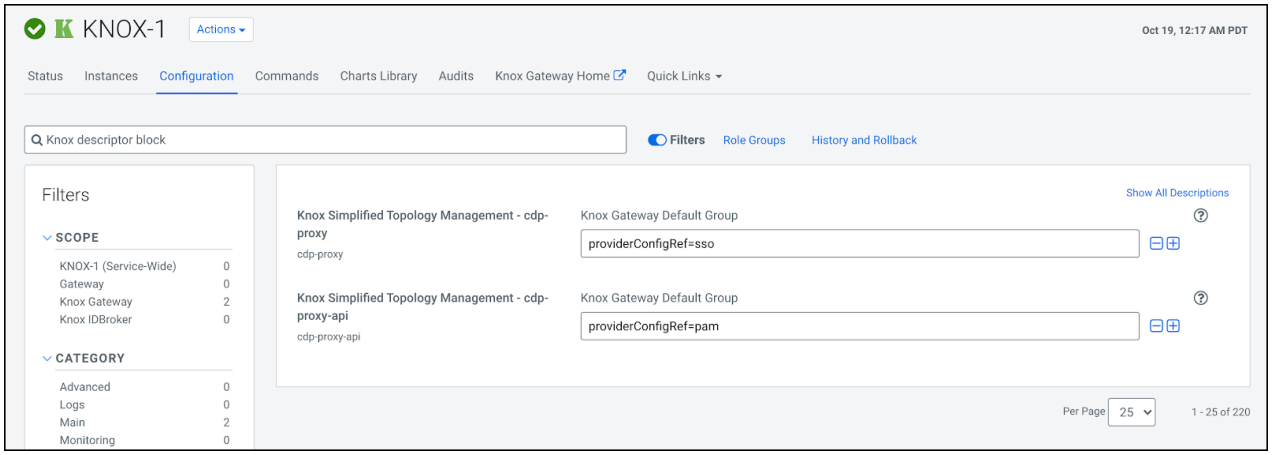
Knox Topology Management in Cloudera Manager

In Cloudera on premises, you can manage Apache Knox topologies via Cloudera Manager using cdp-proxy and cdp-proxy-api.

Shared providers

The Cloudera Manager configurations where the cdp-proxy and cdp-proxy-api topologies can be managed are:

- Knox Simplified Topology Management - cdp-proxy
- Knox Simplified Topology Management - cdp-proxy-api



- The SSO authentication provider is used by the UIs using the Knox SSO capabilities, such as the Home Page UI.
- The API authentication provider is used by predefined topologies, such as admin, metadata or cdp-proxy-api.
- You can add or modify new or existing shared provider configurations.
- You can save aliases using a new Knox Gateway command.

Services

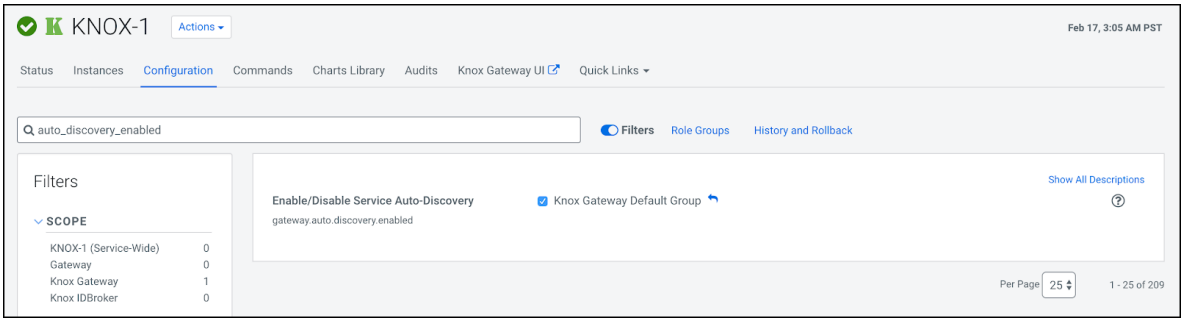
You can enable or disable known or custom services in Knox proxy via Cloudera Manager.

There are two kinds of services in cdp-proxy:

- Known: officially-supported Knox services. Cloudera Manager provides and manages all the required service definition files.
- Custom: unofficial, tech preview, or community feature Knox services. You must supply the service definition files (service.xml and rewrite.xml) that exist in the $CDP_PARCEL_DIR/lib/knox/data/services folder. These are not recommended for production environments, and not supported by Cloudera.

Important:

These topologies will be deployed by Cloudera Manager only if Knox's service auto-discovery feature is turned on using the Enable/Disable Service Auto-Discovery checkbox on Cloudera Manager UI:



Important: Adding a custom service will only work if you provide the service definition files (service.xml and rewrite.xml) in the $CDP_PARCEL_DIR/lib/knox/data/services folder.

Service parameters

You can add, modify, or remove custom service parameters in Knox proxy via Cloudera Manager.

Considerations for Knox

Learn about the considerations before you get started with Knox.

Default PAM settings for Knox

Secure clusters require local OS accounts. Local OS accounts are most often achieved by using something like SSSD or Centrify which localizes user accounts from user stores or directories including LDAP/AD and so on.

This means that you should be able to use PAM and the local OS accounts straight away as long as they are on the same host as Knox. There are various ways to make local OS accounts available. SSSD or Centrify are technical solutions that make it seem like there are local OS accounts even though they are only in LDAP/AD. You can use real local OS accounts as well.

Proxy Cloudera Manager through Apache Knox

In order to have Cloudera Manager proxied through Knox, there are some steps you must complete.

Procedure

1. Set the value for frontend_url: Cloudera Manager Administration Settings Cloudera Manager Frontend URL :
   - Non-HA value: https://$Knox_host:$knox_port
   - HA value: https://$Knox_loadbalancer_host:$Knox_loadbalancer_port
2. Set allowed groups, hosts, and users for Knox Proxy: Cloudera Manager Administration Settings External Authentication :
   - Allowed Groups for Knox Proxy: *

- Allowed Hosts for Knox Proxy: *
- Allowed Users for Knox Proxy: *

3. Enable Kerberos/SPNEGO authentication for the Admin Console and API: Cloudera Manager Administration Settings External Authentication Enable SPNEGO/Kerberos Authentication for the Admin Console and API: : true
4. From Cloudera Manager Administration Settings External Authentication , set Knox Proxy Principal: knox.

What to do next

External authentication must be set up correctly. Cloudera Manager must be configured to use LDAP, following the standard procedure for setting up LDAP. This LDAP server should be the same LDAP that populates local users on Knox hosts (if using PAM authentication with Knox), or the same LDAP that Knox is configured to use (if using LDAP authentication with Knox).

However in cases where no LDAP server is available ,corresponding Cloudera Manager local users can be created and assigned roles manually in Cloudera Manager Administrator Users & Roles Add Local User .

Knox database requirements

Before adding Knox as a service, review the database requirements and supported database types for Apache Knox, including setup requirements by Cloudera Manager version and available database configurations.

A database is only required if you want to use the Knox Token Management feature, specifically for High Availability (HA) deployments where multiple Knox instances need a persistent, shared database to verify tokens. If you install Knox without a database initially, you can still configure the database later if you decide to enable Knox Token Management.

Supported database types

The following database types are supported for Knox:

- **PostgreSQL:** Supported across all Cloudera Manager versions.
- **MySQL:** Supported from Cloudera Manager versions 7.13.1 and 7.13.2 onwards.



Important: Cloudera Manager 7.11.3 does not support MySQL for Knox.

Database setup requirements by Cloudera Manager version

The following table describes the database setup requirements for Knox based on your Cloudera Manager version:

Cloudera Manager version	Cloudera on premises version	Database setup requirement
7.13.2.x	7.3.2, 7.3.1, 7.1.9	Not mandatory
7.13.1.800 and later	7.3.1, 7.1.9	Not mandatory
7.13.1.700 and earlier	7.3.1	**Mandatory**
7.13.1.700 and earlier	7.1.9	Not mandatory
7.11.3	7.1.9	Not mandatory

Knox database configurations

You can set the following initial database configurations for at Cloudera Manager Knox Configuration :

Property	Description
Knox Gateway Database Type	Database type to be used. Valid values are postgresql and mysql. Default is postgresql.
Knox Gateway Database Name	Name of Knox Gateway database. Default is knoxgateway.
Knox Gateway Database Host	Hostname of the database used by Knox Gateway. If the port is non-default for your database type, use host:port notation.
Knox Gateway Database Port	Port for Knox Gateway database. Default is 5432.
Knox Gateway Database User	User for Knox Gateway database. Default is knoxgateway.
Knox Gateway Database User Password	Password for Knox Gateway database.
Knox Gateway JDBC URL Override	Specify JDBC URL override for connecting to Knox Gateway database. If specified, the database properties (host, port, user, password) will be ignored.
Knox Gateway Database - SSL Enabled	Knox Gateway Database SSL enabled. Default is false.
Knox Gateway Database - SSL Verify Server Certificate	Verify Knox Gateway Database SSL server certificate. Default is false.
Knox Gateway Database - SSL Certificate File	Specifies database certificate file location.
Knox Gateway Database - SSL Certificate File Password	The password for Knox Gateway SSL certificate file.

Figure 1: Knox Gateway Database Type configuration

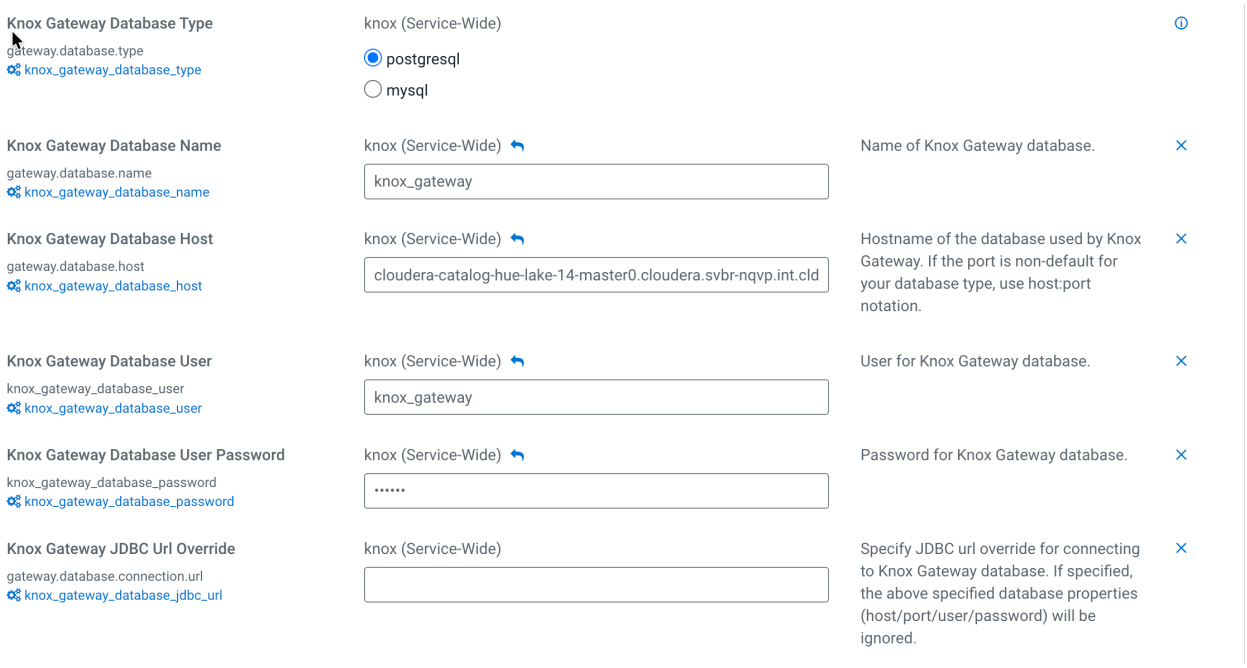
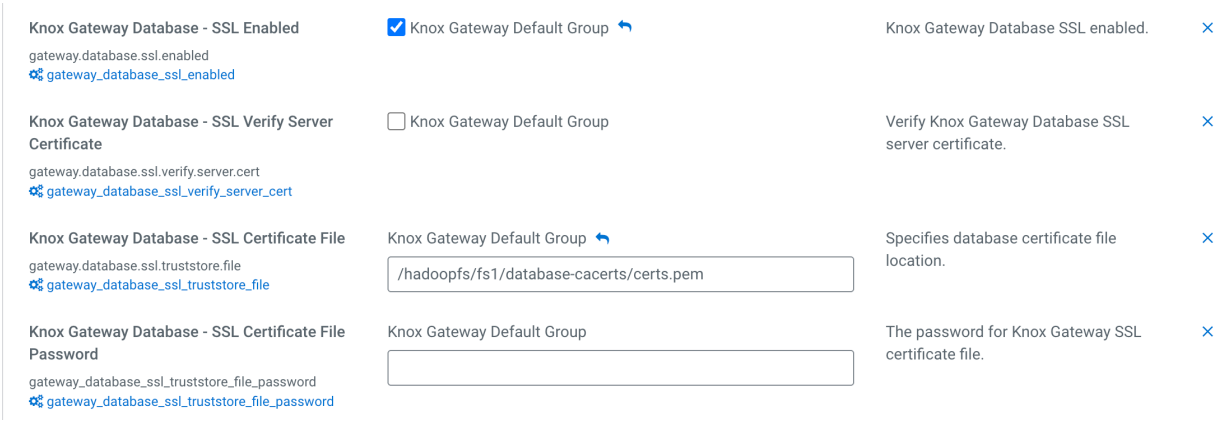


Figure 2: Knox Gateway Database SSL configuration



Related Information

Knox Gateway token integration

Installing Apache Knox

This document provides instructions on how to install Apache Knox using the Cloudera Base on premises installation process.

About this task

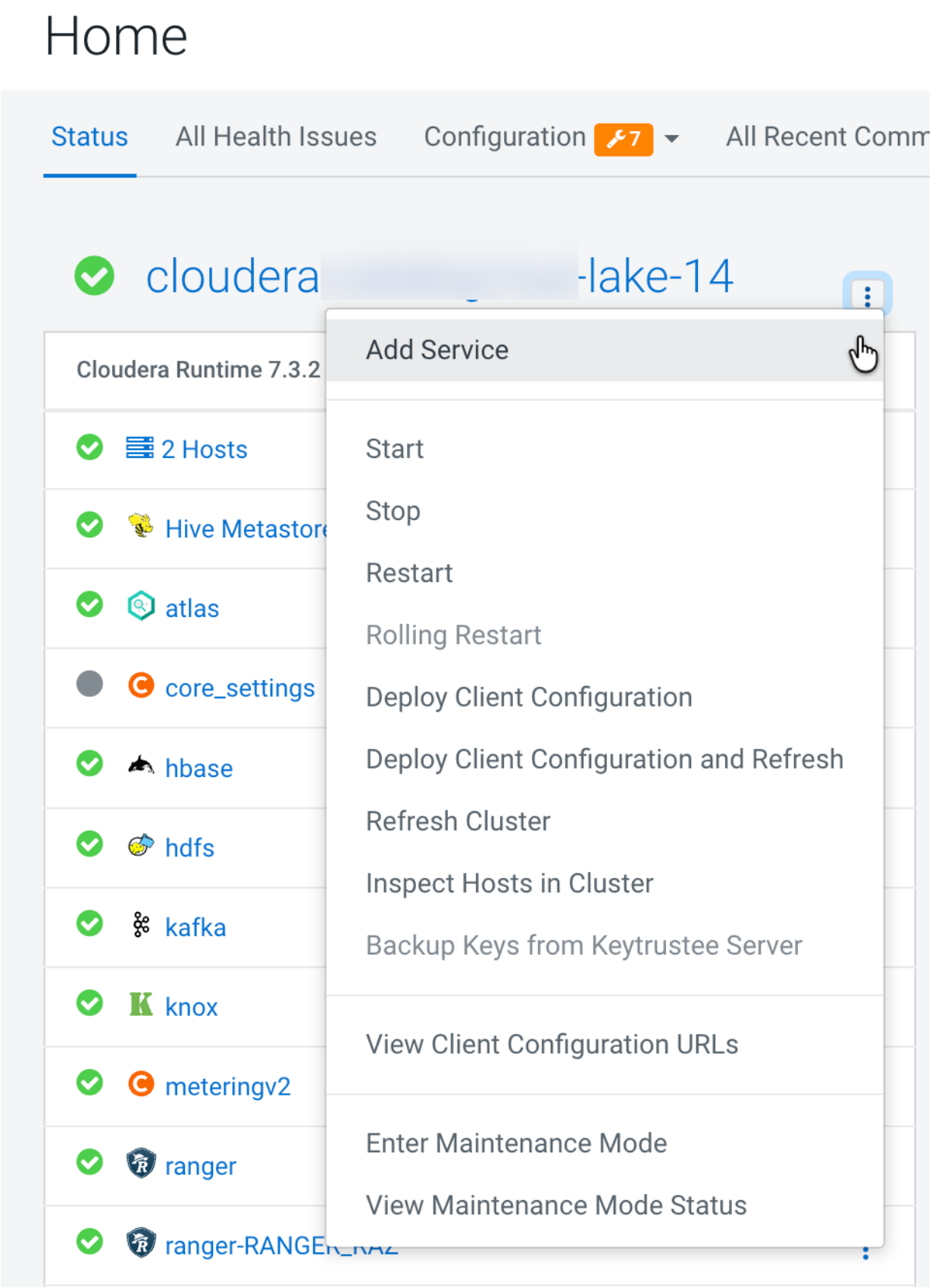
Apache Knox is an application gateway for interacting with the REST APIs and UIs. The Knox Gateway provides a single access point for all REST and HTTP interactions in your Cloudera cluster.

Before you begin

When installing Knox, you must have Kerberos enabled on your cluster.

Procedure

1. From your Cloudera Manager homepage, go to Status tab $Cluster Name ... Add Service



2. From the list of services, select Knox and click Continue.

3. On the **Select Dependencies** page, choose the dependencies you want Knox to set up:

HDFS, Ranger, Solr, Zookeeper	For users that require Apache Ranger for authorization. HDFS with Ranger. HDFS depends on Zookeeper, and Ranger depends on Solr.
HDFS, Zookeeper	HDFS depends on Zookeeper.
No optional dependencies	For users that do not wish to have Knox integrate with HDFS or Ranger.

4. On the **Assign Roles** page, select role assignments for your dependencies and click Continue:

Knox service roles	Description	Required?
Knox Gateway	If Knox is installed, at least one instance of this role should be installed. This role represents the Knox Gateway which provides a single access point for all REST and HTTP interactions with Apache Hadoop clusters.	Required
KnoxIDBroker*	It is strongly recommended that this role is installed on its own dedicated host. As its name suggests this role will allow you to take advantage of Knox's Identity Broker capabilities, an identity federation solution that exchanges cluster authentication for temporary cloud credentials.*	Optional*
Gateway	This role comes with the CSD framework. The gateway structure is used to describe the client configuration of the service on each host where the gateway role is installed.	Optional

* Note: KnoxIDBroker appears in the Assign Roles page, but it is not currently supported in Cloudera Base on premises.

5. On the **Review Changes** page, most of the default values are acceptable, but you must Enable Kerberos Authentication and supply the Knox Master Secret. There are additional parameters you can specify or change, listed in "Knox Install Role Parameters".
   a) Click Enable Kerberos Authentication

      Kerberos is required where Knox is enabled.
   b) Supply the Knox Master Secret, e.g. knoxsecret.
   c) Click Continue.
6. The **Command Details** page shows the status of your operation. After completion, your system admin can view logs for your installation under stdout.

Apache Knox install role parameters

Reference information on all the parameters available for Knox service roles.

Service-level parameters

Table 2: Required service-level parameters

Name	In Wizard	Type	Default Value
kerberos.auth.enabled*	Yes	Boolean	false
ranger_knox_plugin_hdfs_audit_directory	No	Text	${ranger_base_audit_url}/knox
autorestart_on_stop	No	Boolean	false
knox_pam_realm_service	No	Text	login

Knox Gateway role parameters

Table 3: Required parameters for Knox Gateway role

Name	In Wizard	Type	Default Value
gateway_master_secret	Yes	Password	-
gateway_conf_dir	Yes	Path	/var/lib/knox/gateway/conf
gateway_data_dir	Yes	Path	/var/lib/knox/gateway/data
gateway_security_dir	Yes	Path	/var/lib/knox/gateway/data/security
gateway_port	No	Port	8443
gateway_health_port	No	Port	8443[1]
gateway_path	No	Text	gateway
gateway_heap_size	No	Memory	1 GB (min = 256 MB; soft min = 512 MB)
gateway_ranger_knox_plugin_conf_path	No	Path	/var/lib/knox/ranger-knox-plugin
gateway_ranger_knox_plugin_policy_cache_directory	No	Path	/var/lib/ranger/knox/gateway/policy-cache
gateway_ranger_knox_plugin_hdfs_audit_spool_directory	No	Path	/var/log/knox/gateway/audit/hdfs/spool
gateway_ranger_knox_plugin_solr_audit_spool_directory	No	Path	/var/log/knox/gateway/audit/solr/spool

Table 4: Optional parameters for Knox Gateway role

Name	Type	Default Value
gateway_default_topology_name	Text	cdp-proxy
gateway_auto_discovery_enabled	Boolean	true
gateway_cluster_configuration_monitor_interval	Time	60 seconds (minimum = 30 seconds)
gateway_auto_discovery_advanced_configuration_monitor_interval	Time	10 seconds (minimum = 5 seconds)
gateway_cloudera_manager_descriptors_monitor_interval	Time	10 seconds (minimum = 5 seconds)
gateway_auto_discovery_cdp_proxy_enabled_*	Boolean	true
gateway_auto_discovery_cdp_proxy_api_enabled_*	Boolean	true
gateway_descriptor_cdp_proxy	Text Array	Contains the required properties of cdp-proxy topology
gateway_descriptor_cdp_proxy_api	Text Array	Contains the required properties of cdp-proxy-api topology
gateway_sso_authentication_provider	Text Array	Contains the required properties of the authentication provider used by the UIs using the Knox SSO capabilities (such as Home Page UI). Defaults to PAM authentication.
gateway_api_authentication_provider	Text Array	Contains the required properties of the authentication provider used by pre-defined topologies such as admin, metadata or cdp-proxy-api. Defaults to PAM authentication.
gateway_save_alias_command_input	Text	-

[1] Set it to the value of the gateway_port if it is not the default value.

Management of Knox shared providers in Cloudera Manager

Information on Cloudera on premises topology management for Knox from within Cloudera Manager.

- Modifying the SSO authentication provider used by the UIs using the Knox SSO capabilities, such as the Home Page UI.
- Modifying the API authentication provider used by predefined topologies, such as admin, metadata or cdp-proxy-api.
- Adding/modifying new/existing shared provider configurations.
- Saving aliases using a new Knox Gateway command.

Configure Apache Knox authentication for PAM

Knox authentication configurations for PAM in Cloudera Manager. PAM is the default SSO authentication provider in Cloudera on premises.



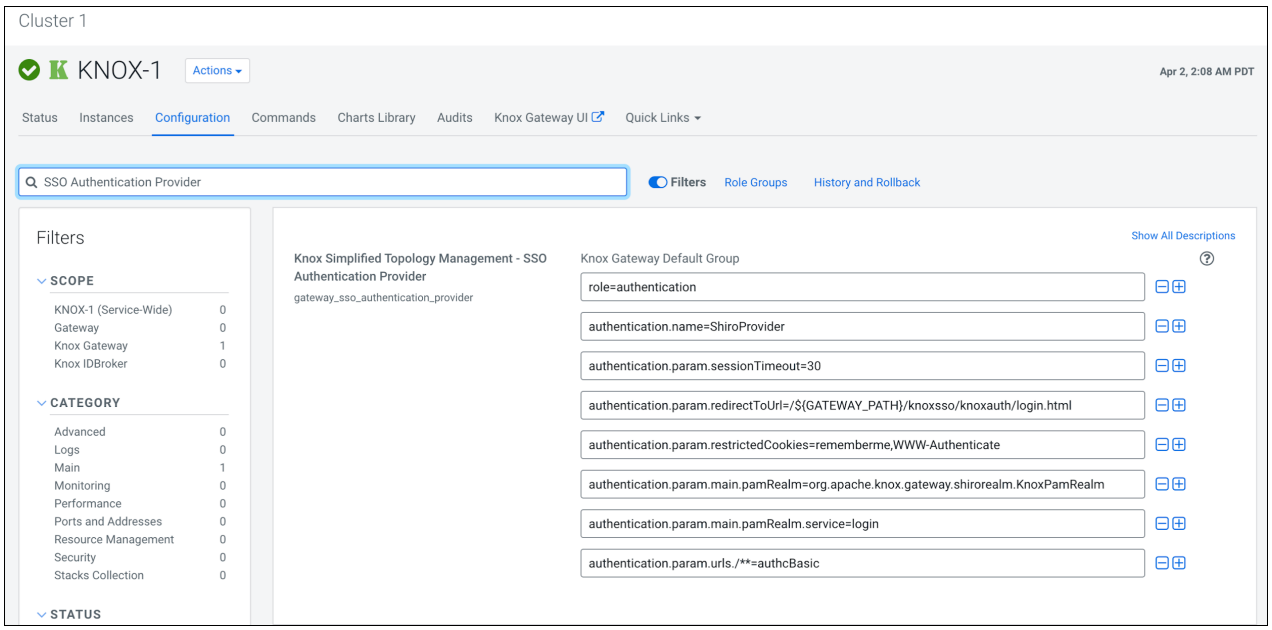
Note: The knox user needs permission on /etc/shadow for PAM authentication. Allow the knox user to read the /etc/shadow file:

```
groupadd shadow
usermod -a -G shadow knox
chgrp shadow /etc/shadow
chmod g+r /etc/shadow
```

SSO authentication for PAM

In Cloudera on premises, Cloudera Manager added a new Knox configuration, called Knox Simplified Topology Management - SSO Authentication Provider, with the following initial configuration:

```
role=authentication
authentication.name=ShiroProvider
authentication.param.sessionTimeout=30
authentication.param.redirectToUrl=/${GATEWAY_PATH}/knoxsso/knoxauth/login.
html
authentication.param.restrictedCookies=rememberme,WWW-Authenticate
authentication.param.urls./**=authcBasic
authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.KnoxP
amRealm
authentication.param.main.pamRealm.service=login
```



Every change here is applied to the knoxsso topology that affects manager, homepage and cdp-proxy topologies as they are using the federation provider.

API authentication for PAM

A new Knox configuration has been added for Cloudera on premises, called Knox Simplified Topology Managemen t - API Authentication Provider, with the following initial configuration:

```
role=authentication
authentication.name=ShiroProvider
authentication.param.sessionTimeout=30
authentication.param.urls./**=authcBasic
authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.Knox
PamRealm
authentication.param.main.pamRealm.service=login
```

Every change here is applied to the admin, metadata, and cdp-proxy-api topologies.

Configure Apache Knox authentication for AD/LDAP

Knox authentication configurations for LDAP and AD in Cloudera Manager.

SSO authentication for AD/LDAP

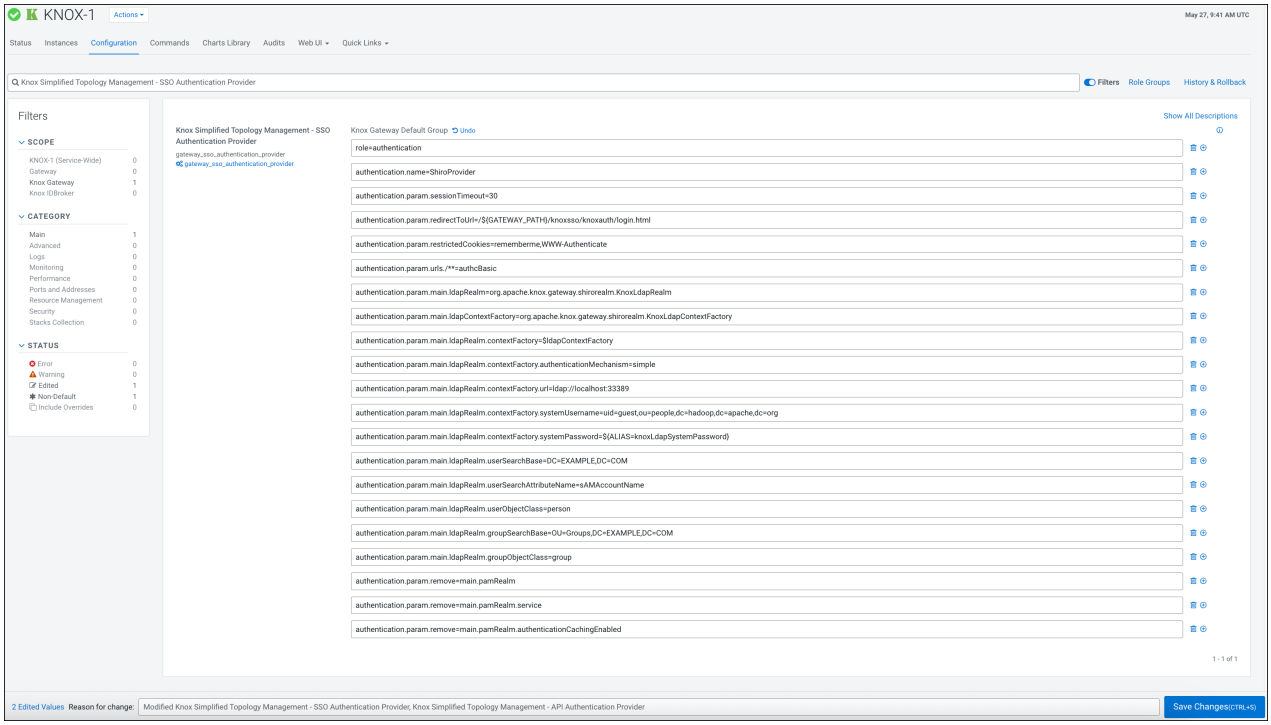
In Cloudera Manager, update Knox Simplified Topology Management - SSO Authentication Provider to change the default PAM authentication to LDAP authentication. The following sample shows how to remove the default PAM-related configuration in ShiroProvider and add LDAP-related properties (for example, with a demo LDAP server configuration):

```
role=authentication
authentication.name=ShiroProvider
authentication.param.sessionTimeout=30
authentication.param.redirectToUrl=/${GATEWAY_PATH}/knoxsso/knoxauth/login.
html
authentication.param.restrictedCookies=rememberme,WWW-Authenticate
authentication.param.urls./**=authcBasic
authentication.param.main.ldapRealm=org.apache.knox.gateway.shirorealm.Knox
LdapRealm
```

```
authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shiro
realm.KnoxLdapContextFactory
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory
authentication.param.main.ldapRealm.contextFactory.authenticationMechanism=s
imple
authentication.param.main.ldapRealm.contextFactory.url=ldap://localhost:33
389
authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=guest,
ou=people,dc=hadoop,dc=apache,dc=org
authentication.param.main.ldapRealm.contextFactory.systemPassword=${ALIAS=k
noxLdapSystemPassword}
authentication.param.main.ldapRealm.userSearchBase=DC=EXAMPLE,DC=COM
authentication.param.main.ldapRealm.userSearchAttributeName=sAMAccountName
authentication.param.main.ldapRealm.userObjectClass=person
authentication.param.main.ldapRealm.groupSearchBase=OU=Groups,DC=EXAMPLE,D
C=COM
authentication.param.main.ldapRealm.groupObjectClass=group
authentication.param.remove=main.pamRealm
authentication.param.remove=main.pamRealm.service
authentication.param.remove=main.pamRealm.authenticationCachingEnabled
```

After you finished editing the properties you have to save the configuration changes. This will make the Refresh Ne eded stale configuration indicator appear. Once the cluster refresh finishes, all topologies that are configured to use Knox SSO will be authenticated by the configured LDAP server.

Figure 3: Knox Simplified Topology Management - SSO Authentication Provider



Note:

As you can see we used a Knox alias when we declared the system password instead of writing the plain text password there. To make it easier for the end-users a new Knox Gateway command was created that allows them to save aliases on all hosts where a Knox Gateway is running. See Saving aliases.

To verify:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/ga
teway/admin/api/v1/providerconfig/knoxsso'
```

```
...
  }, {
    "role" : "authentication",
    "name" : "ShiroProvider",
    "enabled" : true,
    "params" : {
      "main.ldapContextFactory" : "org.apache.knox.gateway.shirorealm.Knox
LdapContextFactory",
      "main.ldapRealm" : "org.apache.hadoop.gateway.shirorealm.KnoxLdapReal
m",
      "main.ldapRealm.contextFactory" : "$ldapContextFactory",
      "main.ldapRealm.contextFactory.authenticationMechanism" : "simple",
      "main.ldapRealm.contextFactory.systemPassword" : "${ALIAS=knoxLdapSys
temPassword}",
      "main.ldapRealm.contextFactory.systemUsername" : "uid=guest,ou=peop
le,dc=hadoop,dc=apache,dc=org",
      "main.ldapRealm.contextFactory.url" : "ldap://localhost:33389",
      "main.ldapRealm.userDnTemplate" : "uid={0},ou=people,dc=hadoop,dc=ap
ache,dc=org",
      "redirectToUrl" : "/${GATEWAY_PATH}/knoxsso/knoxauth/login.html",
      "restrictedCookies" : "rememberme,WWW-Authenticate",
      "sessionTimeout" : "30",
      "urls./**" : "authcBasic"
    }
```

Note: Any change in SSO authentication configuration alters the Knox SSO topology. This affects the manager, homepage, and cdp-proxy topologies because the SSO cookie federation provider is used.

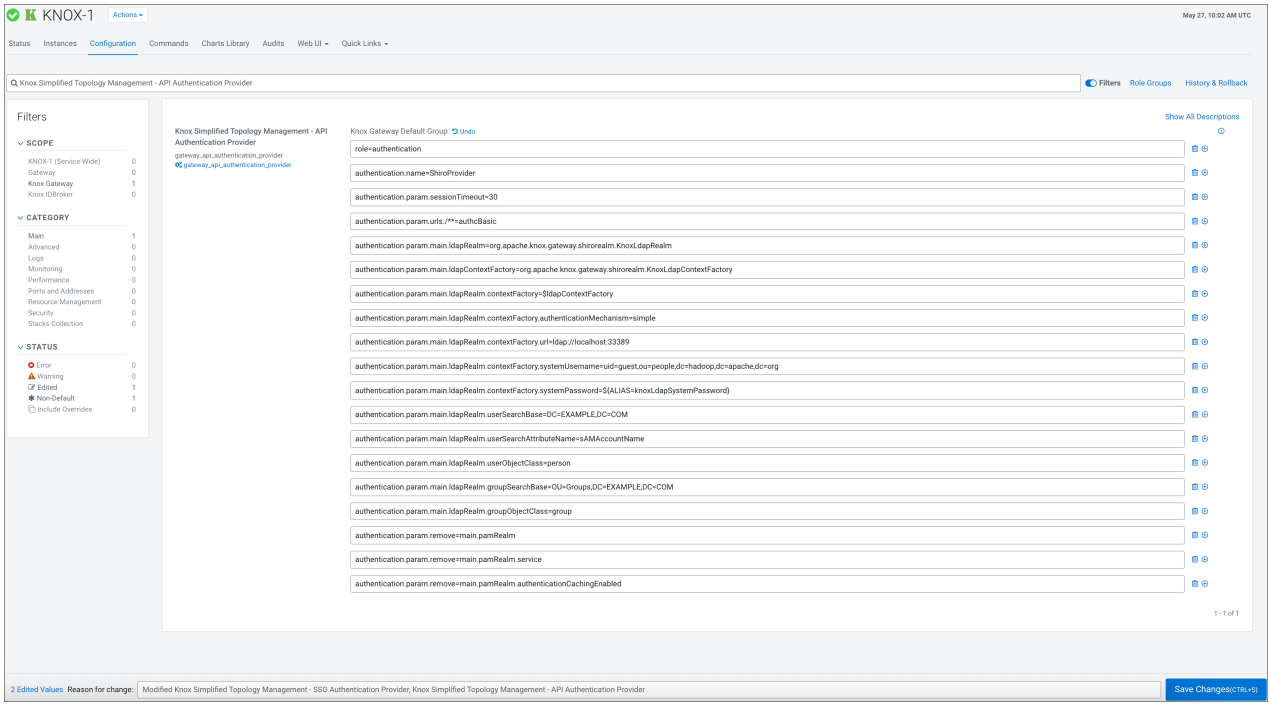
API authentication for AD/LDAP

In Cloudera Manager, update Knox Simplified Topology Management - API Authentication Provider to change the default PAM authentication to LDAP authentication. The following sample shows how to remove the default PAM-related configuration in ShiroProvider and add LDAP-related properties:

```
role=authentication
authentication.name=ShiroProvider
authentication.param.sessionTimeout=30
authentication.param.urls./**=authcBasic
authentication.param.main.ldapRealm=org.apache.knox.gateway.shirorealm.Kno
xLdapRealm
authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shir
orealm.KnoxLdapContextFactory
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory
authentication.param.main.ldapRealm.contextFactory.authenticationMechanism=
simple
authentication.param.main.ldapRealm.contextFactory.url=ldap://localhost:3
3389
authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=guest
,ou=people,dc=hadoop,dc=apache,dc=org
authentication.param.main.ldapRealm.contextFactory.systemPassword=${ALIAS=
knoxLdapSystemPassword}
authentication.param.main.ldapRealm.userSearchBase=DC=EXAMPLE,DC=COM
authentication.param.main.ldapRealm.userSearchAttributeName=sAMAccountName
authentication.param.main.ldapRealm.userObjectClass=person
authentication.param.main.ldapRealm.groupSearchBase=OU=Groups,DC=EXAMPLE,
DC=COM
authentication.param.main.ldapRealm.groupObjectClass=group
authentication.param.remove=main.pamRealm
authentication.param.remove=main.pamRealm.service
authentication.param.remove=main.pamRealm.authenticationCachingEnabled
```

Every change here goes directly into admin, metadata, and cdp-proxy-api topologies.

Figure 4: Knox Simplified Topology Management - API Authentication Provider



Use advanced LDAP authentication

With advanced LDAP authentication, you find the client bind DN by searching the LDAP directory instead of interpolating the bind DN from userDNTemplate.

Example search filter to find the client bind DN is as follows, assuming:

- ldapRealm.userSearchAttributeName=uid
- ldapRealm.userObjectClass=person
- client specified login id = “guest”

LDAP filter for searching the bind DN would be:

```
(&(uid=guest)(objectclass=person))
```

This could find the following bind DN:

```
uid=guest,ou=people,dc=hadoop,dc=apache,dc=org
```

Note: The userSearchAttributeName attribute does not need to be a part of the bind DN. For example, you could use the following:

- ldapRealm.userSearchAttributeName=email
- ldapRealm.userObjectClass=person
- client specified login id = "john_doe@gmail.com"

LDAP filter for searching the bind DN would be:

```
(&(email=john_doe@gmail.com)(objectclass=person))
```

This could find the following bind DN:

```
uid=johnd,ou=contractors,dc=hadoop,dc=apache,dc=org
```

Knox CLI testing tools

Learn how to use the Knox Command Line Interface (CLI) to run diagnostic tests.

The Knox CLI is a command line utility that can be used to manage and test various aspects of a Knox deployment.

You must set the following environment variables before using the Knox CLI:

```
export KNOX_GATEWAY_DATA_DIR="/var/lib/knox/gateway/data"
export KNOX_GATEWAY_CONF_DIR="/var/lib/knox/gateway/conf"
```

The knoxcli.sh command line utility script is located in the /opt/cloudera/parcels/CDH/lib/knox/bin directory.

Invoke the CLI by using the following command:

```
/opt/cloudera/parcels/CDH/lib/knox/bin/knoxcli.sh
```

Knox CLI LDAP authentication and authorization testing

You can use the following command format to authenticate a user name and password against LDAP.

```
bin/knoxcli.sh user-auth-test [--cluster c] [--u username] [--p password] [-
-g] [--d] [--help]
```

This command tests the ability of a topology to connect, authenticate, and authorize a user with an LDAP server. The only required argument is the --cluster argument to specify the name of the topology you want to use. The topology must be valid (passes a validate-topology command). If the –u and –p arguments are not specified, you are prompted for a user name and password.

If authentication is successful, the command attempts to use the topology to do an LDAP group lookup. The topology must be configured correctly to do this. If the topology is not configured correctly, groups are not returned and no errors are printed unless the --g argument is specified. Currently, this command only works if a topology supports the use of ShiroProvider for authentication.

Table 5: LDAP authentication and authorization arguments

Argument	Description	Required?
--cluster	The name of the cluster to authenticate.	Yes
--u	The user name to authenticate with.	No
--p	The password to authenticate with.	No
--g	Specifies that you want to return a user's groups. If not specified, group lookup errors will not be returned.	No

Argument	Description	Required?
--d	Print extra debug information for a failed authentication.	No

Configure Apache Knox Authentication for SAML

Knox authentication configurations for SAML in Cloudera Manager. Knox uses pac4j provider for SAML.

Configuring SAML with the pac4j provider

You can configure SAML in Cloudera Manager through Knox Configuration Knox Simplified Topology Management - SSO Authentication Provider. An example minimal configuration is shown below:

```
role=authentication
authentication.name=ShiroProvider
authentication.enabled=false
role=federation
federation.name=pac4j
federation.param.clientName=SAML2Client federation.param.pac4j.callbackUrl=
https://knox.example.com:8443/gateway/knoxsso/api/v1/websso federation.param
.saml.identityProviderMetadataPath=/etc/knox/conf/idp.xml federation.param.s
aml.serviceProviderMetadataPath=/etc/knox/conf/sp.xml
federation.param.saml.serviceProviderEntityId=knox-sp-entity
authentication.param.remove=main.pamRealm
authentication.param.remove=main.pamRealm.service
```

You can find additional advanced configuration options in the upstream Apache Knox and pac4j documentation.

If the cluster is FIPS-enabled, review Knox SSO SAML on FIPS-enabled clusters before you finalize IdP encryption and signing settings.

You can obtain the Identity Provider (IdP) metadata that Knox needs from your IdP admins. The information required to configure the SAML Identity Provider, including the Knox entity id and AssertionConsumerService endpoint, is contained in the SAML Service Provider (SP) metadata which Knox generates automatically. Once the SP metadata has been written to the serviceProviderMetadataPath on the Knox host, you can send it to the IdP admins to complete the configuration at the IdP side.

Add a new shared provider configuration

Provider configurations are definitions of authentication and authorization controls for services proxied by Knox, which may be referenced by one or more descriptors.

Procedure

1. Define the providers:
   a) From Cloudera Manager Knox Configuration , add a new entry in Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml_role_safety_valve.
   b) Name the provider configuration, and specify the desired providers and their corresponding configuration attributes.

      Provider configuration entries are named as providerConfigs:*TOPOLOGY_NAME* (E.G., providerConfigs: myTopology).
   c) For each provider, the role is declared (E.G., role=authentication, role=authorization), and subsequently configured by defining properties of that role (E.G., authentication.name=ShiroProvider, authentication.param .sessionTimeout=30).

   Example (LDAP authentication and Ranger authorization)

- Name=providerConfigs:ldap-ranger-provider
- Value=

```
role=authentication#
authentication.name=ShiroProvider#
authentication.param.sessionTimeout=30#
authentication.param.main.ldapRealm=org.apache.hadoop.gateway.shirorealm
.KnoxLdapRealm#
authentication.param.main.ldapContextFactory=org.apache.knox.gateway.shi
rorealm.KnoxLdapContextFactory#
authentication.param.main.ldapRealm.contextFactory=$ldapContextFactory#
authentication.param.main.ldapRealm.contextFactory.authenticationMechani
sm=simple#
authentication.param.main.ldapRealm.contextFactory.url=ldap://ldap-ho
st:33389#
authentication.param.main.ldapRealm.contextFactory.systemUsername=uid=
guest,ou=people,dc=hadoop,dc=apache,dc=org#
authentication.param.main.ldapRealm.userDnTemplate=uid={0},ou=people,dc=
hadoop,dc=apache,dc=org#
authentication.param.urls./**=authcBasic#
role=authorization#
authorization.name=XASecurePDPKnox
```

2. Save your changes.
3. Refresh your cluster: the Refresh needed stale configuration indicator appears; click it and wait until the refresh process completes.
4. Validate using the Knox homepage.

   Verify that your topology is generated with the services and URLs you specified.

TLS Mutual Authentication

Mutual authentication with TLS provides the Knox gateway with the means to establish a strong trust relationship with another party. This is especially useful when applications that act on behalf of end-users send requests to Knox.

While this feature does establish an authenticated trust relationship with the client application, it does not determine the end-user identity through this authentication. It will continue to look for credentials or tokens that represent the end-user within the request and authenticate or federate the identity accordingly.

To enable TLS Mutual Authentication, set the following in CM Knox Configuration Knox Service (or Gateway) Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml:

```
gateway.client.auth.needed = true
```

The truststore path for client authentication can be set in Clouder Manager Knox Configuration Knox Service (or Gateway) Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml

```
gateway.truststore.path
```

This parameter can point to the standard truststore (typically truststore.jks) on the host if it contains all of the necessary TrustedCertEntry's for client authentication. Even if the JKS is password protected, Knox can still get TrustedCertEntry content from it, so no password is necessary.

If gateway.client.auth.needed = true and gateway.truststore.path is unset,then it will look at this default location / var/lib/knox/gateway/data/security/keystores/gateway.jks for truststore AND keystore entries, which is an atypical configuration for JKS usage in our stack and not recommended.

These two parameters are distinct, but can point to the same truststore: gateway.truststore.path is for client authentication in the context of TLS Mutual Authentication, and gateway.httpclient.truststore.path is for when the Knox Gateway is an HTTP Client to other TLS servers.

Generate client certificates for TLS Mutual Authentication

Learn how to generate and configure client certificates for TLS Mutual Authentication.

Procedure

1. Generate a private key for the SSL client:

```
openssl genrsa -out client.key 4096
```

2. Create a certificate signing request (CSR) using the client's private key:

```
openssl req -new -key client.key -out client.req
```

3. If you have access to a Certificate Authority (CA) private key (-CAkey), you can sign the certificate using OpenSSL:

```
openssl x509 -req -in client.req -CA ca.cer -CAkey ca.key -set_seria
l [***CERTIFICATE SERIAL NUMBER***] -extensions client -days 365 -outform
 PEM -out client.cer
```



Note: You can customize the -days value to define the certificate's validity period based on your specific requirements.



Important: If you don't have a CA private key, you must submit the CSR to a third-party for signing.

4. Convert the client certificate and private key to PKCS #12 format (.p12) for browser compatibility:

```
openssl pkcs12 -export -inkey client.key -in client.cer -out client.p12
```

5. Import the CA certificate that signed the client certificate into the Java KeyStore (JKS) file located at gateway.trus tstore.path.



Important: If you are using a self-signed client certificate, ensure that you import the client certificate itself into the truststore. In this case, the client certificate acts as its own root CA.

6. Restart Knox to apply the updated configuration.
7. Verify the certificates using OpenSSL:

```
openssl s_client -connect [***HOSTNAME***]:[***PORT NUMBER***] -CA
file [***PATH TO CA CERTIFICATE***] -cert client.cer -key client.key -d
ebug
```

Check the output for any errors. Note that a successful connection does not guarantee successful authentication.

8. Optionally, test the certificates with a curl command to the Knox gateway (using AutoTLS CA certificates):

```
curl https://[***HOSTNAME***]:[***PORT NUMBER***]/gateway/homepage/home -
-cacert /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
 --cert ./client.cer --key ./client.key
```



Note: The default port number for the Apache Knox Gateway is 8443.

Management of existing Apache Knox shared providers

You can add, modify, or disable an existing shared provider configuration in Apache Knox via Cloudera Manager.

The following default shared provider configurations are deployed in Cloudera on premises with Knox:

Table 6: Default shared provider configurations

Configuration	Used by these topologies
admin	admin
homepage	homepage
knoxsso	homepage cdp-proxy manager
manager	manager
metadata	metadata
pam	cdp-proxy-api
sso	cdp-proxy

Note: pam and sso are available only if service auto-discovery is enabled for Knox Gateway role.

The following changes are allowed in any of these shared providers:

- Disable a particular provider
- Modify a particular provider
- Add a new provider

All of these actions can be done via editing the Knox Gateway Advanced Configuration Snippet (Safety Valve) for c onf/cdp-resources.xml by implementing the following language elements:

- The key of a new entry should be like this: providerConfigs: providerConfig_1 [,providerConfig_2,..,provider Config_3]
- The value should contain the following name/value pairs separated by a hash (#) character:

```
role=webappsec|authentication|federation|identity-assertion|authorization|
hostmap|ha
$role.name=ROLE_NAME (e.g. ShiroProvider)
$role.enabled=true|false (optional; defaults to 'true')
$role.param.param_1=value_1 (parameters are optional too)
...
$role.param_N.param1=value_N
```

Add a new provider in an existing provider configuration

An example of how to add a new provider to the authorization provider in the manager shared provider configuration.

About this task

In this example you will see how to add a new HA provider (this time only the ATLAS service will be configured for high availability) in the manager shared provider configuration . This particular authorization provider is set as follows (in its JSON descriptor):

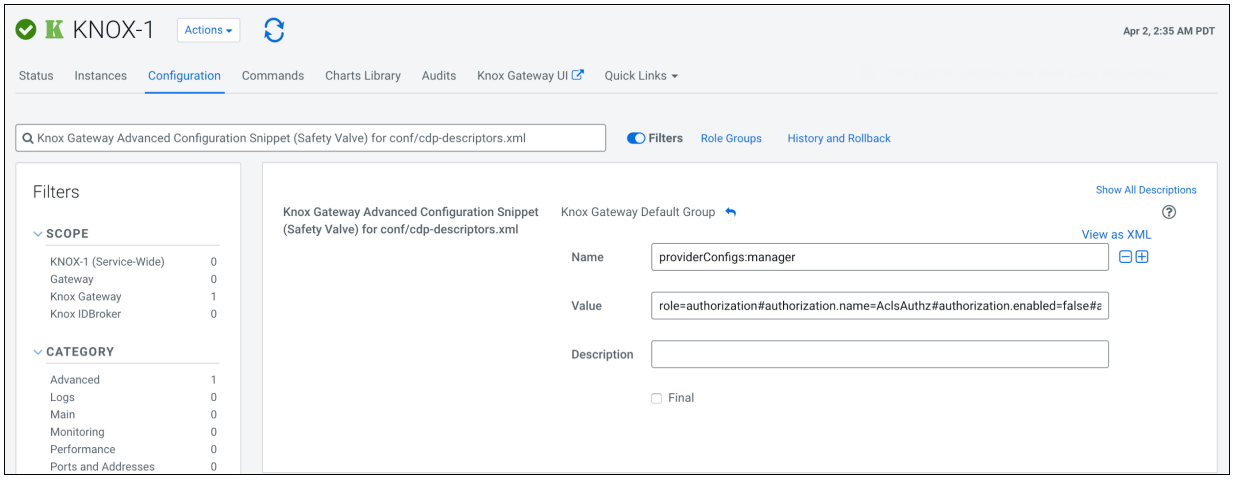
```
{
        "role": "authorization",
        "name": "AclsAuthz",
        "enabled": "true",
        "params": {
           "knox.acl.mode": "OR",
           "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*"
        }
```

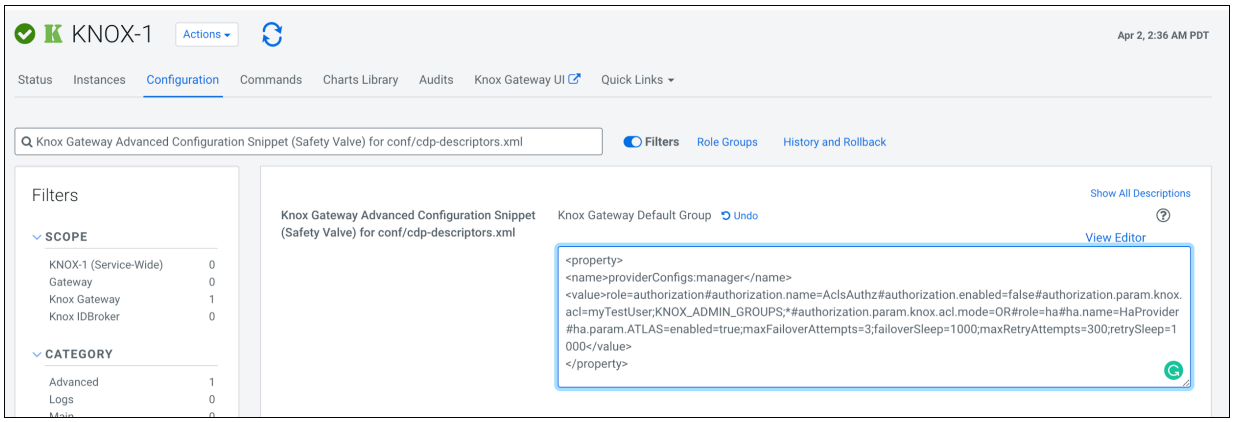
```
        }
```

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Co nfiguration Snippet (Safety Valve) for conf/cdp-resources.xml:
   - name = providerConfigs:manager
   - value =

```
role=authorization#authorization.name=AclsAuthz#authoriz
ation.enabled=false#authorization.param.knox.acl=myTestU
ser;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mod
e=OR#role=ha#ha.name=HaProvider#ha.param.ATLAS=enabled=true;maxFailoverAttempts=3;f
```





2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/
gateway/admin/api/v1/providerconfig/manager'
{
  "providers" : [
 ...
  }, {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
```

```
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;m
axRetryAttempts=300;retrySleep=1000"
    }
  } ]
}
```

Modify a provider in an existing provider configuration

An example of how to modify the authorization provider in the manager shared provider configuration.

About this task

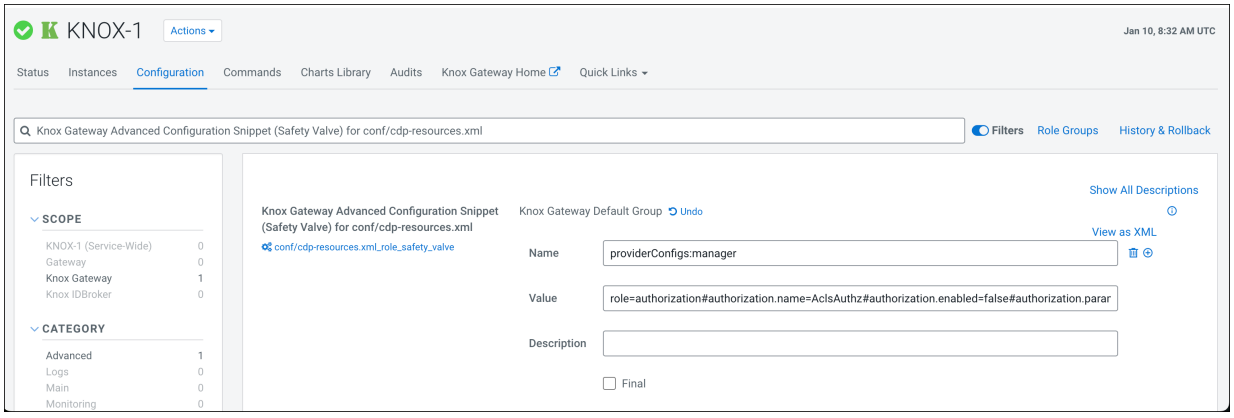
In this example you will see how to modify the authorization provider in the manager shared provider configuration. This particular authorization provider is set as follows (in its JSON descriptor):

```
{
        "role": "authorization",
        "name": "AclsAuthz",
        "enabled": "true",
        "params": {
           "knox.acl.mode": "OR",
           "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*"
        }
     }
```

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Config uration Snippet (Safety Valve) for conf/cdp-resources.xml:
   - name = providerConfigs:manager
   - value =

```
role=authorization#authorization.name=AclsAuthz#authorization.enabled=fa
lse#authorization.param.knox.acl=myTestUser;KNOX_ADMIN_GROUPS;*#authori
zation.param.knox.acl.mode=OR
```



With this change you are authorizing a user called myTestUser to login and execute administrative actions on the Knox Admin API.

2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/
gateway/admin/api/v1/providerconfig/manager'
{
  "providers" : [
 ...
  }, {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;m
axRetryAttempts=300;retrySleep=1000"
    }
  } ]
}
```

Disable a provider in an existing provider configuration

An example of how to disable the authorization provider in the manager shared provider configuration.

About this task

In this example you will see how to disable the authorization provider in the manager shared provider configuration. This particular authorization provider is set as follows (in its JSON descriptor):

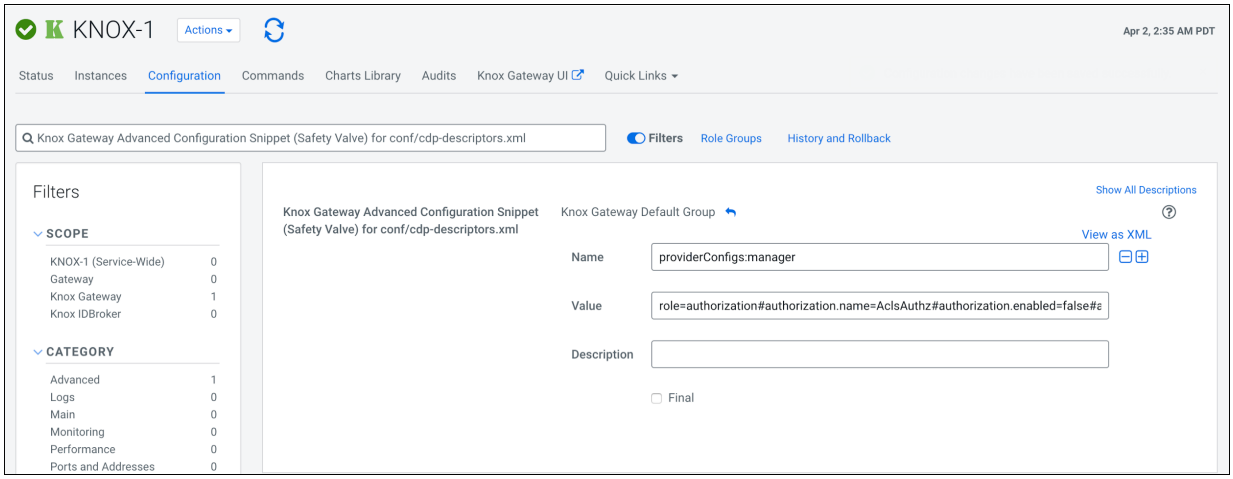
```
{
         "role": "authorization",
         "name": "AclsAuthz",
         "enabled": "true",
         "params": {
            "knox.acl.mode": "OR",
            "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*"
         }
      }
```

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:
   - name = providerConfigs:manager

- value =

```
role=authorization#authorization.name=AclsAuthz#authorization.enable
d=false#authorization.param.knox.acl=KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS
;*#authorization.param.knox.acl.mode=OR
```



2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/
gateway/admin/api/v1/providerconfig/manager'
{
  "providers" : [
 ...
  }, {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;m
axRetryAttempts=300;retrySleep=1000"
    }
  } ]
}
```

What to do next

The only change is that the enabled flag was changed to false.

Remove a provider parameter in an existing provider configuration

An example of how to remove the authentication parameter sessionTimeout from a shared provider configuration.

About this task

In this example you will see how to remove the authentication provider parameter sessionTimeout in the pam shared provider configuration. This particular provider is set as follows:

```
{
  "providers" : [ {
    "role" : "authentication",
    "name" : "ShiroProvider",
    "enabled" : true,
    "params" : {
      "main.pamRealm" : "org.apache.knox.gateway.shirorealm.KnoxPamRealm",
      "main.pamRealm.service" : "login",
      "sessionTimeout" : "30"
    }
  } ],
  "readOnly" : true
}
```

Procedure

1. In Cloudera Manager, select the Knox service.
2. Go to Configuration.
3. Find the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property.
4. Click View as XML and add the following.

```
<property><name>providerConfigs:pam</name><value>role=authentication#aut
hentication.name=ShiroProvider#authentication.param.remove=sessionTimeou
t#authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.
KnoxPamRealm#authentication.param.main.pamRealm.service=login</value></p
roperty>
```



Note: The authentication.param.remove=sessionTimeout specifies the authentication provider parameter to be removed.

5. Save your changes.
6. Refresh the cluster.
7. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/
gateway/admin/api/v1/providerconfig/pam'{
  "providers" : [ {
    "role" : "authentication",
    "name" : "ShiroProvider",
    "enabled" : true,
    "params" : {
      "main.pamRealm" : "org.apache.knox.gateway.shirorealm.KnoxPamRealm",
      "main.pamRealm.service" : "login"
    }
  } ],
  "readOnly" : true
}
```

Remove a shared provider configuration

How to remove a shared provider configuration in Knox through Cloudera Manager.

About this task

You can remove shared provider configurations in Knox when they are no longer required.

Before you begin

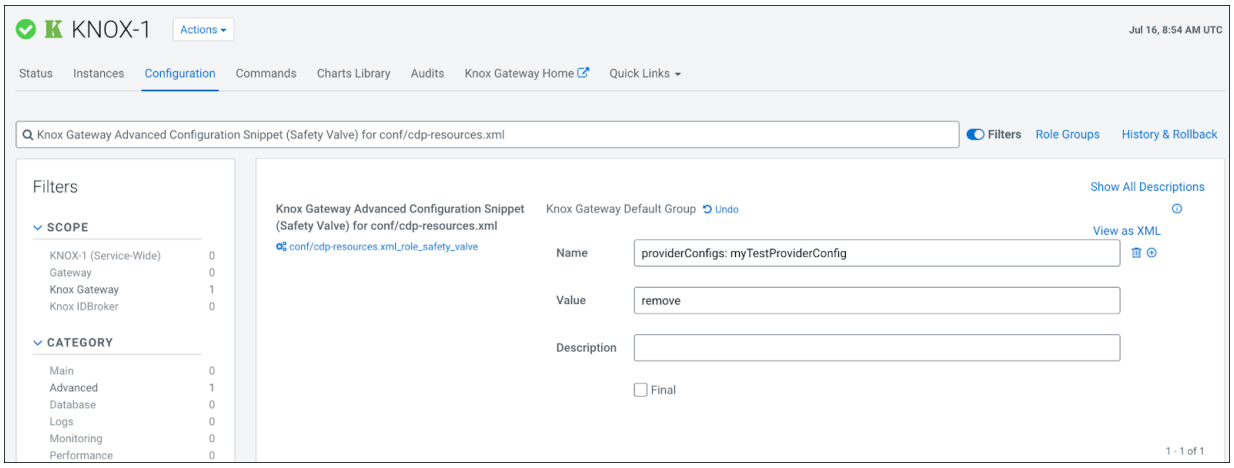
Only non-referenced shared providers can be deleted. If a provider is being used by a descriptor, the provider cannot be deleted.

Procedure

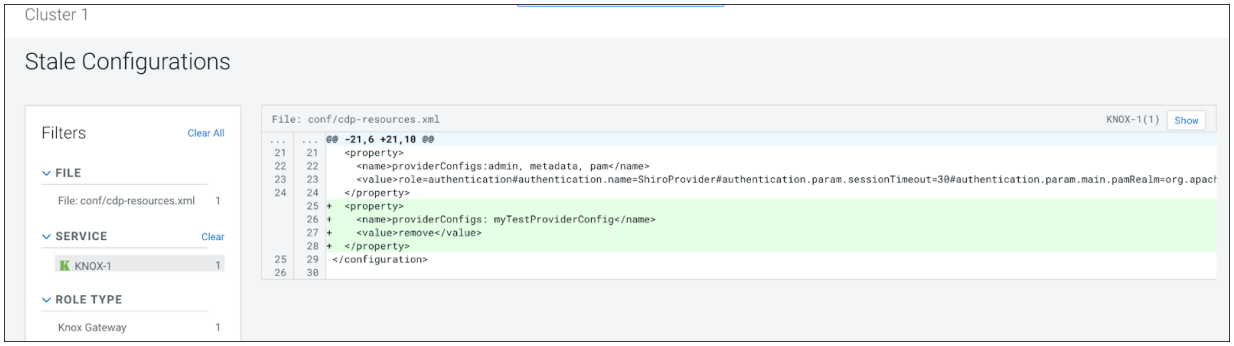
1. In Cloudera Manager, select the Knox service.
2. Go to Configuration.
3. Find the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml advanced configuration snippet.
4. Click the + icon, and add the following entries:
   - Set the Name to providerConfigs:*[***PROVIDER_CONFIGURATION_NAME(S)***]*.

     Note: You can specify multiple provider configuration names to remove by listing them separated with commas.
   - Set the Value to remove.

   In this example, we remove the myTestProviderConfig shared provider.



5. Click the Save Changes(CTRL+S) button.
6. Refresh the Knox instances configuration by clicking the Stale Configuration: Refresh needed indicator and wait until the refresh process completes.



7. Verify that the shared provider configuration file has been removed by running the following command in the Command Line Interface (CLI):

```
ls -al /var/lib/knox/gateway/conf/shared-
providers/[***PROVIDER_CONFIGURATION_NAME***].json
```

8. Check the output. If the file has been successfully removed, you will see the following message:

```
ls: cannot access /var/lib/knox/gateway/conf/shared-
providers/[***PROVIDER_CONFIGURATION_NAME***].json: No such file or
 directory
```

Saving aliases

Learn about saving aliases for the Knox Gateway and IDBroker roles across multiple topologies on each host where an instance of the Knox Gateway or IDBroker is installed, without the need to manually run the Knox CLI tool.

About this task

Two password-type input fields are available at the role level to save aliases:

- gateway_save_alias_command_input: Used for saving Knox Gateway aliases.
- idbroker_save_alias_command_input: Used for saving IDBroker aliases.

Procedure

1. Save Knox Gateway aliases.
   a) Add a Knox Gateway alias using the gateway_save_alias_command_input password-type input field.
   The following format is valid in this input field:

```
topology_name_1[:topology_name_2:...:topology_name_N].alias_name=pas
sword
```

```
cdp-proxy-api:admin:metadata.knoxLdapSystemPassword=guest-password
```

   b) Save the configuration changes.
   c) Run the command using the Knox Gateway Actions Save Alias - Knox Gateway option.



Tip: If you need to add a gateway-level alias, use __gateway as topology name. For example: __gateway.knoxLdapSystemPassword=admin-password.

2. Save IDBroker aliases.
   a) Add an IDBroker alias using the idbroker_save_alias_command_input password-type input field.
   The following format is valid in this input field:

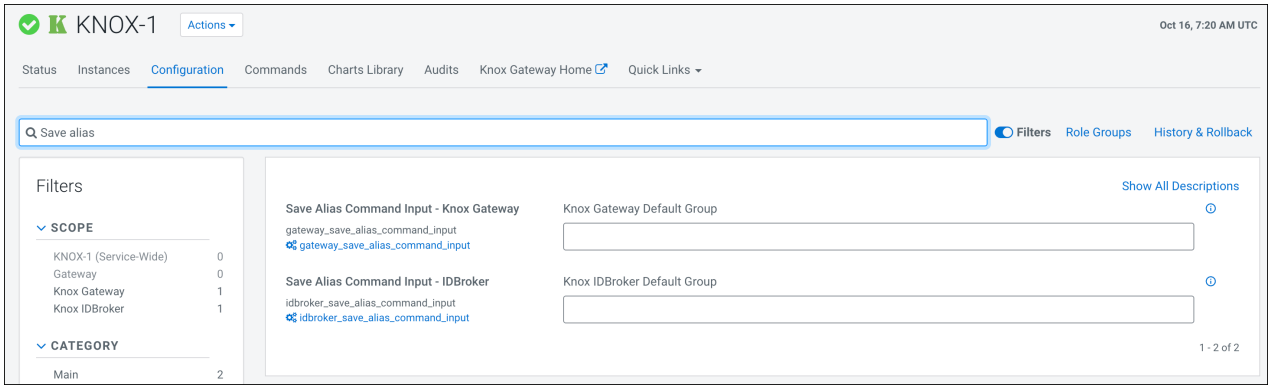
```
topology_name_1[:topology_name_2:...:topology_name_N].alias_name=pas
sword
```

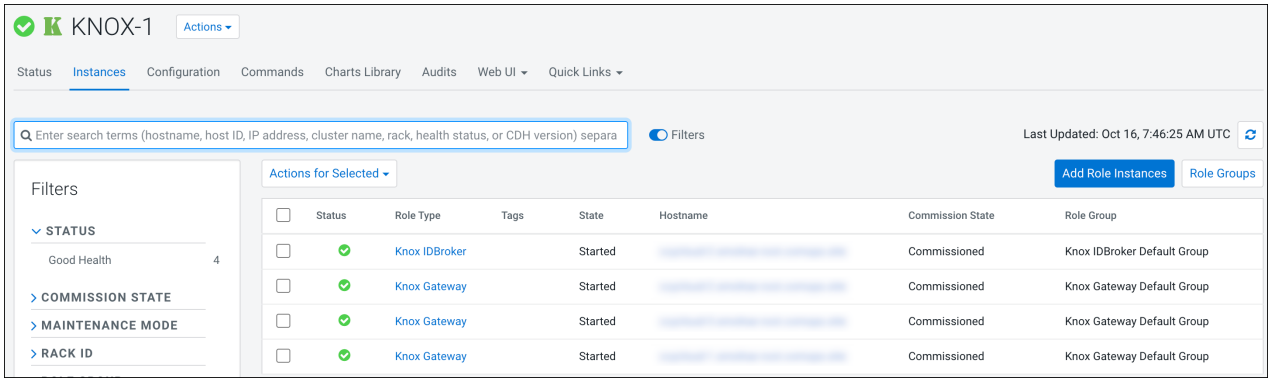
```
aws-cab.aws.credentials.secret=myAwsSecret
```

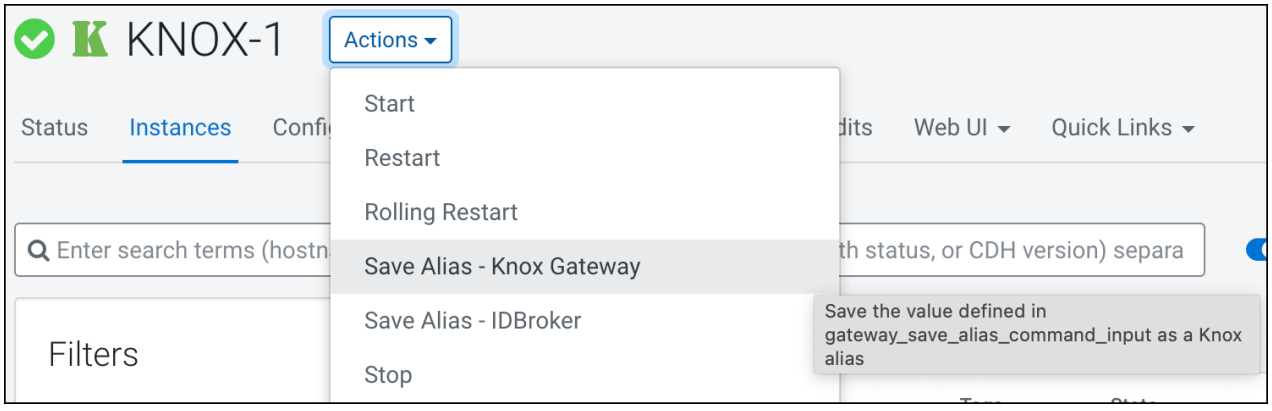
```
azure-cab.azure.storage.account=[***AZURE_STORAGE_ACCOUNT_NAME***]
```

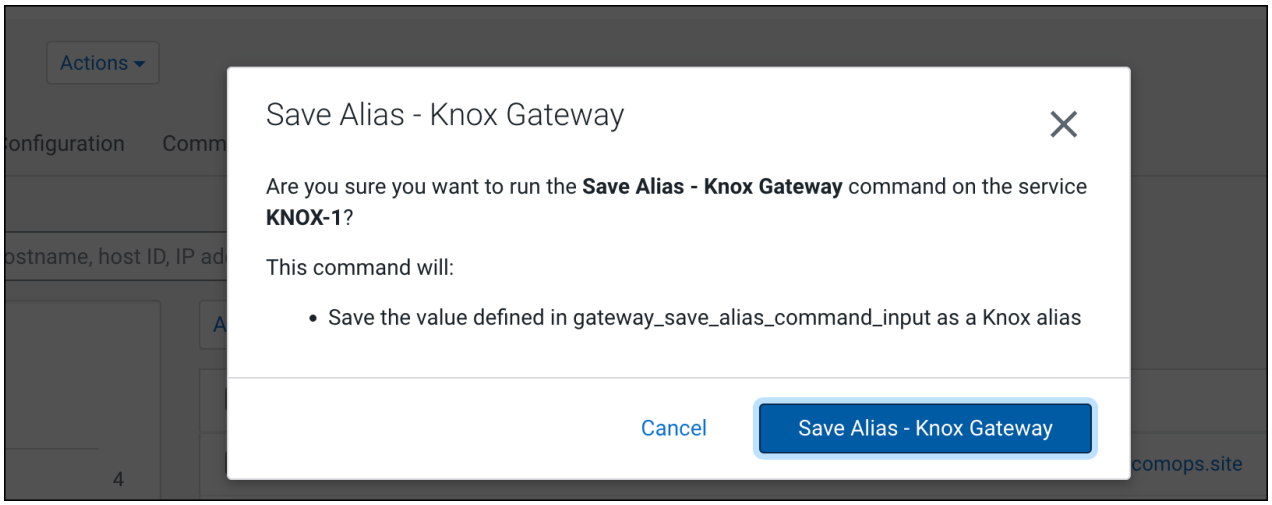
   b) Save the configuration changes.
   c) Run the command using the IDBroker Actions Save Alias - IDBroker option.

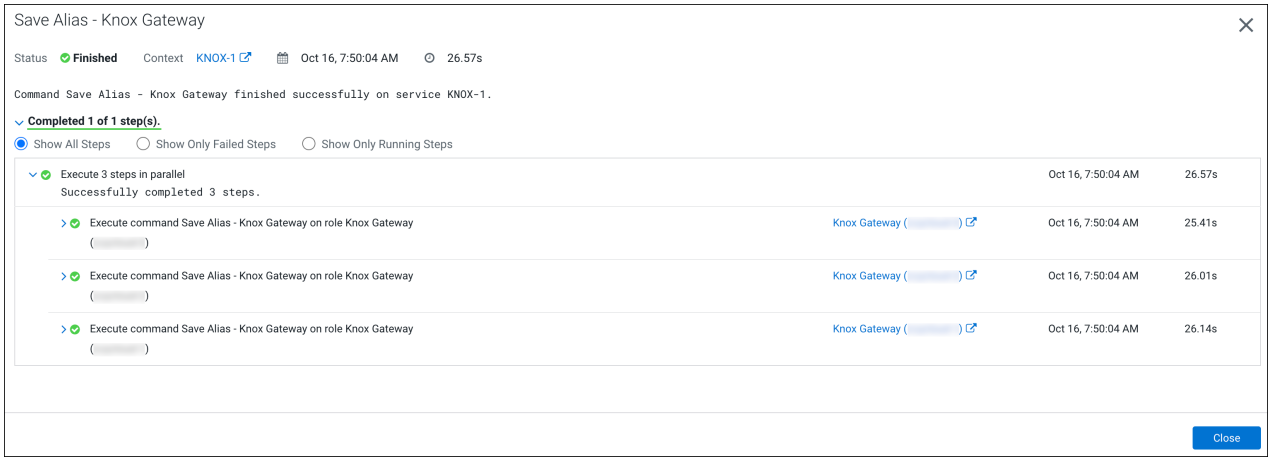
Example

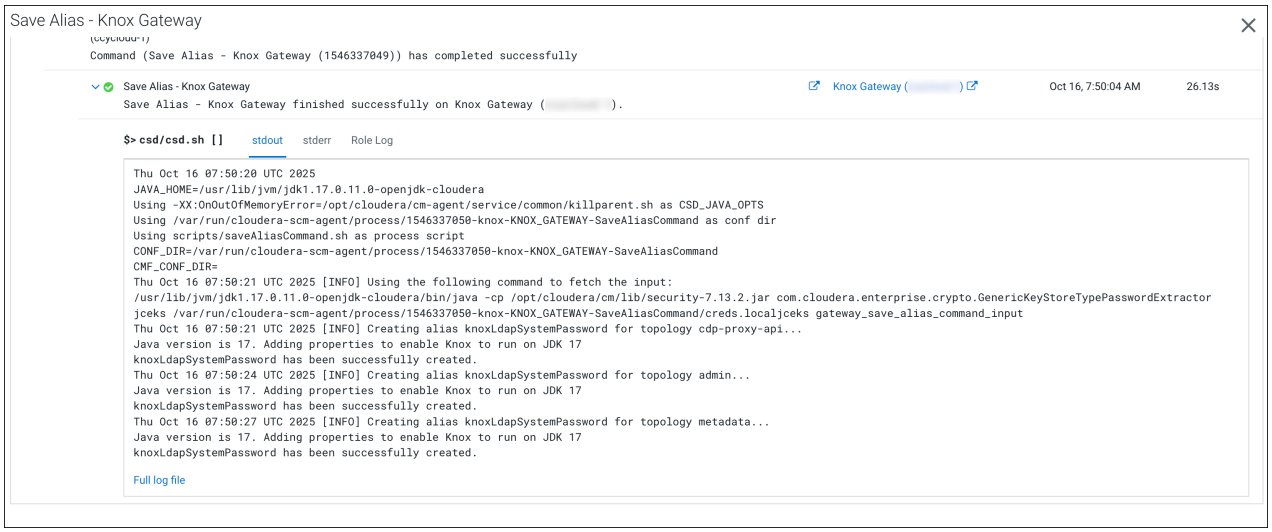

KNOX-1
Actions
Oct 16, 7:20 AM UTC
Status
Instances
Configuration
Commands
Charts Library
Audits
Knox Gateway Home
Quick Links
Save alias
Filters
Role Groups
History & Rollback
Filters
SCOPE
KNOX-1 (Service-Wide) 0
Gateway 0
Knox Gateway 1
Knox IDBroker 1
CATEGORY
Main 2
Show All Descriptions
Save Alias Command Input - Knox Gateway
gateway_save_alias_command_input
gateway_save_alias_command_input
Knox Gateway Default Group
Save Alias Command Input - IDBroker
idbroker_save_alias_command_input
idbroker_save_alias_command_input
Knox IDBroker Default Group
1 - 2 of 2


KNOX-1
Actions
Status
Instances
Configuration
Commands
Charts Library
Audits
Web UI
Quick Links
Enter search terms (hostname, host ID, IP address, cluster name, rack, health status, or CDH version) separa
Filters
Last Updated: Oct 16, 7:46:25 AM UTC
Filters
STATUS
Good Health 4
COMMISSION STATE
MAINTENANCE MODE
RACK ID
Actions for Selected
Add Role Instances
Role Groups
Status
Role Type
Tags
State
Hostname
Commission State
Role Group
Knox IDBroker
Started
Commissioned
Knox IDBroker Default Group
Knox Gateway
Started
Commissioned
Knox Gateway Default Group
Knox Gateway
Started
Commissioned
Knox Gateway Default Group
Knox Gateway
Started
Commissioned
Knox Gateway Default Group


KNOX-1
Actions
Start
Restart
Rolling Restart
Save Alias - Knox Gateway
Save Alias - IDBroker
Stop
Status
Instances
Web UI
Quick Links
Filters
Save the value defined in gateway_save_alias_command_input as a Knox alias


Actions
Save Alias - Knox Gateway
Are you sure you want to run the Save Alias - Knox Gateway command on the service KNOX-1?
This command will:
Save the value defined in gateway_save_alias_command_input as a Knox alias
Cancel
Save Alias - Knox Gateway





Configuring Kerberos authentication in Apache Knox shared providers

An example of how to add the kerberos-auth configuration provider from Cloudera Manager.

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:
   - Name = providerConfigs:kerberos-providers
   - Value =

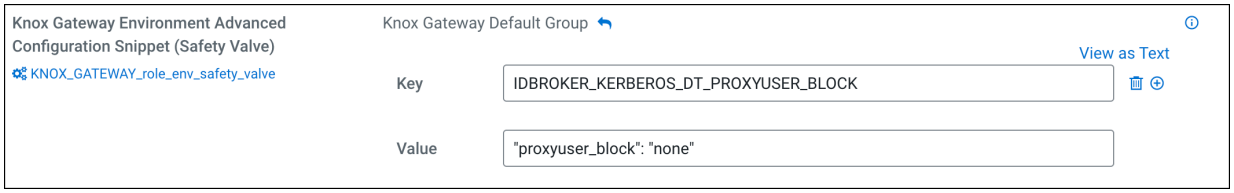
```
role=authentication#
authentication.name=HadoopAuth#
authentication.param.sessionTimeout=30#
authentication.param.config.prefix=hadoop.auth.config#
authentication.param.hadoop.auth.config.type=kerberos#
authentication.param.hadoop.auth.config.signature.secret=${ALIAS=AUTH
_CONFIG_SIGNATURE_SECRET}
authentication.param.hadoop.auth.config.token.validity=1800#
authentication.param.hadoop.auth.config.cookie.path=/#
authentication.param.hadoop.auth.config.simple.anonymous.allowed=false#
authentication.param.hadoop.auth.config.kerberos.principal=AUTH_CONFIG
_KERBEROS_PRINCIPAL#
```

```
authentication.param.hadoop.auth.config.kerberos.keytab=AUTH_CONFIG_KER
BEROS_KEYTAB#
authentication.param.hadoop.auth.config.kerberos.name.rules=DEFAULT
```

Important: Paste the value as a single line, without line-breaks.

2. Add a safety valve name/value pair in Cloudera Manager Knox Configuration ,in Knox Gateway Environment Advanced Configuration Snippet (Safety Valve):

```
Name = IDBROKER_KERBEROS_DT_PROXYUSER_BLOCK
Value = "proxyuser_block": "none"
```



3. Save your changes.
4. Refresh the cluster.
5. Validate with a curl command: curl -k https://*HOST-10-00-100-100*:8443/gateway/admin/api/v1/providerconfig/kerberos-providers.

```
# curl -k https://host-10-00-100-100:8443/gateway/admin/api/v1/providerc
onfig/kerberos-providers
{
  "providers" : [ {
    "role" : "authentication",
    "name" : "HadoopAuth",
    "enabled" : true,
    "params" : {
      "config.prefix" : "hadoop.auth.config",
      "hadoop.auth.config.kerberos.keytab" : "/var/run/cloudera-scm-agent/
process/81-knox-KNOX_GATEWAY/knox.keytab",
      "hadoop.auth.config.kerberos.name.rules" : "DEFAULT",
      "hadoop.auth.config.kerberos.principal" : "HTTP/host-10-00-100-100.
coe.cloudera.com@CLOUDERA.COM",
      "hadoop.auth.config.signature.secret" : "${ALIAS=AUTH_CONFIG_SIGNA
TURE_SECRET}",
      "hadoop.auth.config.simple.anonymous.allowed" : "false",
      "hadoop.auth.config.token.validity" : "1800",
      "hadoop.auth.config.type" : "kerberos",
      "proxyuser_block" : "none"
    }
  }, {
    "role" : "identity-assertion",
    "name" : "HadoopGroupProvider",
    "enabled" : true,
    "params" : {
      "CENTRAL_GROUP_CONFIG_PREFIX" : "gateway.group.config."
    }
  }, {
    "role" : "authorization",
    "name" : "XASecurePDPKnox",
    "enabled" : true,
    "params" : { }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
```

```
    "enabled" : true,
    "params" : {
      "HBASE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "HIVE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true;zoo
keeperEnsemble=maxFailoverAttempts=3;failoverSleep=1000;enabled=true;zoo
keeperEnsemble=gbl20175161.systems.uk.company:2181,gbl20175162.systems.u
k.company:2181,gbl20175163.systems.uk.company:2181;zookeeperNamespace=hi
veserver2",
      "OOZIE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "WEBHCAT" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "WEBHDFS" : "maxFailoverAttempts=3;failoverSleep=1000;maxRetryAtte
mpts=300;retrySleep=1000;enabled=true"
    }
  } ],
  "readOnly" : true
}
```

Related Information

Saving aliases

Configuring group mapping in Knox

Learn how to use HadoopGroupProvider to configure group mapping.

About this task

The role for this provider is identity-assertion and name is HadoopGroupProvider:

```
<provider>
   <role>identity-assertion</role>
   <name>HadoopGroupProvider</name>
   <enabled>true</enabled>
   <<param> ... </param>
</provider>
```

All the configurations for HadoopGroupProvider reside in the provider section of a gateway topology file. The hadoop.security.group.mapping property determines the implementation. Some of the valid implementations are as follows:

- org.apache.hadoop.security.JniBasedUnixGroupsMappingWithFallback

  This is the default implementation and is picked up if hadoop.security.group.mapping is not specified. This implementation determines if the Java Native Interface (JNI) is available. If JNI is available, the implementation uses the API to resolve a list of groups for a user. If JNI is not available then the shell implementation, org.apache.hadoop.security.ShellBasedUnixGroupsMapping is used. It shells out with the `bash -c groups` command (for a Linux/Unix environment) or the `net group` command (for a Windows environment) to resolve a list of groups for a user.
- org.apache.hadoop.security.LdapGroupsMapping

  This implementation connects directly to an LDAP server to resolve the list of groups. However, this should only be used if the required groups reside exclusively in LDAP, and are not materialized on the Unix servers.

To enable group lookup using identity assertion as HadoopGroupProvider, perform the following steps:

Procedure

1. Go to Cloudera Manager Knox Configuration .

2. Search for the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property, and add the following entry:

```
# name = providerConfigs:sso
# value = role=federation#federation.name=SSOCookieProvider#federation.
param.sso.authentication.provider.url=
https://<knox_hostname>:8443/gateway/knoxsso/api/v1/websso#role=identity-
assertion#identity-assertio...
HadoopGroupProvider#identity-assertion.enabled=true#identity-assertion.pa
ram.CENTRAL_GROUP_CONFIG_PREFIX=gateway.group.
config#role=authorization#authorization.name=XASecurePDPKnox#authorizatio
n.enabled=true
```



Note: Replace your Knox hostname in <knox_hostname>.

The following is a sample demo of LDAP configurations for the LDAP setting:

```
<param>
<name>gateway.group.config.hadoop.security.group.mapping</name>
 <value>org.apache.hadoop.security.LdapGroupsMapping</value>

</param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.bind.user</
name>
 <value>uid=tom,ou=people,dc=hadoop,dc=apache,dc=org</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.bind.pass
word</name>
 <value>tom-password</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.url</name>
 <value>ldap://localhost:33389</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.base</name>
 <value></value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.search.fil
ter.user</name>
 <value>(&amp;(|(objectclass=person)(objectclass=applicationProcess))(cn
={0}))</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.search.filt
er.group</name>
 <value>(objectclass=groupOfNames)</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.search.attr
.member</name>
 <value>member</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.search.a
ttr.group.name</name>
 <value>cn</value>
 </param>
 </provider>
```

3. Save your changes.
4. Refresh the cluster.

Management of services for Apache Knox through Cloudera Manager

You can enable or disable known or custom services in Knox proxy through Cloudera Manager.

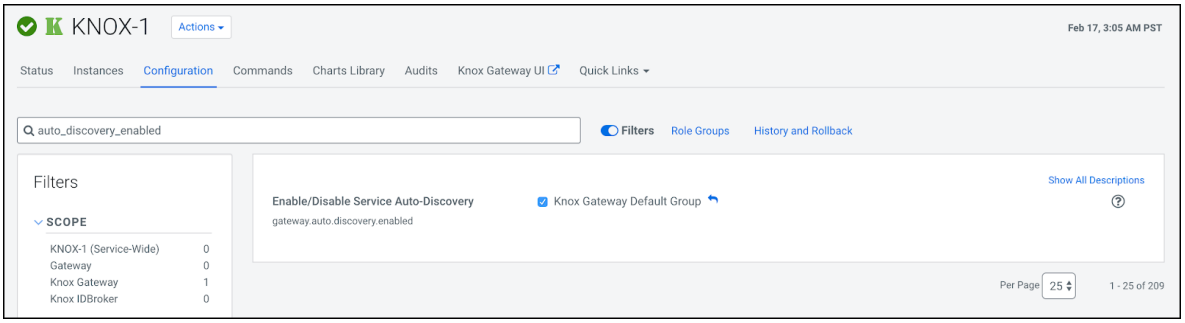
There are two kinds of services in cdp-proxy:

- Known: Officially-supported Knox services. Cloudera Manager provides and manages all the required service definition files.
- Custom: Unofficial, tech preview, or community feature Knox services. You must ensure that the service definition files (service.xml and rewrite.xml) exist in the $CDP_PARCEL_DIR/lib/knox/data/services folder. These are not recommended for production environments, and not supported by Cloudera.

Note: If you upgrade to a newer version of Cloudera, the previously added custom service definitions might disappear from the new location. Therefore, you must ensure that those files are there after upgrading to a newer Cloudera version.

Important:

These topologies will be deployed by Cloudera Manager only if Knox's service auto-discovery feature is turned on using the Enable/Disable Service Auto-Discovery checkbox on Cloudera Manager UI:



For a comprehensive list of known services that can be enabled, see "Knox Supported Services Matrix".

Related Information

Knox Supported Services Matrix

Enable proxy for a known service in Apache Knox

How to enable auto-discovery for a known service in Knox proxy via Cloudera Manager.

About this task

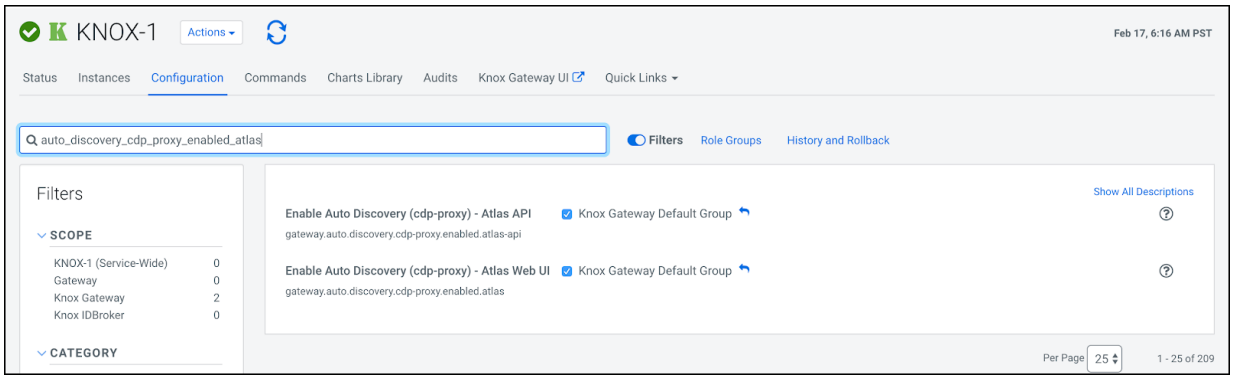
"Known" services are officially-supported Knox services (like Apache Atlas, Ranger, Solr, etc.) Cloudera Manager provides and manages all the required service definition files.

For the purposes of this example, we add ATLAS and ATLAS UI to cdp-proxy. You can add more services; for a comprehensive list of knoxn services that can be enabled, see "Knox Supported Services Matrix".

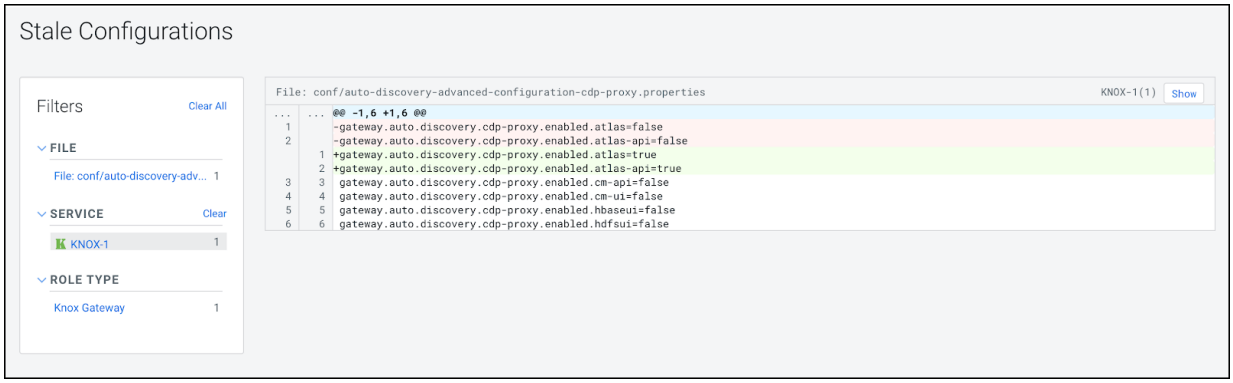
Procedure

1. From Cloudera Manager Knox Configuration , check the Gateway Auto Discovery (cdp-proxy) - $Component boxes.
   In this example, we enable:

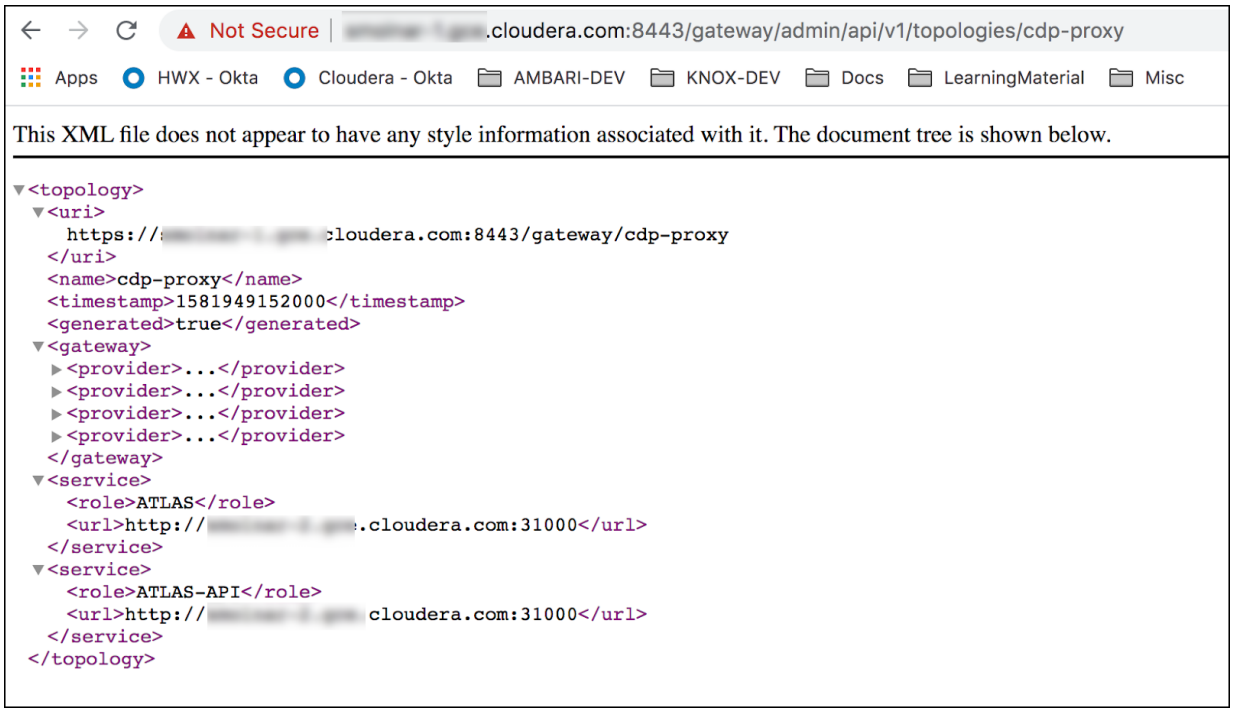
- gateway_auto_discovery_cdp_proxy_enabled_atlas
- gateway_auto_discovery_cdp_proxy_enabled_atlas_ui



2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process finishes.



4. Validate that ATLAS in cdp-proxy was added by going to the following URL: http s://*$KNOX_GATEWAY_HOST*:*$PORT*/*$GATEWAY_PATH*/admin/api/v1/topologies/cdp-proxy.



Related Information

Add custom service parameter to descriptor
Knox Supported Services Matrix

Disable proxy for a known service in Apache Knox

How to remove auto-discovery for a known service in Knox proxy via Cloudera Manager.

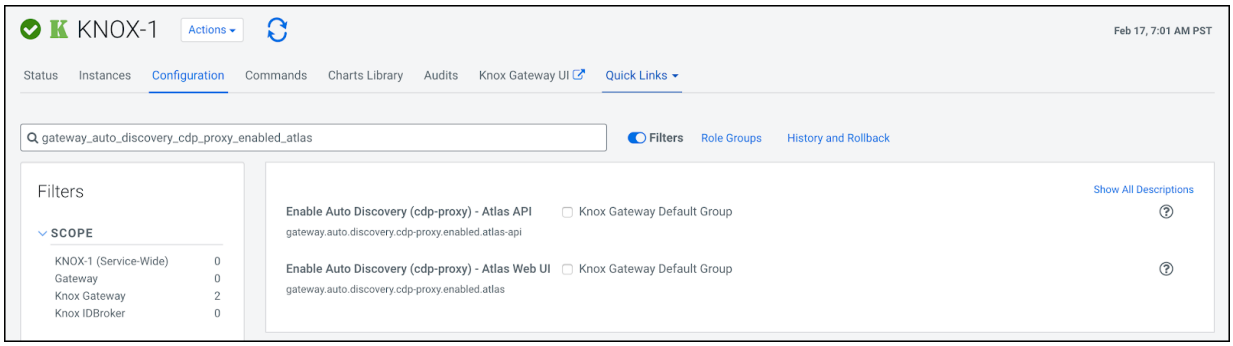
About this task

“Known” services are officially-supported Knox services (like Apache Atlas, Ranger, Solr, etc.) Cloudera Manager provides and manages all the required service definition files.

In this example, we are going to remove the previously added ATLAS and ATLAS-UI services from cdp-proxy. We disable the gateway_auto_discovery_cdp_proxy_enabled_atlas and gateway_auto_discovery_cdp_proxy_enabled_atlas_ui checkboxes on Knox’s Configuration page in CM, save the changes and refresh the cluster.

Procedure

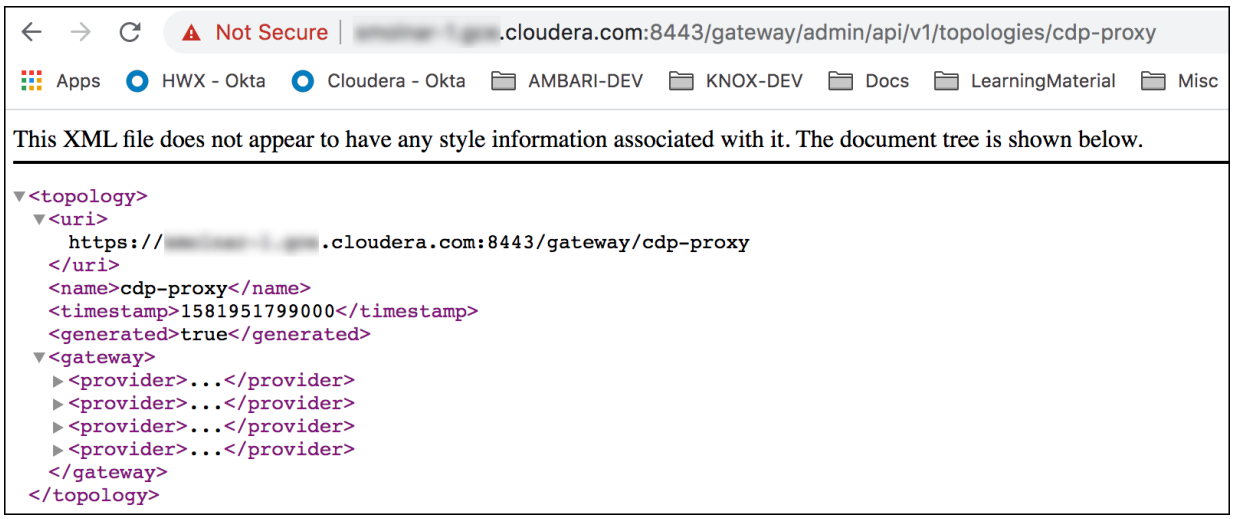
1. From Cloudera Manager Knox Configuration , uncheck the Gateway Auto Discovery (cdp-proxy) - $Component boxes.
   In this example, we disable:
   - gateway_auto_discovery_cdp_proxy_enabled_atlas
   - gateway_auto_discovery_cdp_proxy_enabled_atlas_ui



2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process finishes.



4. Validate that custom service got removed by going to the following URL: http s://*$KNOX_GATEWAY_HOST*:*$PORT*/*$GATEWAY_PATH*/admin/api/v1/topologies/cdp-proxy.



Add custom service to existing descriptor in Apache Knox Proxy

How to add a custom service to an existing descriptor in Knox proxy using Cloudera Manager.

About this task

“Custom” services are unofficial, tech preview, or community feature Knox services. You must supply the service definition files (service.xml and rewrite.xml) which exist in the $CDP_PARCEL_DIR/lib/knox/data/services folder. These are not recommended for production environments, and not supported by Cloudera.

In this example, a custom service (*MY_SERVICE*) is added in cdp-proxy with the following attributes:

- Version : the service’s version, for example, 1.0.0.
- URL: the service URL, for example, https://sampleHost:1234.
- Service parameter: a sample service parameter, for example, myValue.

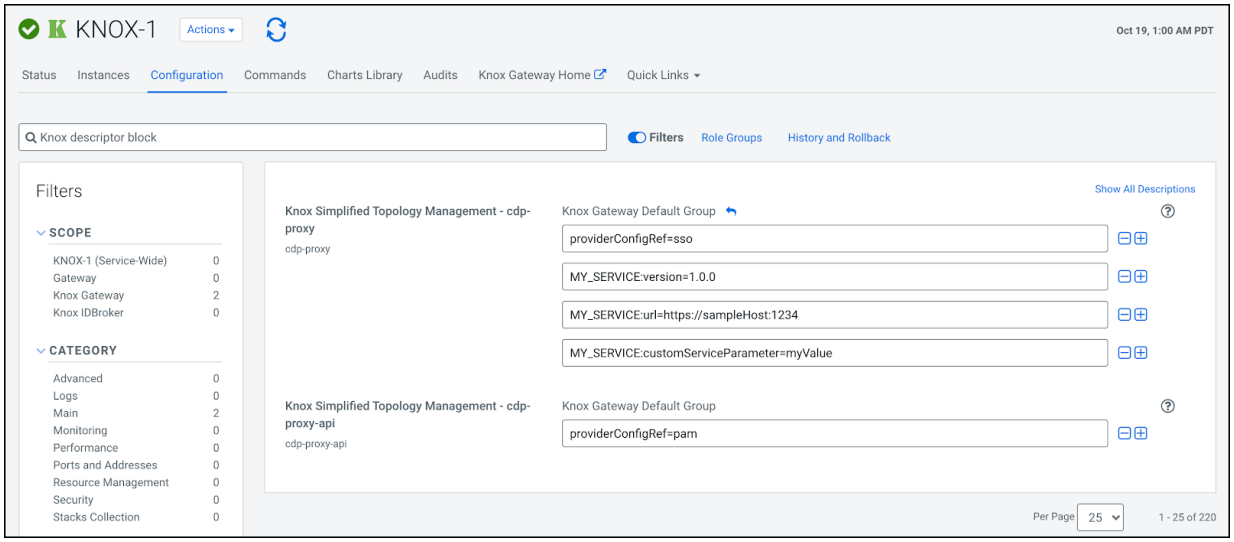
Important: Adding a custom service only works if you provide the service definition files (service.xml and rewrite.xml) in the $CDP_PARCEL_DIR/lib/knox/data/services folder.

To achieve the goals you need to add three new entries with the above-listed parameters in Knox Simplified Topology Management - cdp-proxy. Then you save the changes, refresh the cluster and check if the newly added custom service is available in cdp-proxy.

Procedure

1. From Cloudera Manager Knox Configuration , add the three new entries with the above-listed parameters.

```
MY_SERVICE:version=1.0.0
MY_SERVICE:url=https://sampleHost:1234
MY_SERVICE:customServiceParameter=myValue
```



2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process completes.



4. Validate that MY_SERVICE in cdp-proxy is added by navigating to the following URL: http s://*$KNOX_GATEWAY_HOST*:*$PORT*/*$GATEWAY_PATH*/admin/api/v1/topologies/cdp-proxy.

Add a custom descriptor to Apache Knox

How to add a custom descriptor to Apache Knox using Cloudera Manager.

About this task

Custom descriptors can be deployed to Apache Knox using Cloudera Manager. These descriptors, combined with referenced provider configurations, are transformed into Knox topologies. Using Cloudera Manager means that these descriptors only ever need to be changed in one place to affect all Knox Gateway instances in the cluster.

Fundamentally, descriptors contain the declaration of services to proxy and a reference to provider configuration defining how authentication and authorization for those proxied services should be handled. A descriptor also may similarly declare Knox applications as topologies do.

Service declarations consist of at least the name of the service being proxied. They optionally include one or more endpoint URLs and one or more service-specific parameters.

Descriptors optionally include discovery information, allowing Knox to dynamically discover the endpoint URLs for the declared services.

Procedure

1. Define the descriptor contents:
   a) From Cloudera Manager Knox Configuration , add a new entry in Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml_role_safety_valve.
   b) Name the topology, specify the providerConfigRef, and enumerate the services and associated service URLs.
   Optional service details include version (E.G., HIVE:version=0.13.0) and service parameters (E.G., HIVE:httpclient.connectionTimeout=5m)

   Static URL Example (HIVE and WEBHDFS with PAM authentication)

   - Name=my-custom-topology
   - Value=

   ```
   providerConfigRef=pam#
   HIVE:url=https://hive-host-1:10001/cliservice#
   WEBHDFS:url=https://hdfs-host-1:20470/webhdfs#
   WEBHDFS:url=https://hdfs-host-2:20470/webhdfs
   ```

   Discovery Example (HIVE and WEBHDFS with PAM authentication)

   Note: If the Cloudera cluster is not enabled with Auto-TLS, then you must add the Cloudera Manager certificate to the Knox truststore and restart the Knox service.

   - Name=my-discoverable-topology
   - Value=

   ```
   discoveryType=ClouderaManager#
   discoveryAddress=https://cm-host:7183#
   cluster=Cluster 1#
   providerConfigRef=pam#
   HIVE#
   WEBHDFS
   ```

2. Save the changes.
3. Refresh the Knox instances' configuration: the Refresh needed stale configuration indicator appears; click it and wait until the refresh process completes.
4. Validate using the Knox homepage.
   Verify that your topology is generated with the services and URLs you specified.

   Note: If you want to remove the topology, delete the custom descriptor using the following API call:

   ```
   curl -ku knoxui:knoxui 'https://c2235-node4.coelab.cloudera.com:8443/
   gateway/admin/api/v1/descriptors/infra-solr' -X DELETE
   ```

Remove a custom descriptor from Apache Knox

How to remove a custom descriptor from Knox using Cloudera Manager.

About this task

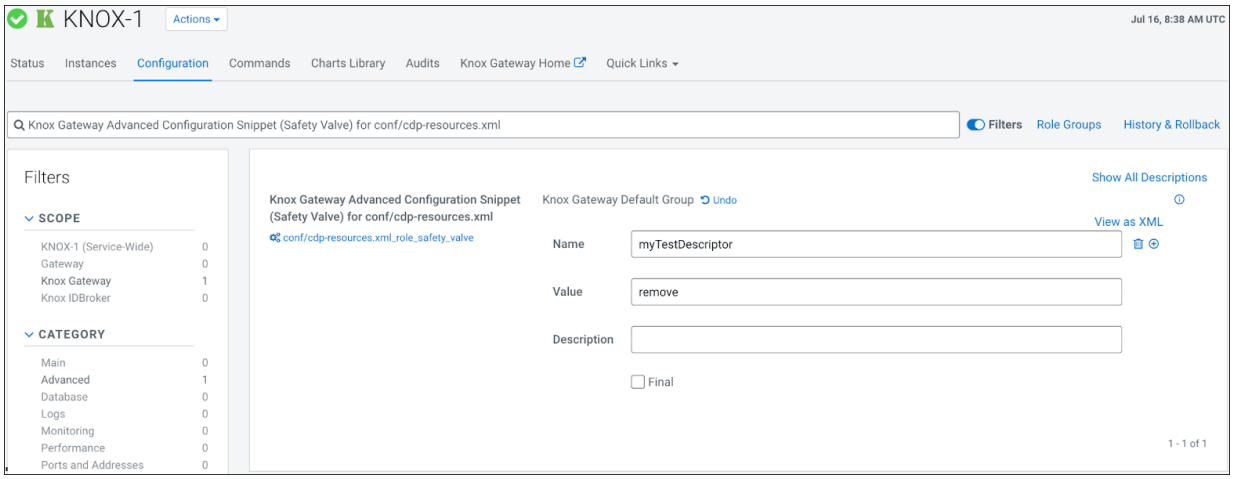
You can remove a custom descriptor from Knox when the descriptor is no longer required.

Procedure

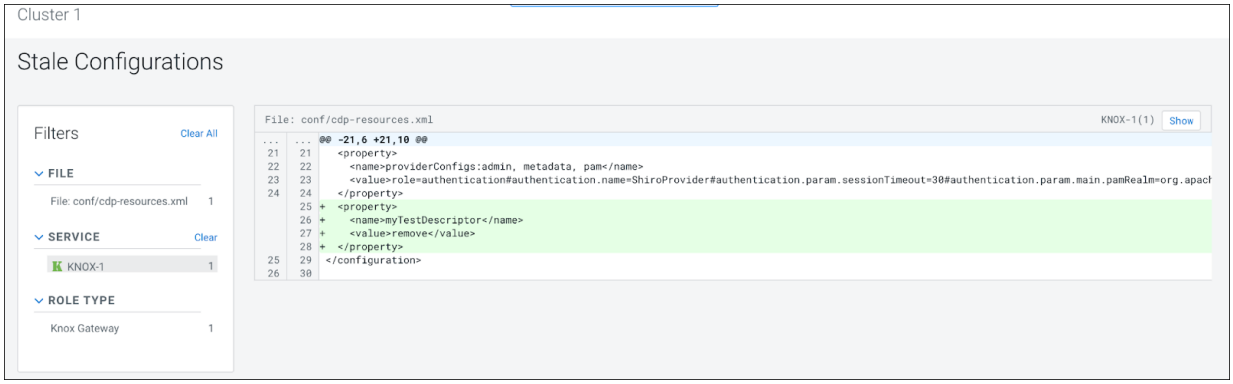
1. In Cloudera Manager, select the Knox service.
2. Go to Configuration.

3. Find the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml advanced configuration snippet.
4. Click the + icon, and add the following entries
   - Set the Name to the name of the descriptor that you want to delete.
   - Set the Value to remove.

   In this example, we remove the myTestDescriptor custom descriptor.



5. Click the Save Changes(CTRL+S) button.
6. Refresh the Knox instances configuration by clicking the Stale Configuration: Refresh needed indicator and wait until the refresh process completes.



7. Repeat the previous steps for any number of descriptors that you want to remove.
8. Verify that the descriptor file has been removed by running the following command in the Command Line Interface (CLI):

```
ls -al /var/lib/knox/gateway/conf/descriptors/[***DESCRIPTOR_NAME***].json
```

9. Check the output. If the file has been successfully removed, you will see the following message:

```
ls: cannot access /var/lib/knox/gateway/conf/
descriptors/[***DESCRIPTOR_NAME***].json: No such file or directory
```

Migrate Knox Gateway service from one host to another

Learn how to migrate the Knox Gateway service from one host to another host in your cluster using Cloudera Manager.

Before you begin

Before migrating the Knox Gateway service, verify if any custom topologies have been manually added to the cluster. If custom topologies exist, convert them to shared provider configurations and descriptors using Knox simplified topology management in Cloudera Manager. This ensures they are automatically replicated to the new host. For detailed instructions, see Add a new shared provider configuration and Add a custom descriptor to Apache Knox.

Procedure

1. In Cloudera Manager, select the Knox service.
2. Click Instances Add Role Instances .
3. Select the new host where you want to migrate the Knox Gateway service.

   When you add the new Knox Gateway role instance, the default topology files are automatically copied to the new host. If you converted custom topologies to shared provider configurations and descriptors using Knox simplified topology management in Cloudera Manager, they are also automatically deployed to the new host.
4. Click Continue.
5. Start the Knox Gateway role on the new host.
6. Test the Knox Gateway functionality on the new host by verifying that you can access Knox Gateway services through the new host and that all topologies and descriptors are functioning correctly.
7. Monitor the Knox Gateway service on the new host for any issues or anomalies. Review the logs and metrics to ensure the service is operating normally after migration.
8. After confirming that the Knox Gateway service is functioning correctly on the new host, remove the Knox Gateway role instance from the old host.
   a) Stop the Knox Gateway role on the old host.
   b) Delete the Knox Gateway role instance from the old host.

Configure Knox auto-discovery for large clusters

How to configure Knox's service auto-discovery feature for large clusters.

Before you begin

By default, the auto-discovery feature of Cloudera Manager through Knox uses the API v32 endpoint:

```
GET /clusters/{clusterName}/services/{serviceName}/roles/{roleName}/config
```

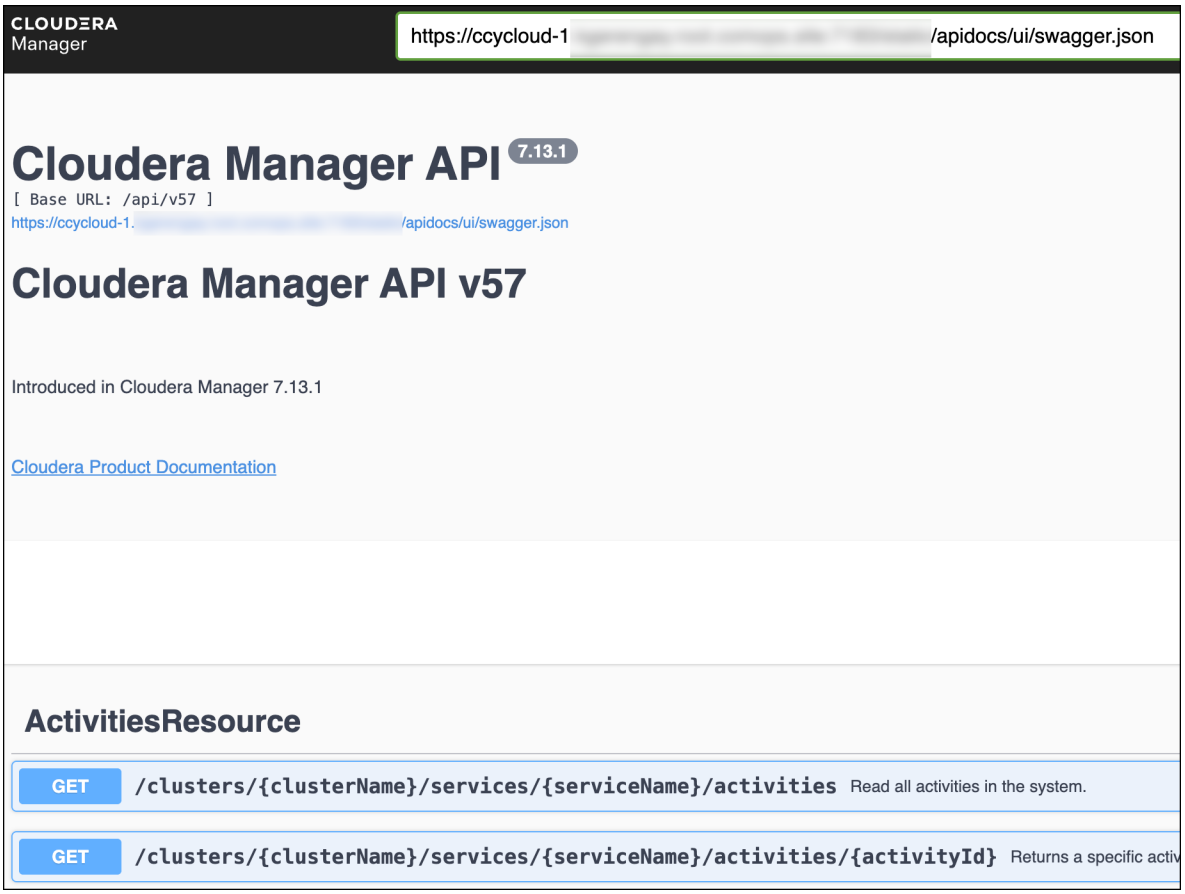
This method fetches the role configuration parameters for each service role. This causes a large number of REST API calls in case of large clusters.

For large clusters, use the following endpoint:

```
GET /clusters/{clusterName}/services/{serviceName}/roles/configs
```

Note: The endpoint is available in the Cloudera Manager API with Cloudera Manager version 7.13.1 using API v57.

You can check the API endpoint and the Cloudera Manager version through Cloudera Manager Support API Explorer :



If the endpoint is available in your Cloudera Manager, follow these steps to configure the auto-discovery method in Knox:

Procedure

1. From Cloudera Manager Knox Configuration , add a new entry in Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml, and insert these properties:

```
gateway.cloudera.manager.service.discovery.role.fetch.strategy=byService
gateway.cloudera.manager.service.discovery.api.version=v57
```

2. Save the changes.
3. Refresh the Knox instances' configuration: the Refresh needed stale configuration indicator appears; click it and wait until the refresh process completes.
4. Validate using the Knox homepage.
5. The default auto-discovery method can be reset by one of the following methods:
   - Remove the properties from Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml
   - Insert these properties into Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml:

```
gateway.cloudera.manager.service.discovery.role.fetch.strategy=byRole
```

```
gateway.cloudera.manager.service.discovery.api.version=v32
```

Configuring Knox IDBroker with HashiCorp Vault

Learn how to configure Knox IDBroker with HashiCorp Vault to securely manage AWS credentials.

About this task

Knox IDBroker can be configured with HashiCorp Vault to enhance AWS credentials management. HashiCorp Vault enables IDBroker to authenticate with AWS using short-lived credentials from Vault instead of storing long-lived credentials for this purpose.

Before you begin

- You must install and properly configure HashiCorp Vault.
- An AWS account must be available with the required IAM roles.
- You must establish network connectivity between Knox and the HashiCorp Vault server.

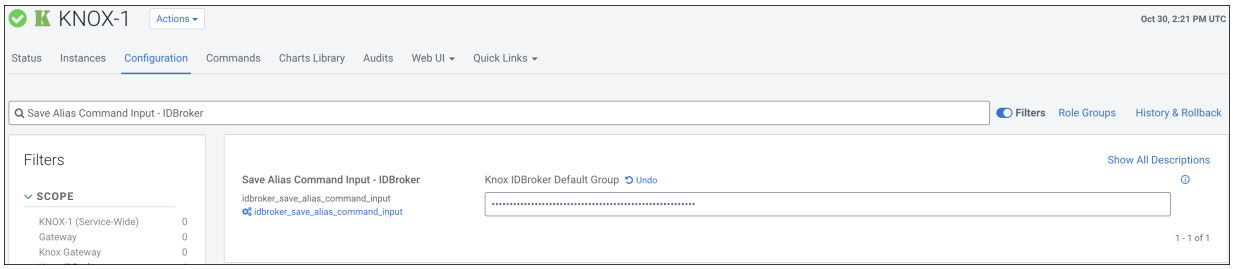
Procedure

1. In Cloudera Manager, select the Knox service.
2. Go to the Configuration tab.
3. Search for the Save Alias Command Input - IDBroker property.
4. Enter the following parameter:

   ```
   [***TOPOLOGY NAME***].vaultTokenAlias=[***VAULT TOKEN***]
   ```

   Replace *[***TOPOLOGY NAME***]* with the name of your topology, and *[***VAULT TOKEN***]* with the specific token for the vault.

   Figure 5: Save Alias Command Input - IDBroker

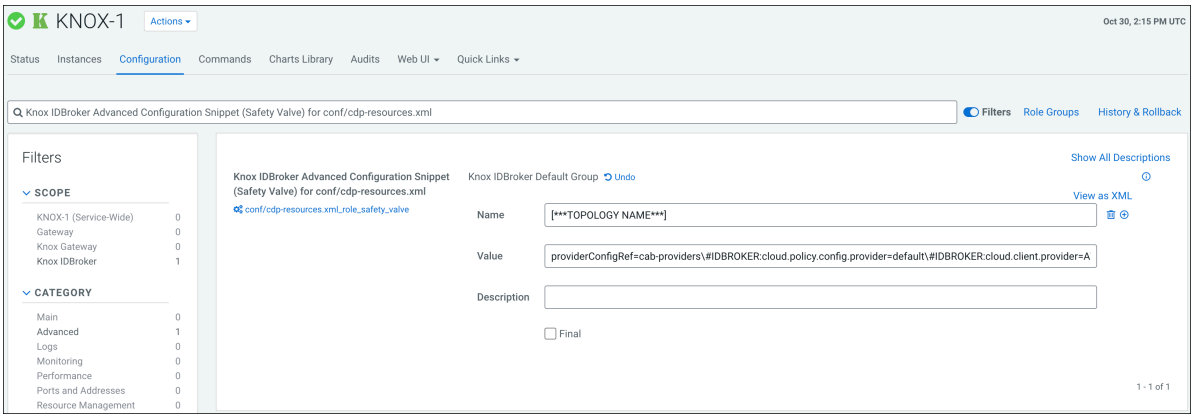
5. Click the Save Changes(CTRL+S) button.
6. Go to Actions Save Alias - IDBroker .
7. Find the Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml advanced configuration snippet.
8. Click the + icon, and add the following entries:
   - Name=*[***TOPOLOGY NAME***]*.

     Replace *[***TOPOLOGY NAME***]* with the specific name of your topology.
   - Value=

     ```
     providerConfigRef=cab-providers\#IDBROKER:cloud.policy.config.provider=default\#IDBROKER:cloud.client.provider=AWS\#IDBROKER:hashicorp.vault.enabled=true\#IDBROKER:hashicorp.vault.address=[***VAULT ADDRESS***]\#ID
     ```

```
BROKER:hashicorp.vault.path=aws/sts\#IDBROKER:hashicorp.vault.server.cer
t.path=[***VAULT CERTIFICATE PATH***]
```

Figure 6: Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml



The value consists of the following elements:

- hashicorp.vault.enabled=true – Enables the vault integration for that specific topology.
- hashicorp.vault.address=*[***VAULT ADDRESS***]* – Specifies the actual address of the vault.
- hashicorp.vault.path=aws/sts – Specifies the path for the STS credentials.
- (Optional) hashicorp.vault.server.cert.path=*[***VAULT CERTIFICATE PATH***]* – Adds the path for the vault certificate. This value is only required in case of self-signed certificates. Replace *[***VAULT CERTIFICATE PATH***]* with the path to the certificate.

9. Click the Save Changes(CTRL+S) button.
10. Search for the Knox IDBroker AWS User Mapping property.
11. Enter the following parameter:

```
hdfs=arn:aws:iam::[***ID***]:role/[***ROLE***]
```

Replace *[***ID***]* with your AWS account ID, and *[***ROLE***]* with the specific IAM role.

Figure 7: Knox IDBroker AWS User Mapping



12. Refresh the Knox instances configuration by clicking the Stale Configuration: Refresh needed indicator and wait until the refresh process completes.
13. Verify that the vault integration was successful by running the following command in the Command Line Interface (CLI):

```
kinit hdfs
hdfs dfs -ls s3a://[***RESOURCE***]
```

Replace *[***RESOURCE***]* with the name of your S3 bucket.

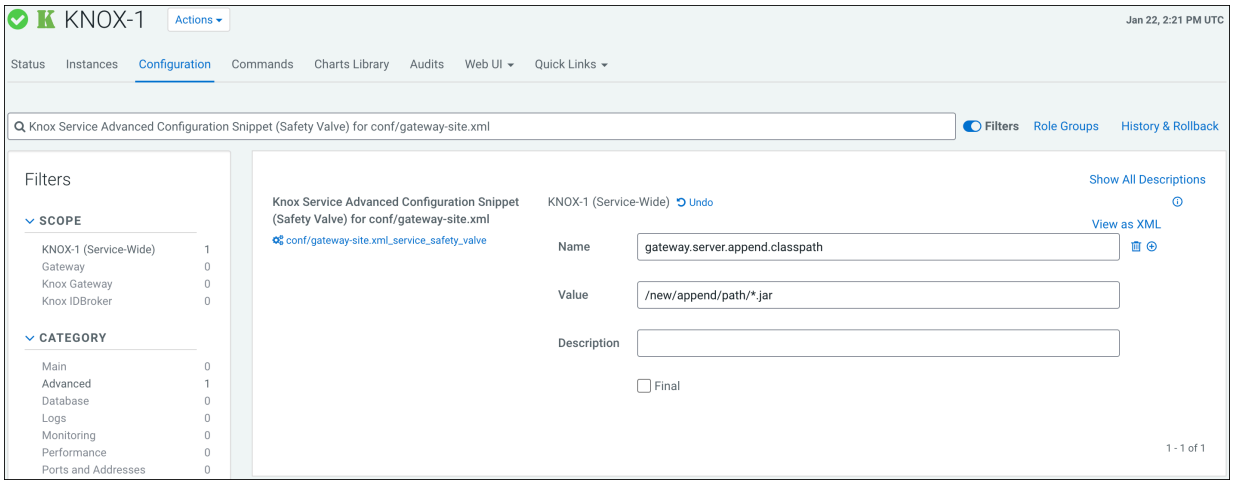
Extend the Knox Gateway classpath

Learn how to extend the Knox Gateway classpath by prepending or appending custom paths to include additional JAR files or dependencies outside the Knox Gateway home directory.

Procedure

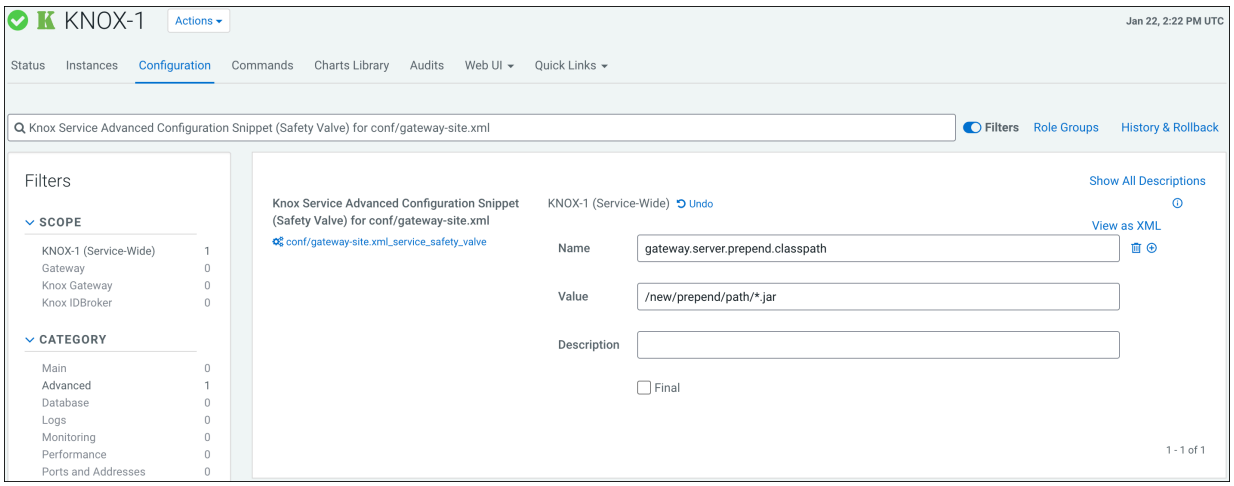
1. In Cloudera Manager, select the Knox service.
2. Go to the Configuration tab.
3. Search for Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml.
4. To append new path(s) to the classpath, click the + icon and add the following property:
   - Name: gateway.server.append.classpath
   - Value: The path to your custom libraries, for example /new/append/path/*.jar

   The gateway.server.append.classpath property adds the specified paths to the end of the Knox Gateway classpath, allowing Knox to load additional libraries after the default ones.



5. To prepend new path(s) to the classpath, click the + icon and add the following property:
   - Name: gateway.server.prepend.classpath
   - Value: The path to your custom libraries, for example /new/prepend/path/*.jar

   The gateway.server.prepend.classpath property adds the specified paths to the beginning of the Knox Gateway classpath, allowing your custom libraries to take precedence over the default Knox libraries.



6. Configure the path value according to your requirements using the following syntax rules:
   - To include multiple locations, separate them with a comma (,) or semicolon (;).
   - Use *.jar to include all JAR files in a folder.
   - Use * to include all files in a folder.
   - Use /folder to include class files from folder hierarchies. For example, to include GatewayServer.class, place it in org/apache/knox/gateway.
7. Click Save Changes(CTRL+S).
8. Restart the Knox service to apply the classpath changes.

   Go to Actions Restart and wait for the restart to complete.

Configure Knox for IPv4-only operation

In dual-stack environments using Cloudera Runtime 7.3.2.0, you must configure Knox to operate in IPv4-only mode to prevent unintended IPv6 communication.

About this task

By default, Knox uses IPv4. However, in dual-stack environments where both IPv4 and IPv6 are available, Knox may inadvertently use IPv6 for communication.

Procedure

1. In Cloudera Manager, select the Knox service.
2. Go to the Configuration tab.
3. Search for the Additional Gateway Java Options property.
4. Enter the following parameter:

   ```
   -Djava.net.preferIPv4Stack=true
   ```
5. Click the Save Changes(CTRL+S) button.
6. Restart the Knox service.

Results

Knox now operates in IPv4-only mode and uses IPv4 addresses for all network communication.

Management of Service Parameters for Apache Knox via Cloudera Manager

You can add, modify, or remove custom service parameters in Knox proxy via Cloudera Manager.

Add custom service parameter to descriptor

How to add a custom service parameter to a descriptor using Cloudera Manager.

Before you begin

The descriptor you wish to add a custom service parameter to must be enabled. See “Add a known service to cdp-proxy”.

About this task

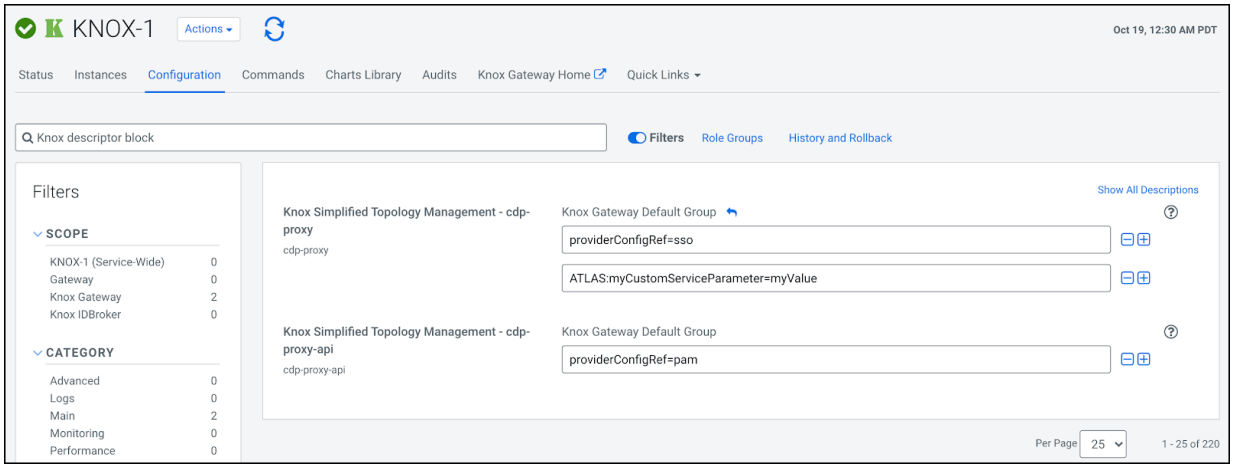
In this example, you are adding a custom service parameter with a custom value (myCustomServiceParameter=myValue) to ATLAS in cdp-proxy.

Procedure

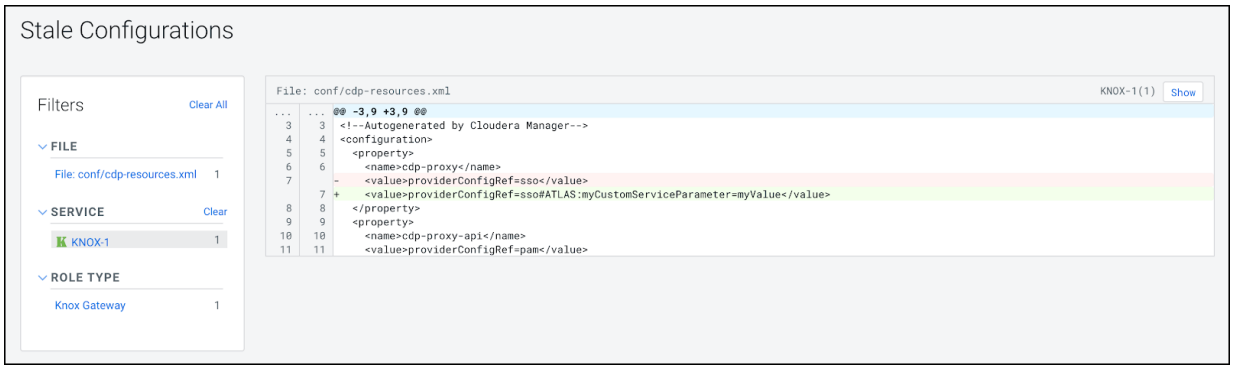
1. From Cloudera Manager Knox Configuration , add a new line in the Knox Simplified Topology Management - cdp-proxy panel in the following format:
   $SERVICE_NAME*[:*$PARAMETER_NAME=$PARAMETER_VALUE].
   ATLAS:myCustomServiceParameter=myValue

   The url and version parameter names are preserved keywords to set the given service's URL and version. Valid declarations:

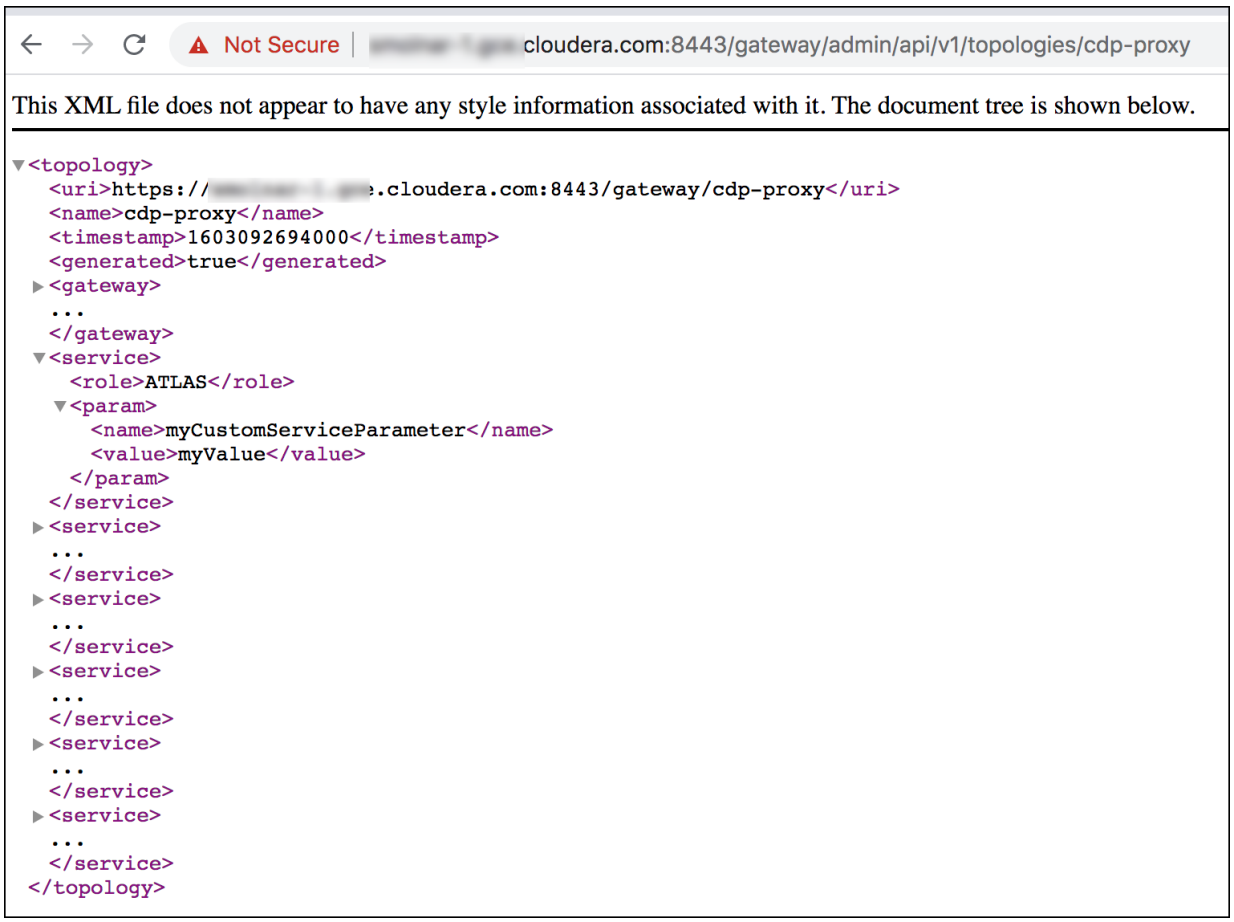
   ```
   ATLAS:url=http://localhost:123
   ATLAS:version:3.0.0
   ATLAS:test.parameter.name=test.parameter.value
   ```



2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process completes.



4. Validate that ATLAS in cdp-proxy got updated with the new service parameter by navigating to the following URL: https://*$KNOX_GATEWAY_HOST*:*$PORT*/*$GATEWAY_PATH*/admin/api/v1/topologies/cdp-proxy.



Modify custom service parameter in descriptor

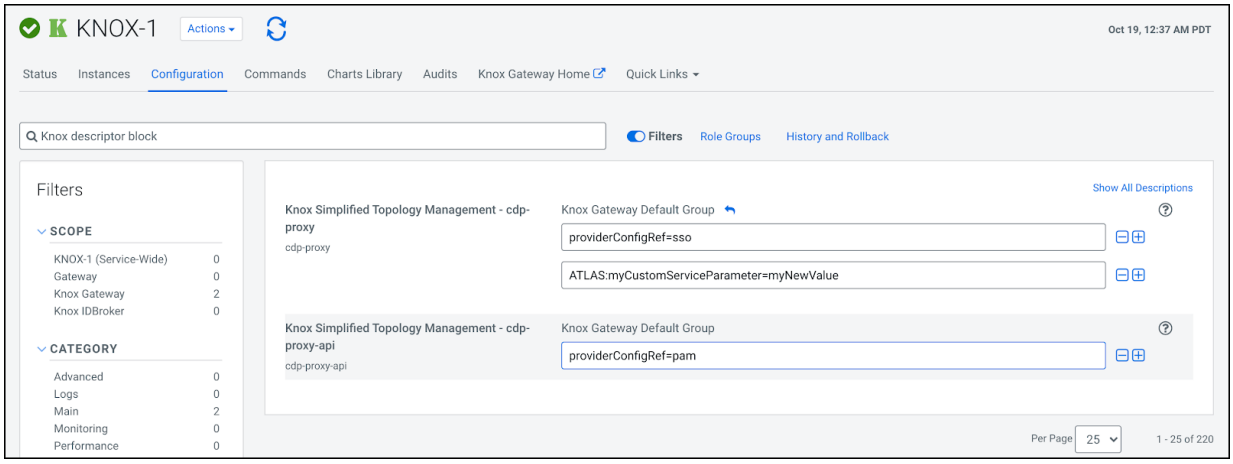
How to edit a custom service parameter in a Knox descriptor using Cloudera Manager.

About this task

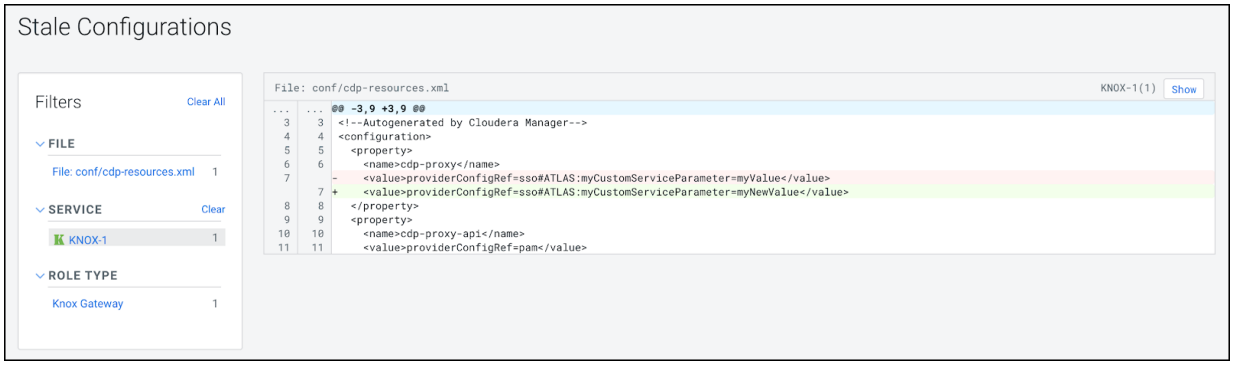
In this sample, we are going to update a previously entered service parameter - myCustomServiceParameter=my Value to myNewValue- for ATLAS in cdp-proxy. We change that entry, save our changes, and refresh our cluster.

Procedure

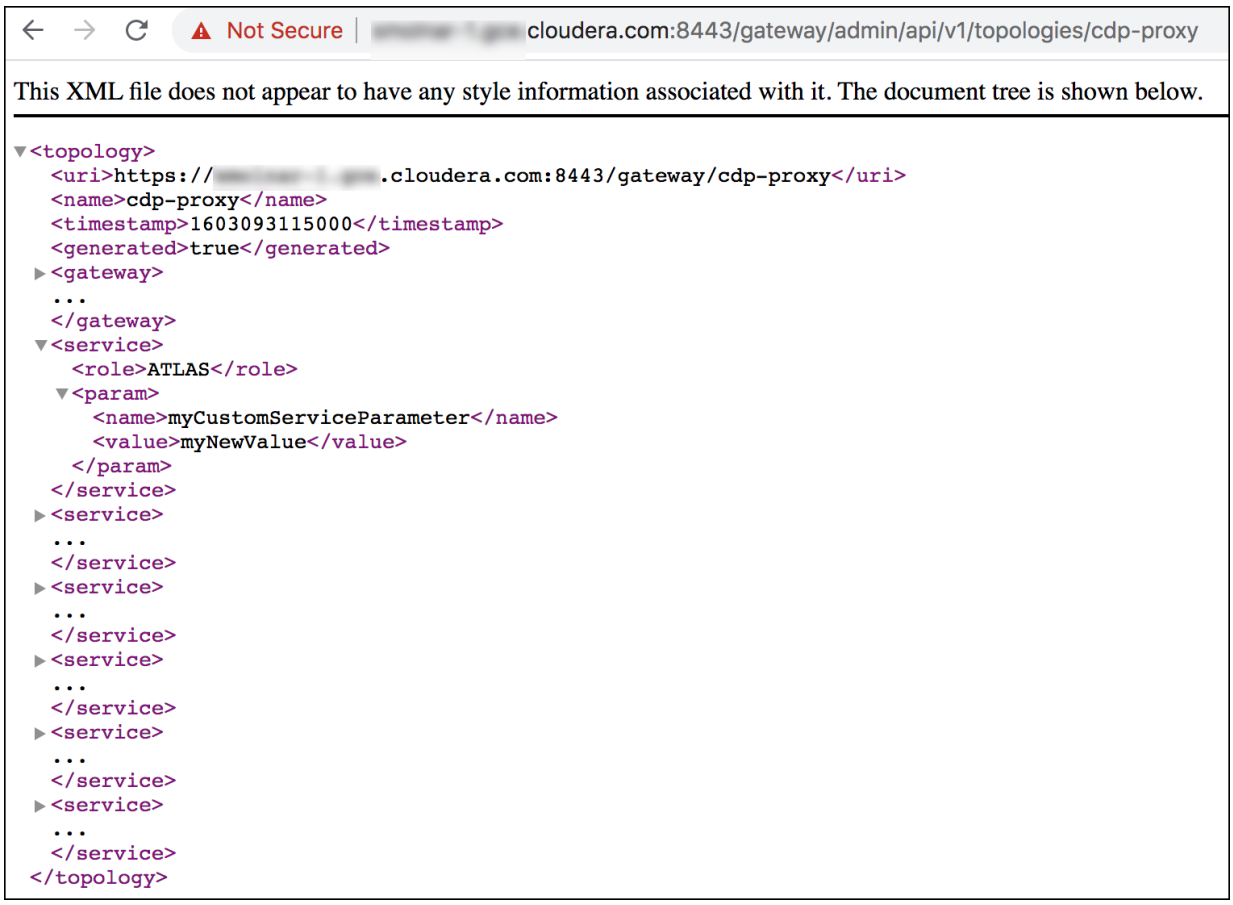
1. From Cloudera Manager Knox Configuration , change the service parameter in the Knox Simplified Topology Management - cdp-proxy panel.
   Change ATLAS:myCustomServiceParameter=myValue to Atlas:myCustomServiceParameter=myNewValue



2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process completes.



4. Validate that custom service parameter got updated with the changes by navigating to the following URL: http s://*$KNOX_GATEWAY_HOST*:*$PORT*/*$GATEWAY_PATH*/admin/api/v1/topologies/cdp-proxy.



Remove custom service parameter from descriptor

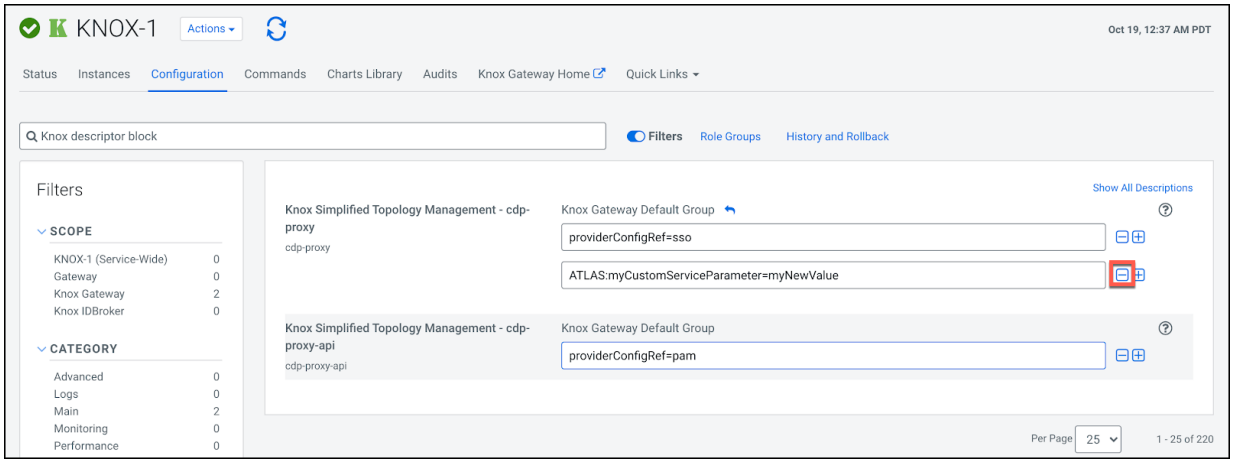
How to remove a custom service parameter from a descriptor using Cloudera Manager.

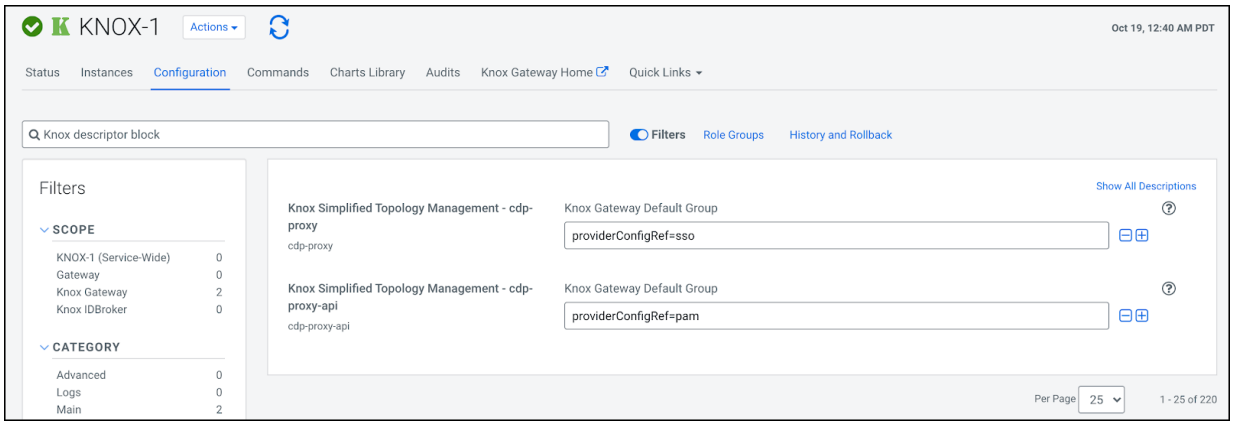
About this task

In this sample, we are going to remove a previously entered service parameter - myCustomServiceParameter=my NewValue - from ATLAS in cdp-proxy. We remove that entry, save our changes, and refresh our cluster.

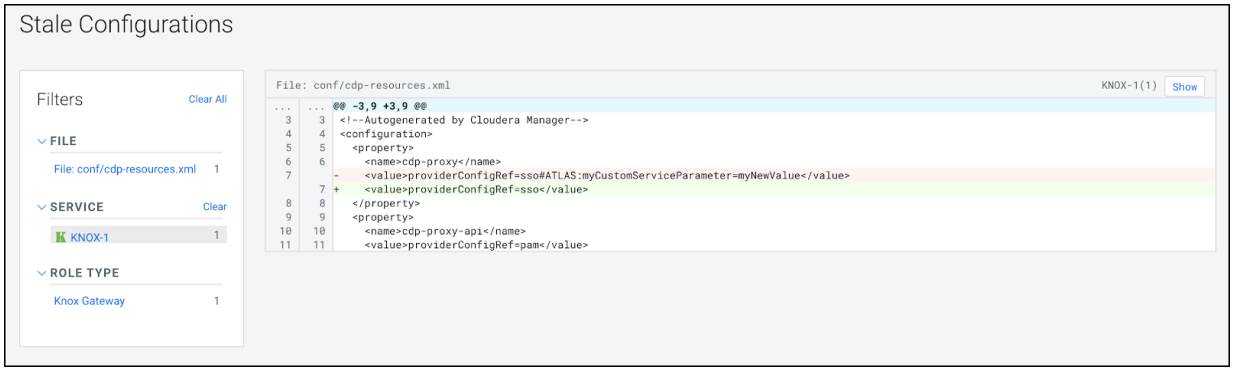
Procedure

1. From Cloudera Manager Knox Configuration , remove the ervice parameter in the Knox Simplified Topology Management - cdp-proxy panel.
   Click the minus (–) sign next to Atlas:myCustomServiceParameter=myNewValue.

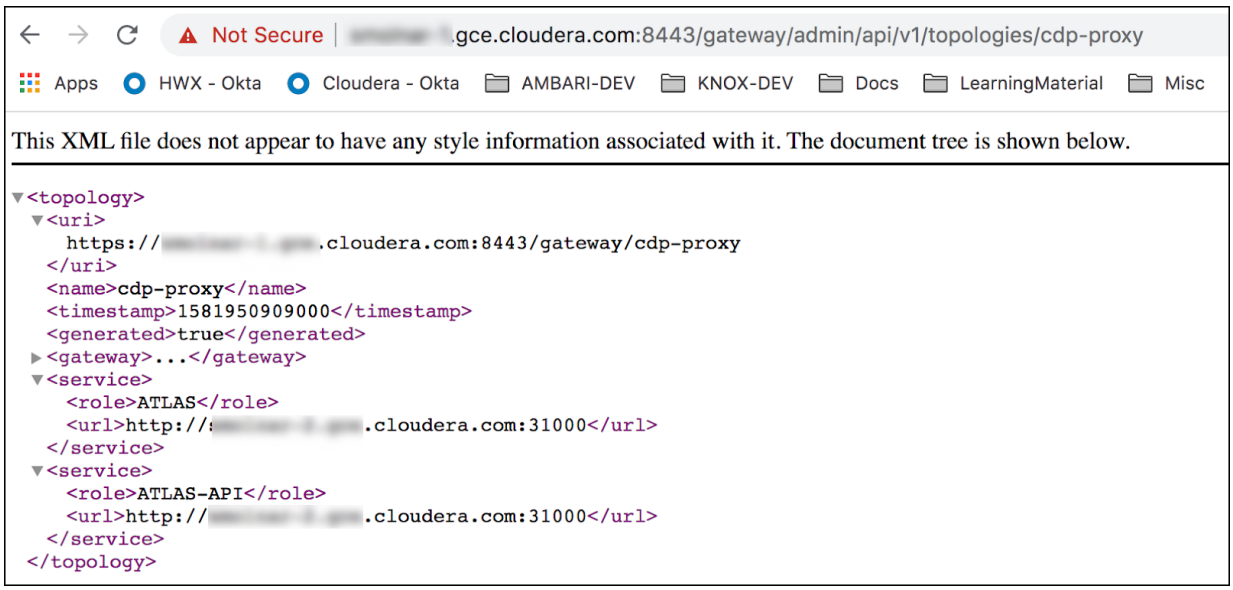




2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process completes.



4. Validate that custom service parameter got removed with the changes by navigating to the following URL: http s://*$KNOX_GATEWAY_HOST*:*$PORT*/*$GATEWAY_PATH*/admin/api/v1/topologies/cdp-proxy.



Load balancing for Apache Knox

Knox offers load balancing using a simple round robin algorithm which prevents load on one specific node.

- For services that are stateless, Knox loadbalances them using a simple round robin algorithm which prevents load on one specific node.
- For services that are stateful (i.e., require sessions, such as Ranger and Hive,) sessions are loadbalanced using a round robin algorithm, where each new session will use a different host and all the requests in the same session will be routed to the same host. This will continue until a session terminates or there is a failover.
- In case of failover, services that are stateful will return error response 502.

This behavior is configurable and can be changed by tuning various flags in Knox HA provider for the respective services.

Load balancing vs high availability (HA)

Currently, Knox offers load balancing using a simple round robin algorithm which prevents load on one specific node.

Because we do not support session persistence, this is not true HA, as there could be a case where stateful service will not failover to other node.

Supported services

The following services support Knox load balancing:

- Hive
- Impala
- Oozie
- Phoenix
- Ranger
- Solr

Default enabled values

The following default values are enabled in the Knox topology. API is located in cdp-proxy-api.xml; UI is located in cdp-proxy.xml.

- Hive
  - API: enableStickySession=true;noFallback=true;enableLoadBalancing=true
- Phoenix
  - API: enableStickySession=true;noFallback=true;enableLoadBalancing=true
- Ranger
  - API: enableStickySession=false;noFallback=false;enableLoadBalancing=true
  - UI: enableStickySession=true;noFallback=true;enableLoadBalancing=true
- Solr
  - API: enableStickySession=false;noFallback=false;enableLoadBalancing=true
  - UI: enableStickySession=true;noFallback=true;enableLoadBalancing=true

Configure load balancing for Apache Hive through Knox

Learn how to configure load balancing for Apache Hive when accessing it through Knox Gateway using the Cloudera ODBC driver for Hive to prevent connectivity errors and distribute workload across Hive instances.

About this task

Load balancing is disabled by default when accessing Apache Hive using the Cloudera ODBC driver for Hive. To enable load balancing, you must configure the disableLoadBalancingForUserAgents parameter in Cloudera Manager.

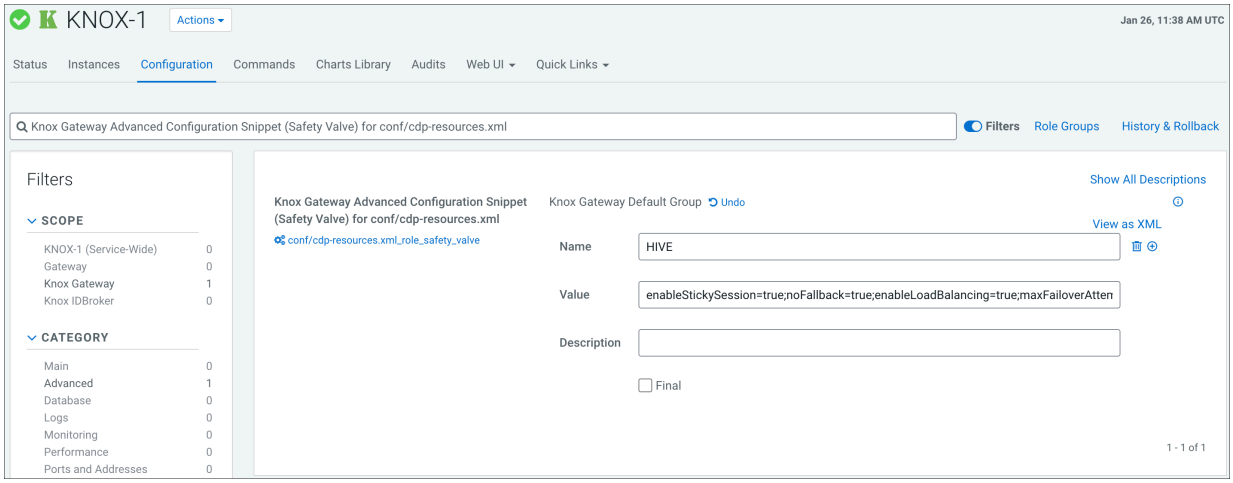


Note: Load balancing for Apache Knox is supported with Cloudera ODBC driver for Hive 2.6.15 and later versions.

Procedure

1. In Cloudera Manager, select the Knox service.
2. Go to the Configuration tab.
3. Search for Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml.
4. Click the + icon and add the following parameter:
   - Name: HIVE
   - Value:

```
enableStickySession=true;noFallback=true;enableLoadBalancing=true;maxFai
loverAttempts=0;disableLoadBalancingForUserAgents=NONE
```



5. Click Save Changes(CTRL+S).
6. Refresh the Knox instances configuration by clicking the Stale Configuration: Refresh needed indicator and wait until the refresh process completes.

Results

This configuration enables load balancing for Hive, ensuring proper workload distribution across Hive instances, and preventing connectivity errors when using the Cloudera ODBC driver for Hive.

Generate and configure a signing keystore for Knox in HA

When Knox is installed on more than one instance (i.e., when Knox is running in HA), then signing keystore configurations must be set in Cloudera Manager.

Procedure

1. Generate your own certificate and keystore file, and then copy to /var/lib/knox/gateway/data/security/keystores/.

   For more details, see Manually Configuring TLS Encryption for Cloudera Manager.
2. Set the following values:
   - gateway_signing_keystore_name: the filename of keystore file that contains the signing keypair.
   - gateway_signing_keystore_type: the type of the keystore file where the signing keypair is stored. In non-FIPS environments, this should be PKCS12.
   - gateway_signing_key_alias: the alias for the signing keypair within the keystore.
3. If you do not want the master secret to be used, you can set an alias for the password to the keystore file that holds the signing keypair.
   a) Go to Saving Aliases and follow the instructions.
   b) From Cloudera Manager Knox Configuration Knox Service (or Gateway) Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml_service_safety_valve: , configure gateway.signing.keystore.password.alias to the alias previously defined.

Knox Gateway token integration

You can use the Apache Knox homepage to generate and manage Knox Gateway tokens for Cloudera.

Related Information

Knox token management (in v1.6.0 and above)

Overview

Instead of using a basic username/password pair, you can improve security by generating Knox Gateway tokens. Tokens are more secure than plaintext username/password because they are signed, anonymized from the source data, and have a specified lifetime (by default, one hour).

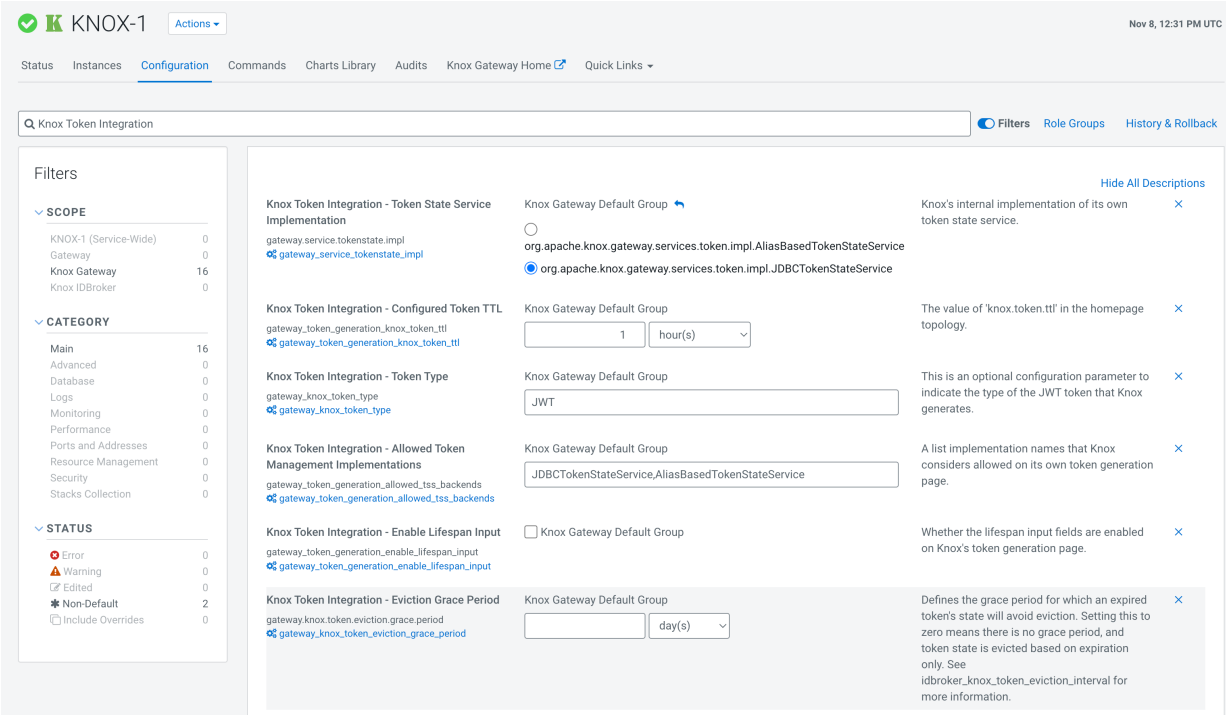
About Knox gateway tokens

Before Cloudera Data Platform 7.1.9, Knox on Cloudera on premises had two default topologies: cdp-proxy and cdp-proxy-api. To enable passcode tokens, a third Knox topology was added: cdp-proxy-token. While very similar to cdp-proxy-api, the authentication provider for cdp-proxy-token is configured with the JWTFederation provider, so that newly generated tokens can be used.

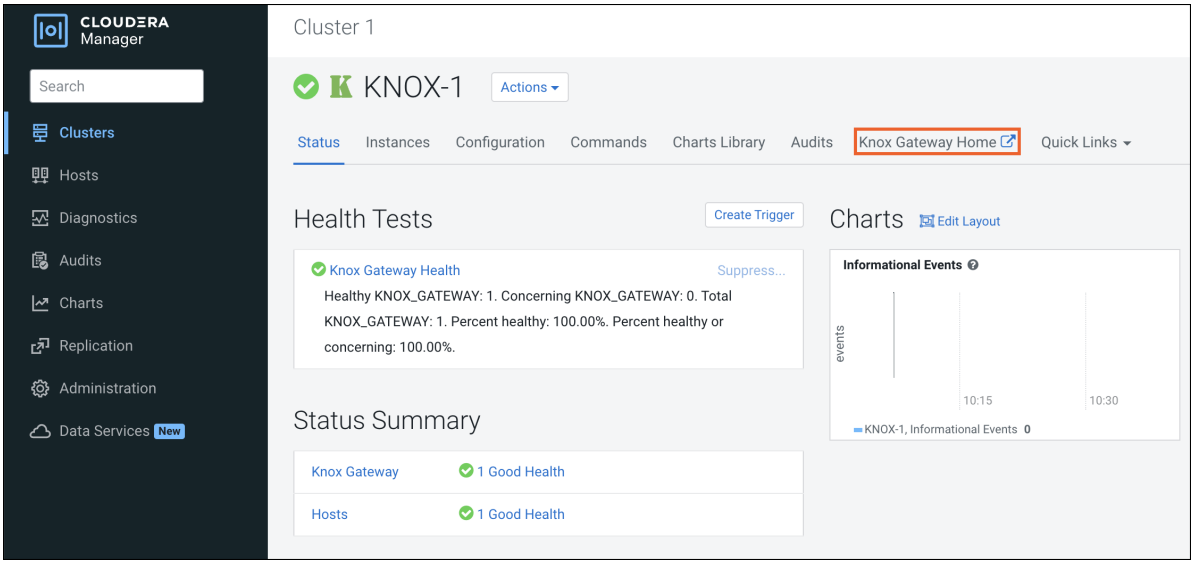
View Knox token integration

Knox token integration can be accessed via Cloudera Manager or the Knox homepage:

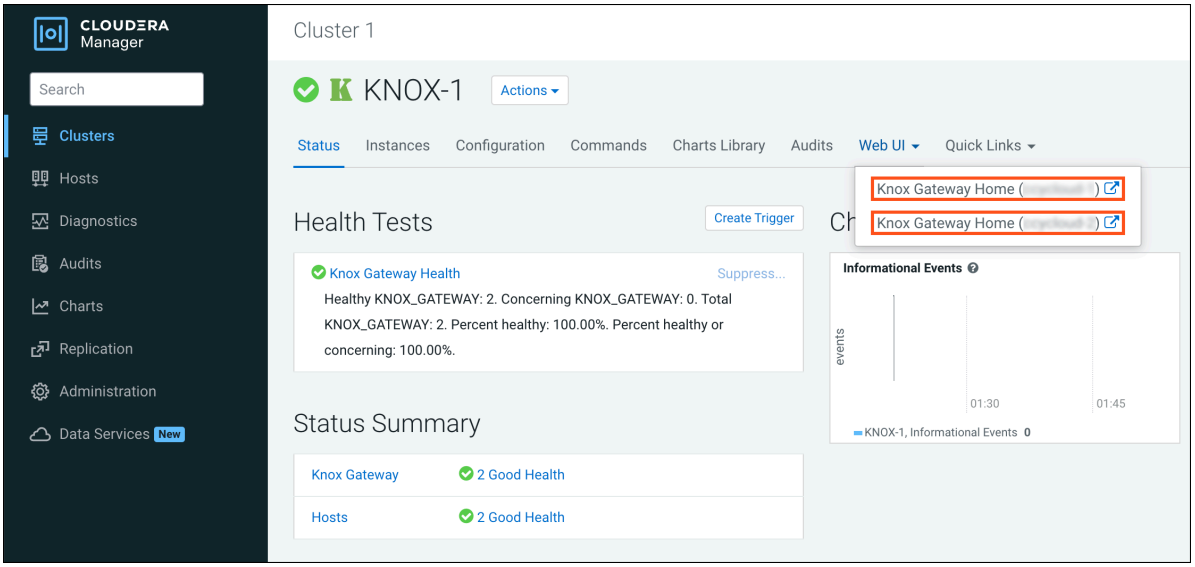
- (Recommended) Cloudera Manager: Cloudera Manager Clusters Knox Configuration and search for “Knox Token Integration”.



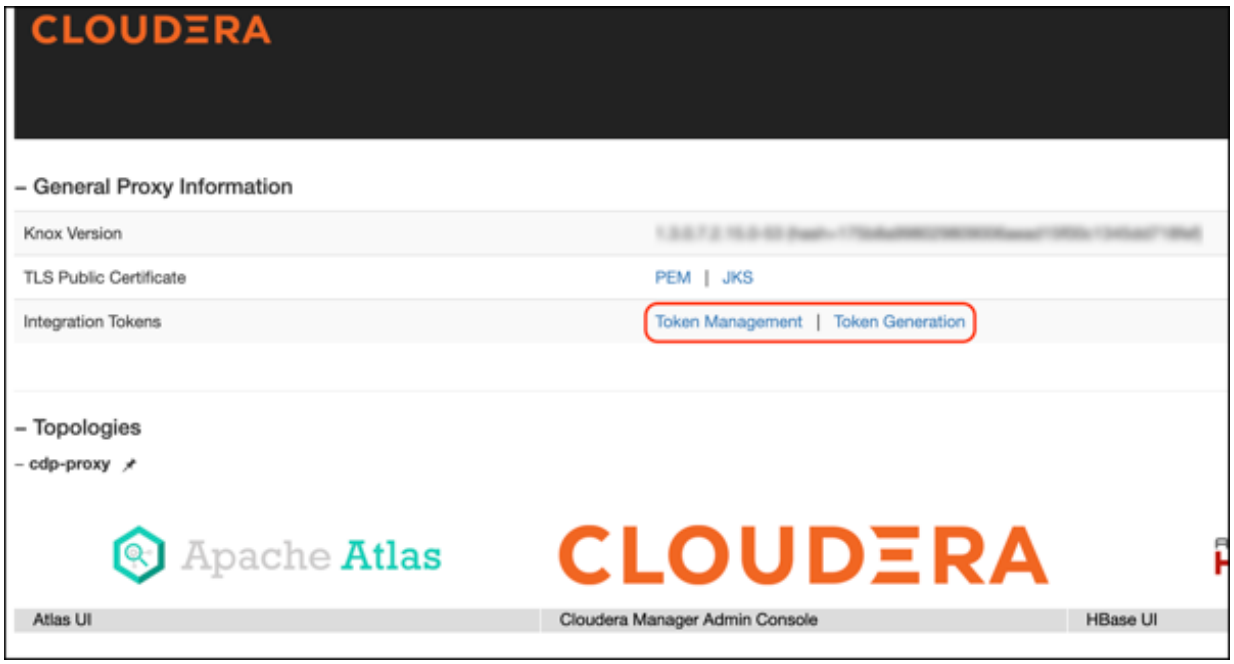
- Navigate to Cloudera Manager Knox . Perform one of the following actions, depending on whether there is one Knox Gateway instance or more than one on the cluster:
  - If only one Knox Gateway instance is installed on the cluster, select Knox Gateway Home.



- If Knox Gateway high availability (HA) is enabled and more than one Knox instance is installed on the cluster, click Web UI, and select any of the Knox Gateway Home links from the drop-down menu.



This will bring you to the Knox homepage. There are two new links on your Knox homepage: Token Management and Token Generation.



Knox token integration in Cloudera works out of the box using the Knox Token Generation page. However, the token integration API can be re-used in your own custom topology.

Attention: The only restriction of the above approach is that your custom topology must not use the HadoopAuth authentication provider because it won't work with the KNOXTOKEN service due to a known issue (which will be fixed in future releases).

Token configurations

The default configurations for Knox token integration are as follows.

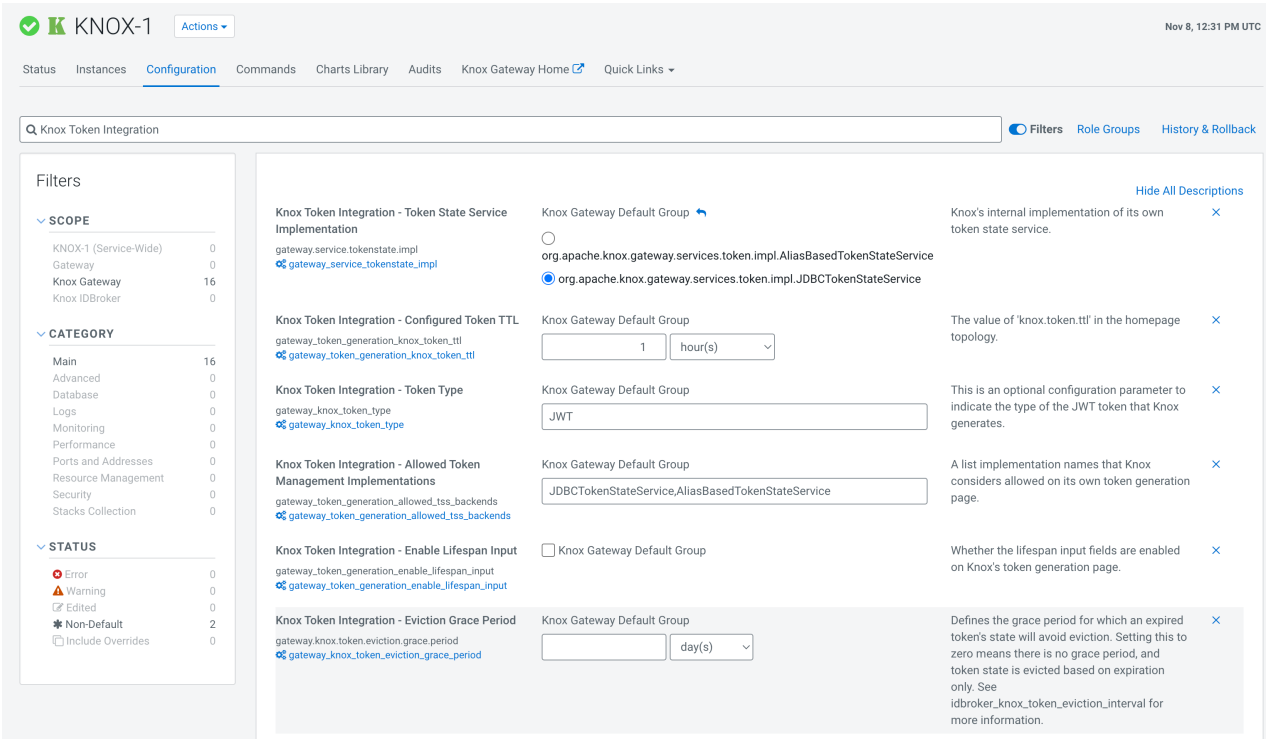
Default configurations

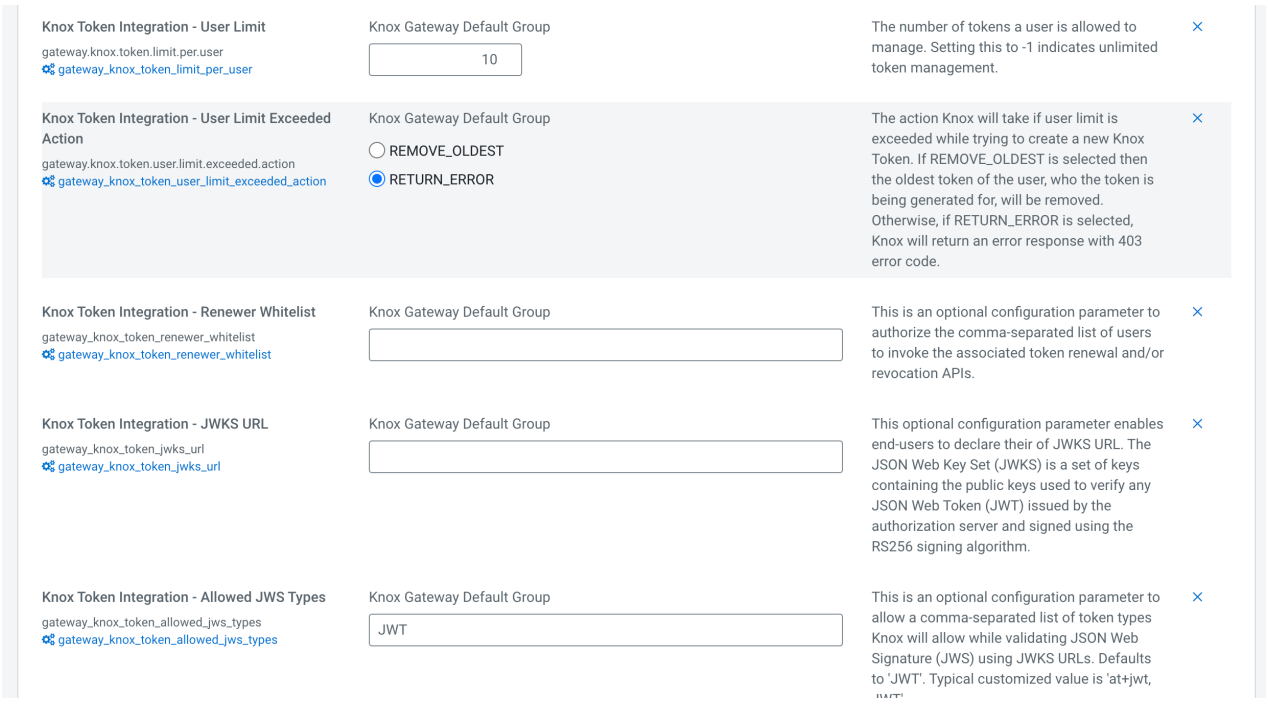
Table 7: Default token configurations

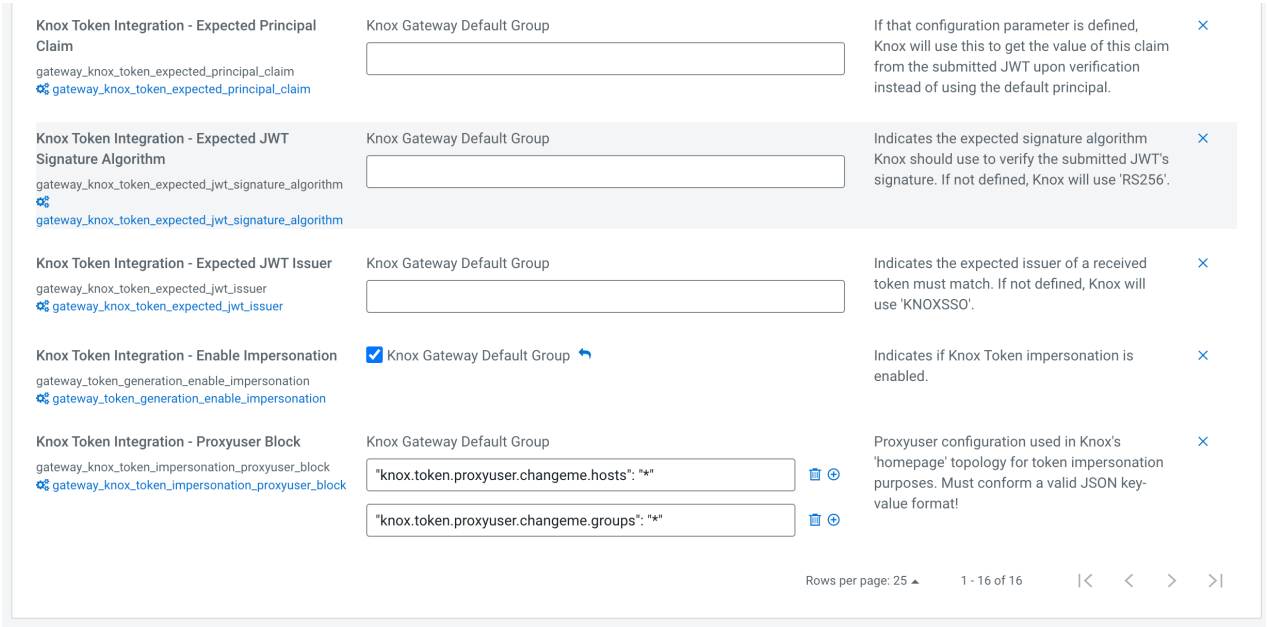
Property	Sample values	Default
Token State Service Implementation	Knox's internal implementation of its own token state service.	org.apache.knox.gateway.services.token.impl.JDBCTokenStateService
Allowed Token Management Implementations	A list of implementation names that Knox considers allowed on its own token generation page.	JDBCTokenStateService,AliasBasedTokenStateService
Configured Token TTL	The value of “knox.token.ttl” in the homepage topology.	1 hour
Token Type	This is an optional configuration parameter to indicate the type of the JWT token that Knox generates.	JWT
Enable Lifespan Input	Whether the lifespan input fields are enabled on Knox's token generation page.	false
User Limit	The number of tokens a user is allowed to manage. Setting this to -1 indicates unlimited token management.	10

Property	Sample values	Default
User Limit Exceeded Action	The action Knox will take if user limit is exceeded while trying to create a new Knox Token. If REMOVE_OLDEST is selected then the oldest token of the user, who the token is being generated for, will be removed. Otherwise, if RETURN_ERROR is selected, Knox will return an error response with 403 error code.	RETURN_ERROR
Renewer Whitelist	This is an optional configuration parameter to authorize the comma-separated list of users to invoke the associated token renewal and/or revocation APIs.	empty string
JWKS URL	This optional configuration parameter enables end-users to declare their JWKS URL. The JSON Web Key Set (JWKS) is a set of keys containing the public keys used to verify any JSON Web Token (JWT) issued by the authorization server and signed using the RS256 signing algorithm.	empty string
Allowed JWS Types	This is an optional configuration parameter to allow a comma-separated list of token types Knox will allow while validating JSON Web Signature (JWS) using JWKS URLs. Defaults to “JWT”. The typical customized value is “at +jwt, JWT”.	JWT
Expected Principal Claim	If that configuration parameter is defined, Knox will use this to get the value of this claim from the submitted JWT upon verification instead of using the default principal.	empty string
Expected JWT Signature Algorithm	Indicates the expected signature algorithm Knox should use to verify the submitted JWT's signature. If not defined, Knox will use 'RS256'.	empty string
Expected JWT Issuer	Indicates the expected issuer of a received token must match. If not defined, Knox will use 'KNOXSSO'.	empty string
Enable Impersonation	Indicates if Knox Token impersonation is enabled.	false
Proxyuser Block	If Knox token impersonation is enabled, it allows the specified user(s) to impersonate members of certain groups from specific hosts. For example, the following values allow a user named "changeme" to impersonate members of any group from any host: • "knox.token.proxyuser.changeme.hosts": "*" • "knox.token.proxyuser.changeme.groups": "*" The values for this property must be in JSON key/value format.	empty string

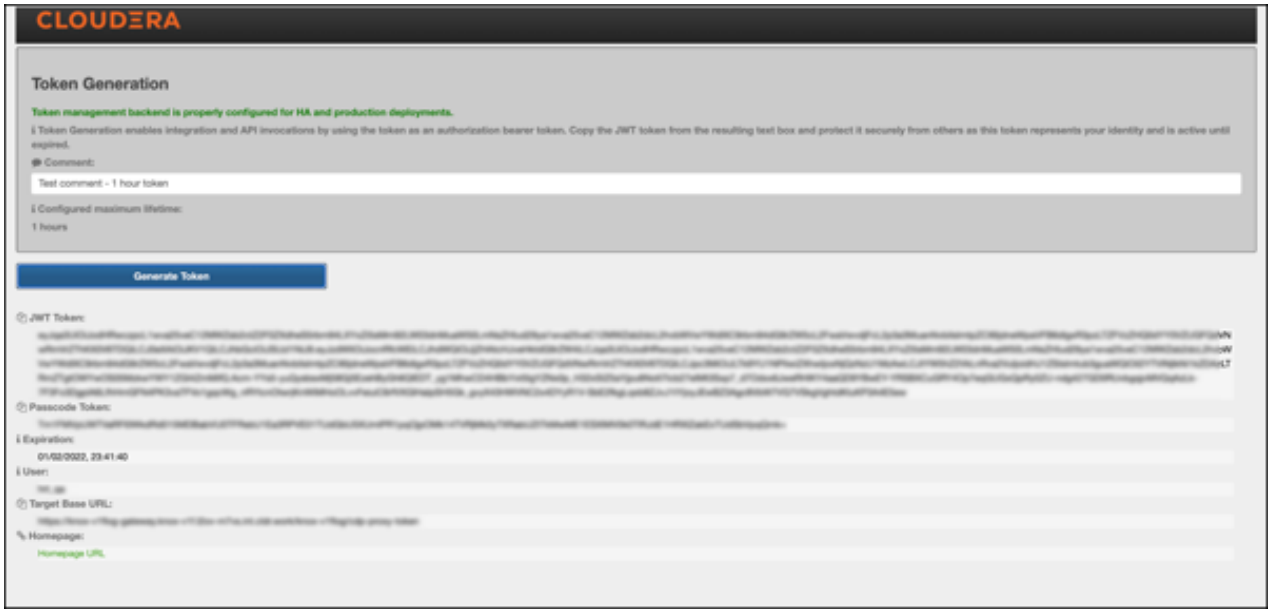
Default configurations seen from Cloudera Manager:


KNOX-1
Actions
Nov 8, 12:31 PM UTC
Status
Instances
Configuration
Commands
Charts Library
Audits
Knox Gateway Home
Quick Links
Knox Token Integration
Filters
Role Groups
History & Rollback
Filters
SCOPE
KNOX-1 (Service-Wide) 0
Gateway 0
Knox Gateway 16
Knox IDBroker 0
CATEGORY
Main 16
Advanced 0
Database 0
Logs 0
Monitoring 0
Performance 0
Ports and Addresses 0
Resource Management 0
Security 0
Stacks Collection 0
STATUS
Error 0
Warning 0
Edited 0
Non-Default 2
Include Overrides 0
Hide All Descriptions
Knox Token Integration - Token State Service Implementation
gateway.service.tokenstate.impl
gateway_service_tokenstate_impl
Knox Gateway Default Group
org.apache.knox.gateway.services.token.impl.AliasBasedTokenStateService
org.apache.knox.gateway.services.token.impl.JDBCTokenStateService
Knox's internal implementation of its own token state service.
Knox Token Integration - Configured Token TTL
gateway_token_generation_knox_token_ttl
gateway_token_generation_knox_token_ttl
Knox Gateway Default Group
1
hour(s)
The value of 'knox.token.ttl' in the homepage topology.
Knox Token Integration - Token Type
gateway_knox_token_type
gateway_knox_token_type
Knox Gateway Default Group
JWT
This is an optional configuration parameter to indicate the type of the JWT token that Knox generates.
Knox Token Integration - Allowed Token Management Implementations
gateway_token_generation_allowed_tss_backends
gateway_token_generation_allowed_tss_backends
Knox Gateway Default Group
JDBCTokenStateService,AliasBasedTokenStateService
A list implementation names that Knox considers allowed on its own token generation page.
Knox Token Integration - Enable Lifespan Input
gateway_token_generation_enable_lifespan_input
gateway_token_generation_enable_lifespan_input
Knox Gateway Default Group
Whether the lifespan input fields are enabled on Knox's token generation page.
Knox Token Integration - Eviction Grace Period
gateway.knox.token.eviction.grace.period
gateway_knox_token_eviction_grace_period
Knox Gateway Default Group
day(s)
Defines the grace period for which an expired token's state will avoid eviction. Setting this to zero means there is no grace period, and token state is evicted based on expiration only. See idbroker_knox_token_eviction_interval for more information.


Knox Token Integration - User Limit
gateway.knox.token.limit.per.user
gateway_knox_token_limit_per_user
Knox Gateway Default Group
10
The number of tokens a user is allowed to manage. Setting this to -1 indicates unlimited token management.
Knox Token Integration - User Limit Exceeded Action
gateway.knox.token.user.limit.exceeded.action
gateway_knox_token_user_limit_exceeded_action
Knox Gateway Default Group
REMOVE_OLDEST
RETURN_ERROR
The action Knox will take if user limit is exceeded while trying to create a new Knox Token. If REMOVE_OLDEST is selected then the oldest token of the user, who the token is being generated for, will be removed. Otherwise, if RETURN_ERROR is selected, Knox will return an error response with 403 error code.
Knox Token Integration - Renewer Whitelist
gateway_knox_token_renewer_whitelist
gateway_knox_token_renewer_whitelist
Knox Gateway Default Group
This is an optional configuration parameter to authorize the comma-separated list of users to invoke the associated token renewal and/or revocation APIs.
Knox Token Integration - JWKS URL
gateway_knox_token_jwks_url
gateway_knox_token_jwks_url
Knox Gateway Default Group
This optional configuration parameter enables end-users to declare their of JWKS URL. The JSON Web Key Set (JWKS) is a set of keys containing the public keys used to verify any JSON Web Token (JWT) issued by the authorization server and signed using the RS256 signing algorithm.
Knox Token Integration - Allowed JWS Types
gateway_knox_token_allowed_jws_types
gateway_knox_token_allowed_jws_types
Knox Gateway Default Group
JWT
This is an optional configuration parameter to allow a comma-separated list of token types Knox will allow while validating JSON Web Signature (JWS) using JWKS URLs. Defaults to 'JWT'. Typical customized value is 'at+jwt,



Default configurations seen from the Knox homepage UI:



Database connection properties

Optional database connection properties that you can declare individually:

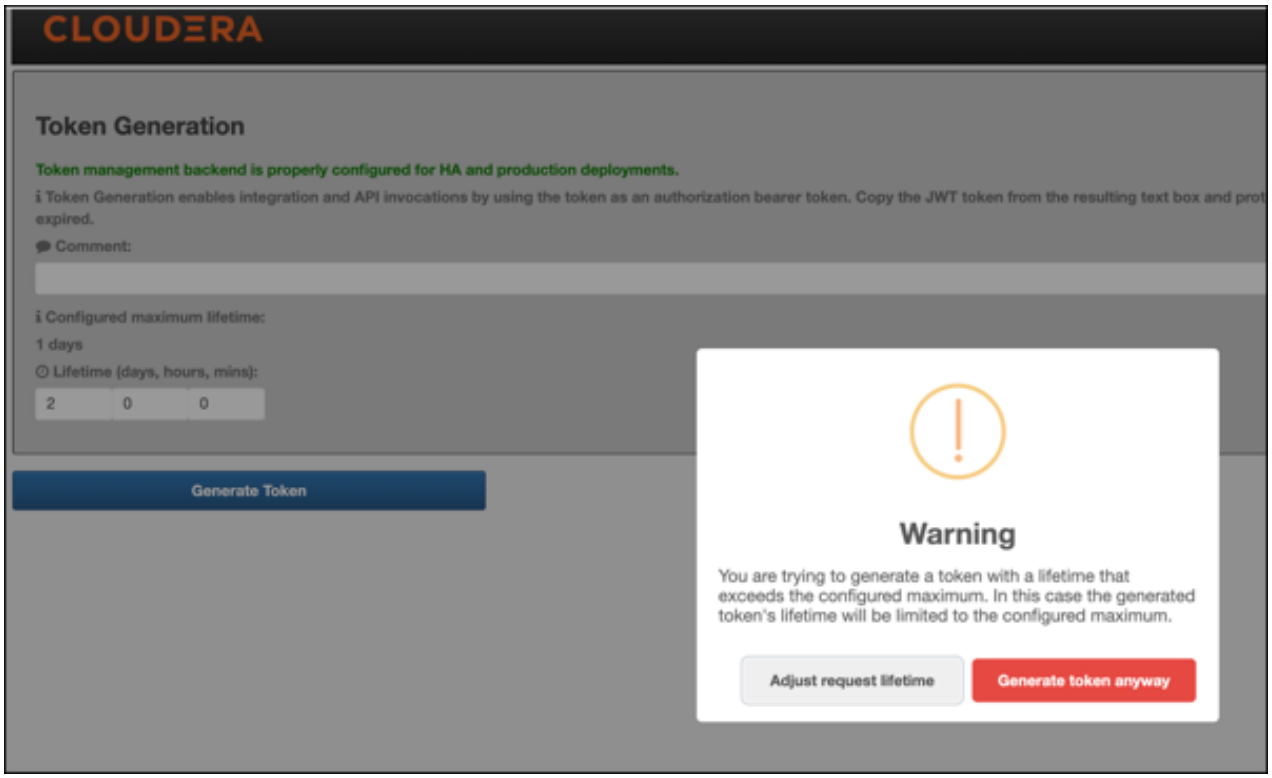
- gateway.database.type: Set to postgresql or mysql.
- gateway.database.host: Host where your DB server is running.
- gateway.database.port: Port that your DB server is listening on.
- gateway.database.name: Name of the database you are connecting to.

Important: From Cloudera Private Cloud Base 7.1.9 SP1 CHF1 release onwards, if you are using a MySQL database, then the JDBC driver must be installed manually to /usr/share/java. For more information, see Installing the MySQL JDBC Driver.

Token TTL details

Out of the box, Knox will display the custom lifetime spinners on the Token Generation page. However, they can be hidden by disabling the Knox Token Integration - Enable Lifespan Input checkbox on the CM UI. Given that input property, and the configured maximum lifetime property, the generated token can have the following TTL value:

- If there is no configured token TTL and lifespan inputs are disabled, the default TTL is used (30 seconds).
- If there is configured TTL and lifespan inputs are disabled, the configured TTL is used.
- If there is configured TTL and lifespan inputs are enabled and lifespan inputs result in a value that is less than or equal to the configured TTL, the lifespan query param is used.
- If there is configured TTL and lifespan inputs are enabled and lifespan inputs result in a value that is greater than the configured TTL, the configured TTL is used.



Generate-jwk options

CM automatically creates a token hash key for you. But if you want to do this manually, such as when scripting, configure the knox.token.hash.key alias with:

```
generate-jwk --saveAlias knox.token.hash.key
```

This generates a JSON Web Key using the supplied algorithm name.

Table 8: Options

Option	Description	Sample values
jwkAlg	(Optional) The desired JSON Web Signature algorithm name. Determines if the gateway-level alias is configured with a 256, 384, or 512-bit length JWK.	HS256 (Default) HS384 HS512
saveAlias	(Optional, Recommended) Given alias name used to save the generated JWK, instead of printing this sensitive information on the screen.	knox.token.hash.key

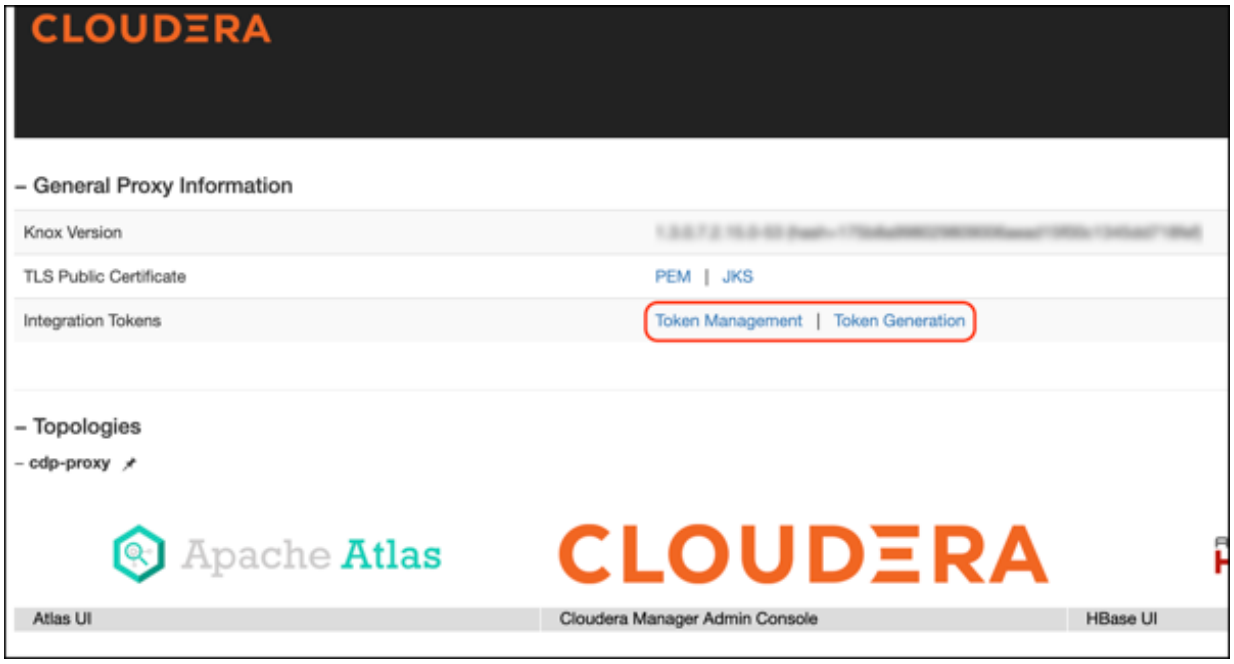
Option	Description	Sample values
topology	(Optional) Name of the topology (i.e., the cluster) to be used when saving the JWK as an alias. If none specified, the alias is going to be saved for the Gateway.	cdp-proxy (Default) cdp-proxy api

Generate tokens

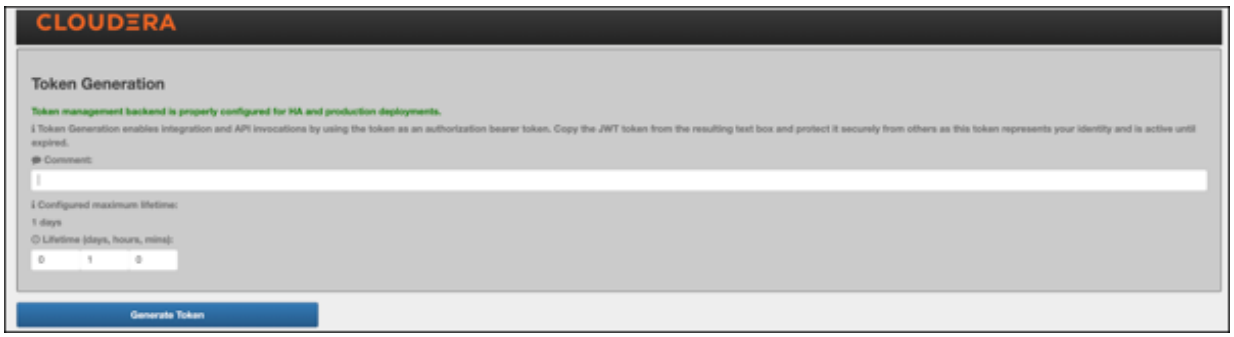
How to generate Knox gateway tokens from the Knox homepage.

Procedure

1. To access Knox generation management, go to https://*KNOX_GATEWAY_HOST*:*PORT*/*GATEWAY_PATH*/homepage/home, e.g. https://localhost:8443/gateway/homepage/home. Click on Token Generation.



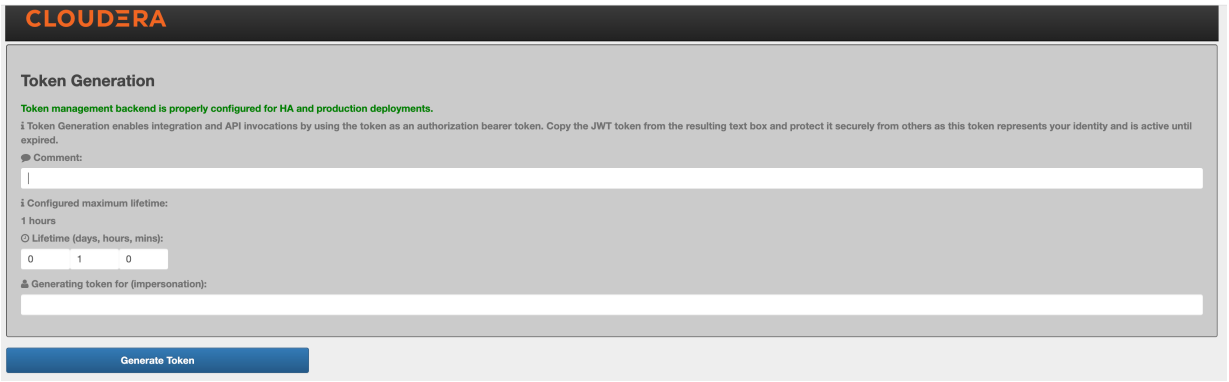
2. The following sections are displayed on the page:
   - Status bar: Message about the configured token state backend. There are 3 different statuses:
     - ERROR: Displayed in red. Indicates a problem with the service backend which makes the feature not work. Usually, this is visible when end-users configure JDBC token state service, but they make a mistake in their DB settings.
     - WARN: Displayed in yellow. Indicates that the feature is enabled and working, but there are some limitations.
     - INFO: Displayed in green. Indicates when the token management backend is properly configured for HA and production deployments.
   - Information label: Explains the purpose of the **Token Generation** page.
   - Comment: Optional input field that allows end-users to add meaningful comments (mnemonics) to their generated tokens. The maximum length is 255 characters.
   - Configured maximum lifetime: Informs the clients about the knox.token.ttl property set in the homepage topology (defaults to 1 day(s)). If that property is not set (e.g. someone removes it from he homepage topology), Knox uses a hard-coded value of 30 seconds (aka. default Knox token TTL).
   - Custom maximum (token) lifetime: Can be set by adjusting the days/hours/minutes fields. The default configuration will yield one hour.



If Knox Token Integration - Enable Impersonation is set to true, another input field is shown on the UI called Generating token for (impersonation).

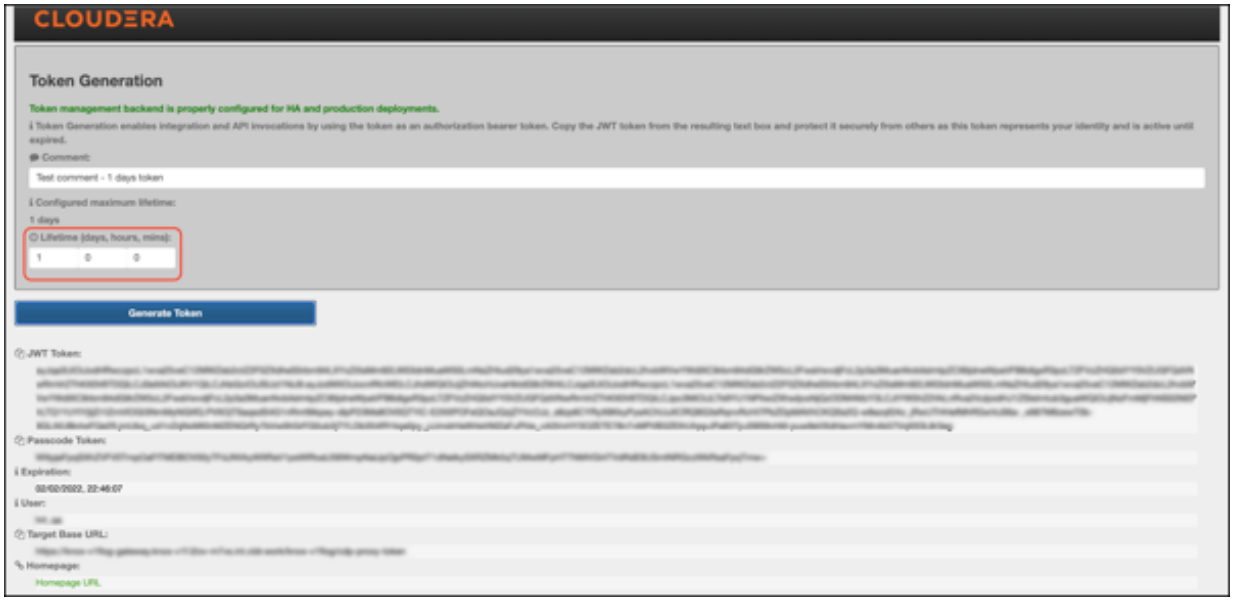
Using that input field our customers should be able to generate tokens on behalf of other users. For this to work, the Knox Token Integration - Proxyuser Block property has to be configured properly.

Important: If Knox is behind a Load Balancer and Token Impersonation support is used while generating tokens (that input field is populated with a username), the Load Balancer host must be added to the Proxy User configuration too. If the user wants to decline requests from a specific host, then that can be configured on the Load Balancer side.



For more information, see Knox Apache User-guide: Token impersonation

3. Click Generate Token.



4. Use the token to authenticate your request. Click the icon beside your choice on the page to copy the value to the clipboard:
   - JWT token: serialized JWT, fully compatible with the old-style bearer authorization method. You can use it as the ‘Token’ user:

```
$ curl -ku Token:eyJqa3U[...]uT5AxQGyMMP3VLGw https:/localhost:8443/gate
way/cdp-proxy-token/webhdfs/v1?op=LISTSTATUS

{"FileStatuses":{"FileStatus":[{"accessTime":0,"blockSize":0,"childre
nNum":1,"fileId":16386,"group":"supergroup",
"length":0,"modificationTime":1621238405734,"owner":"hdfs","pathSuffix"
:"tmp","permission":"1777","replication":0,
"storagePolicy":0,"type":"DIRECTORY"},{"accessTime":0,"blockSize":0,"chi
ldrenNum":1,"fileId":16387,"group":"supergroup",
"length":0,"modificationTime":1621238326078,"owner":"hdfs","pathSuffix"
:"user","permission":"755","replication":0,
"storagePolicy":0,"type":"DIRECTORY"}]}}
```

   - Passcode token: Serialized passcode token, which can be used as the ‘Passcode’ user:

```
$ curl -ku Passcode:WkRFMk1XTmh[...]RVNFpXRTA= https://localhost:8443/ga
teway/cdp-proxy-token/webhdfs/v1?op=LISTSTATUS

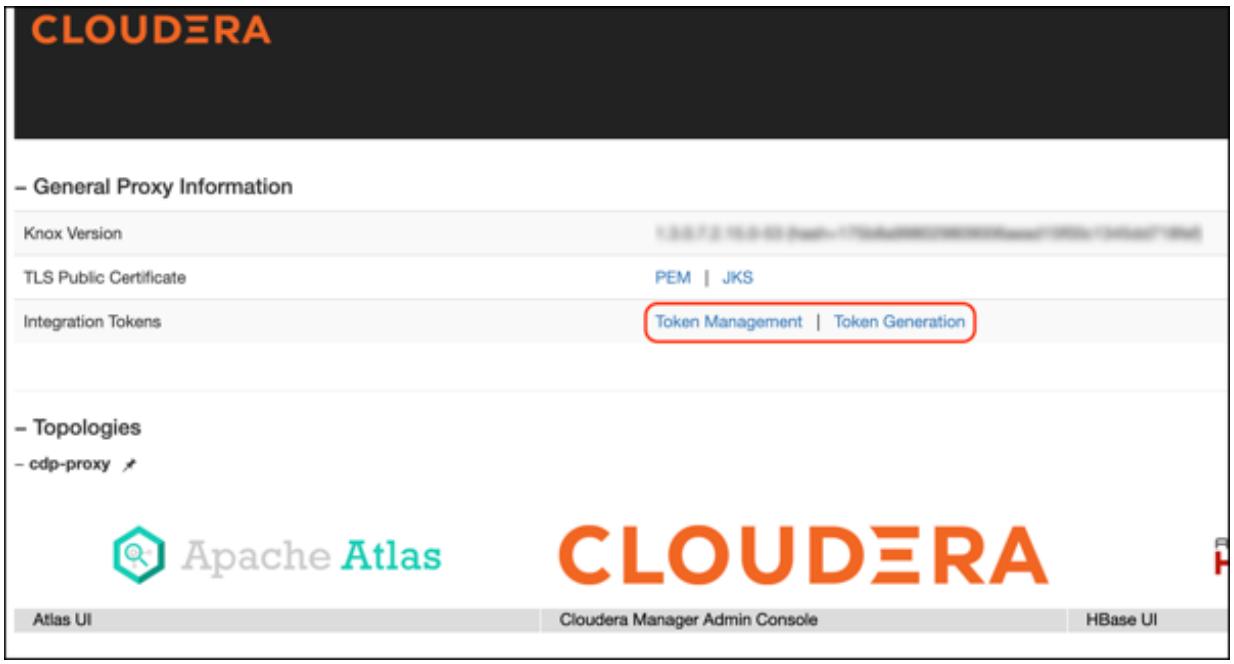
{"FileStatuses":{"FileStatus":[{"accessTime":0,"blockSize":0,"childrenN
um":1,"fileId":16386,"group":"supergroup",
"length":0,"modificationTime":1621238405734,"owner":"hdfs","pathSuffi
x":"tmp","permission":"1777","replication":0,
"storagePolicy":0,"type":"DIRECTORY"},{"accessTime":0,"blockSize":0,"c
hildrenNum":1,"fileId":16387,"group":"supergroup",
"length":0,"modificationTime":1621238326078,"owner":"hdfs","pathSuffi
x":"user","permission":"755","replication":0,
"storagePolicy":0,"type":"DIRECTORY"}]}}
```

Manage Knox Gateway tokens

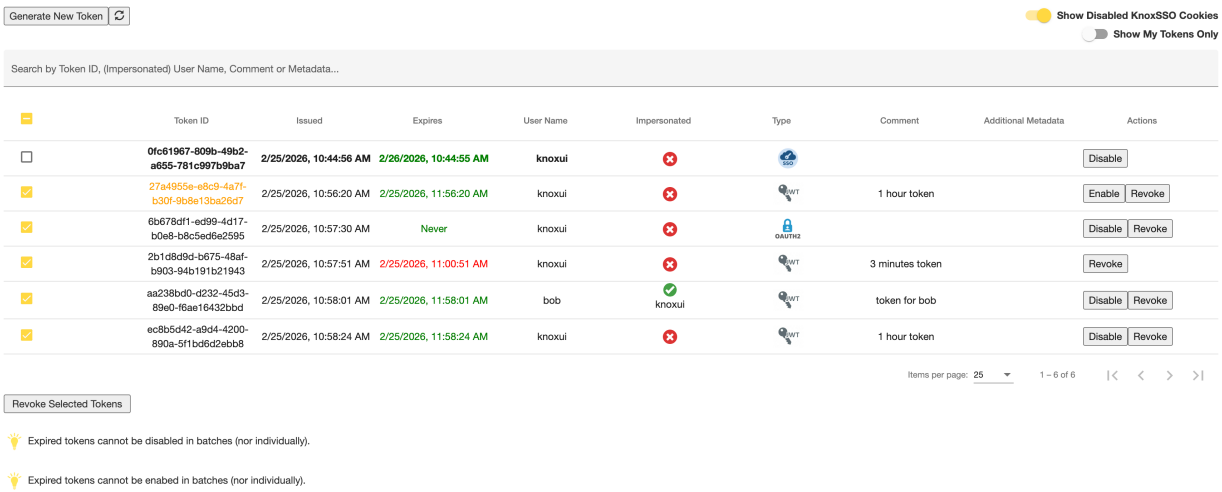
You can enable, disable, or revoke tokens via the Knox homepage.

Procedure

1. To access Knox token management, go to https://*KNOX_GATEWAY_HOST*:*PORT*/*GATEWAY_PATH*/homepage/home, e.g. https://localhost:8443/gateway/homepage/home. Click on Token Management.



A compact view of all tokens generated within the system is shown in a single table with the following information.



a. Each row starts with a selection checkbox for batch operations (except for disabled KnoxSSO cookies, as there is no point in doing anything with them).

b. A unique token identifier. Disabled token's Token ID value is shown in orange.

c. Information on when the token was created and when it will expire.

1. If the token is already expired, the expiration time is shown in red.
2. If the token is still valid, the expiration time is shown in green.

d. Username indicates the user for whom the token is created for.

e. Impersonated is a boolean flag indicating if this is an impersonated token:

- Green check: Yes, this is impersonated. You'll see the user who created the token under the icon.
- Red cross: No, this is not an impersonated token.

f. Indicates which type of token is used for the authentication:

- SSO: marks a KnoxSSO cookie, only available when Knox SSO - Cookie Management is enabled in Cloudera Manager

- OAUTH: participating in OAuth2 type authentication flows such as client_credentials flow used by Data Sharing
- JWT: regular JWT token created by a KNOXTOKEN API call or on the **Token Generation** page

g. Comment: users may add a short comment to the tokens they create to make it easier for them to distinguish certain tokens later (e.g. "1-hour token for user XY")

h. Additional Metadata : In some cases, it's beneficial to add different metadata to the generated token as a key-value pair (e.g. shouldBeRemovedBy=09_Nov_2023). One token can have more than one associated metadata. In this column, we display that information.

i. In the Actions column, you will see

- The enable/disable/revoke actions are visible for impersonated tokens too
- KnoxSSO cookies cannot be revoked nor re-enabled.

In order to refresh the table, you can use the Refresh icon above the table (if you generated tokens on another tab for instance).

2. You can perform batch operations on the tokens. When at least one token is selected, the following buttons are shown under the table:

- Disable - when executed, all the selected tokens become disabled (if they were disabled originally, they will remain disabled). Please note this option is shown only, if there is no expired token selected (i.e. batch disablement only works with live tokens).
- Enable - when executed, all the selected tokens become enabled (if they were enabled originally, they will remain enabled). Please note this option is shown only, if there is no expired token selected (i.e. batch enablement only works with live tokens).
- Revoke - when executed, all the selected tokens will be revoked. Please note this option is shown only, if there is no KnoxSSO cookie (token) selected (i.e. batch revocation only works with regular tokens).



Note: If the selected tokens contain any that cannot be disabled/enabled (expired tokens) or revoked (Knox SSO Cookies), an informational message is displayed below the batch operation buttons indicating the root cause.

3. You can use the Search by field to narrow down tokens by :

- Token ID
- User Name (either own user name or impersonated)
- Comment
- Additional Metadata

4. You can view the disabled Knox cookies or only your tokens by using the following toggle buttons.

- Show Disabled KnoxSSO Cookies - This is true by default. Since disabled KnoxSSO cookies remain in the underlying token state service until they expire, it may bother users to see them in the tokens table. Flipping this toggle button helps to hide them.
- Show My Tokens Only - this toggle button is only visible to users, who can see all tokens. By default, this is false. Enabling it will filter the tokens table in a way such that it will contain tokens only that were generated for the logged-in user (impersonated or not).

5. Click the Refresh icon above the table.

Knox Token API

The Knox Token Service enables the ability for clients to acquire the same JSON Web Token (JWT) that is used for KnoxSSO with WebSSO flows for UIs to be used for accessing REST APIs.

Introduction

By acquiring the token and setting it as a Bearer token on a request, a client is able to access REST APIs that are protected with the JWTProvider federation provider.

The following samples will assume KnoxSSO topology, which is used by Knox for authenticating UIs through SSO. KnoxSSO authentication can be used in the absence of Knox Token Management. To use the Knox Token API, you should copy the hadoop-jwt cookie from your browser and export it as an ENV variable:

```
export HJWT="eyJraWQiOiJtSWdPbjRpSWZ1UmZ1RlVLWjVSb3dxbFh3SGUycGJUcm9lWjlEX1B
LUUJNIiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiJrbm94dWkiLCJraWQiOiJtSWdPbjRpSWZ1UmZ1
RlVLWjVSb3dxbFh3SGUycGJUcm9lWjlEX1BLUUJNIiwiaXNzIjoiS05PWFNTTyIsImV4cCI6MTcw
MjU1MjE1OCwibWFuYWdlZC50b2tlbiI6ImZhbHNlIiwia25veC5pZCI6IjY5M2EwMmE3LTVhYTgt
NDA2MS1iYjMyLTA4MDk4YzdlMTkxYiJ9.tLzmxSd64bAQGpKdETsNXaaDBUKyMZzp0j0YNi-l4Jm
cm1oG5PerUt0OEmLWQnsDBuqtzExkR-g8metxwyIwjV61rqZRXLFycrN8x-nMTCExdxcjtMegIS3
XyETut8MRx8nk6WPVcBlwGHnOCG52CvxsvBe7pUFD4jYYbGzF_WlkPDzPjSRCdQ3xRFDq2IFt7Rx
OIye_50ZdMLbZBm9rNi0RErgdrLKJse68fly-58BcfquubFgWUA0Z0QND7Gg3lPBzyBOhe_5YA23
jQsicgvtc-HhNkY6W2RP-qpXmgjInGcy7dnpvbHXQNfA8cXffdQA6e3bFrTHpjNHgpEsGeg"
```

Alternatively, you can create a custom topology with another authentication provider (PAM, for instance) and use that topology instead of homepage in the following examples.

Important: KnoxToken API 'v1' is no longer supported in 7.2.18 and subsequent Cloudera releases. All the samples use the new 'v2' endpoints with the corresponding HTTP methods (GET, PUT, DELETE).

Acquire a token

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -X GET
 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 11:15:16 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 3073{"access_token":"eyJqa3UiOiJo...4BnSw","token_id":"8f7d
e7f7-a094-4ba4-b64d-f21be86f10eb","managed":"true","target_url":"cdp-proxy-t
oken","homepage_url":"homepage/home?profile=token&topologies=cdp-proxy-token
","endpoint_public_cert":"MIIE1jCCAz...cXff0","token_type":"Bearer","expires
_in":1702469717045,"passcode":"T0dZM1pH...U1ESXo="}
```

In the result JSON, end-users can find the following information:

- accessToken: this is the serialized JWT and is fully compatible with the old-style Bearer authorization method. End-users might want to use it as the 'Token' user in the cdp-proxy-token topology for authentication purposes.
- passcode: this is another sensitive data, the serialized passcode token, which end-users can use as the 'Passcode' user for authentication purposes
- token_id: the unique identifier of the token within Knox
- managed: this is a boolean flag indicating if the token is managed. Managed tokens can be renewed, revoked, disabled, and enabled. By default, in Cloudera, the homepage topology is configured to manage tokens.
- expires: indicates the expiration time of this token. This may be updated with token renewal.

Renew a token

Currently, renewing a token is feasible only if you pass the value of the access_token field as the request payload. For instance:

```
$ export KNOX_TOKEN="eyJqa3UiOiJo...4BnSw"

$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d $KNOX_TOK
EN -X POST 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/
v2/token/renew'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:22:01 GMT
X-Frame-Options: DENY
```

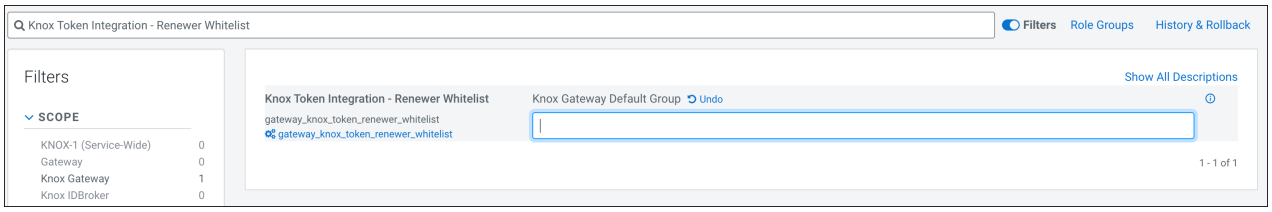
```
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: text/plain
Content-Length: 54

{
    "renewed": "true",
    "expires": "1702567321838"
}
```

The result JSON will tell you if the renewal was a success (this information is stored in the renewed field) and the (new) expiration time. In case of an invalid/unknown token, you should expect a response like this :

```
{
    "renewed": "false",
    "error": "Unknown token: 9caf743e-1e0d-4708-a9ac-a684a576067c"
}
```

Note: Token Renewal is allowed only for a certain set of users, which end-users can define using the Knox Token Integration - Renewer Whitelist Knox configuration on the CM UI.



If the requesting user is an unauthorized caller, you should expect a response like this:

```
{
    "renewed": "false",
    "error": "Caller (myTestUser) not authorized to renew tokens."
}
```

Token revocation

End-users can revoke a token using either the token_id or the access_token fields from the above acquired token response. For instance, the following sample uses the token_id :

```
curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d 'cb538d
38-3076-4a7a-90a5-0f26bff2939a' -X DELETE 'https://[***HOST_NAME***]:8443/
gateway/homepage/knoxtoken/api/v2/token/revoke'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:31:31 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 24

{
    "revoked": "true"
}
```

In case of an invalid or unknown token, you should expect a similar error like this:

```
{
    "revoked": "false",
```

```
    "error": "Unknown token: cb538d38...0f26bff2939a",
    "code": 50
}
```

Token revocation also requires authorization. The same Cloudera Manager configuration should be used that is listed above for token renewals. There is an exception though with revocation: end-users can revoke tokens that belong to them. That is, you can revoke your very own token even if your user name is not defined in the Knox Token Integration - Renewer Whitelist configuration.

Enable/Disable a Token

End-users might need to temporarily disable a token for security purposes and then re-enable the same token. You can do that with the following API calls which use the token_id field from the acquired token response:

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d '1cad
8a13-08e2-4b8a-8076-082678bb641b' -X PUT 'https://[***HOST_NAME***]:8443/
gateway/homepage/knoxtoken/api/v2/token/disable'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:42:57 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 55


{
    "setEnabledFlag": "true",
    "isEnabled": "false"
}


$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d '1cad8
a13-08e2-4b8a-8076-082678bb641b' -X PUT 'https://[***HOST_NAME***]:8443/
gateway/homepage/knoxtoken/api/v2/token/disable'
HTTP/1.1 400 Bad Request
Date: Wed, 13 Dec 2023 15:43:21 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 86


{
    "setEnabledFlag": "false",
    "error": "Token is already disabled",
    "code": 60
}

$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d '1cad
8a13-08e2-4b8a-8076-082678bb641b' -X PUT 'https://[***HOST_NAME***]:8443/
gateway/homepage/knoxtoken/api/v2/token/enable'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:43:45 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 54
```

```
{
    "setEnabledFlag": "true",
    "isEnabled": "true"
}
```

Fetching user tokens

The KnoxToken API provides a powerful way to fetch/filter previously created tokens. See the following samples :

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -X GET
 'https://[***HOST_NAME***]:8443/gateway/homepage/knoxtoken/api/v2/token/
getUserTokens?userName=knoxui'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:46:50 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 413


{"tokens":[{"tokenId":"1cad8a13-08e2-4b8a-8076-082678bb641b","issueTime":
"2023-12-13T15:42:17.965+0000","expiration":"2023-12-13T16:42:17.957+0000","
maxLifetime":"2023-12-20T15:42:17.965+0000","metadata":{"knoxSsoCookie":fals
e,"customMetadataMap":{},"createdBy":null,"comment":null,"enabled":true,"use
rName":"knoxui"},"maxLifetimeLong":1703086937965,"issueTimeLong":17024821379
65,"expirationLong":1702485737957}]}
```

Manage Knox metadata

This document describes how to manage Token Metadata.

As indicated in the previous sections, the KNOXTOKEN service maintains some hard-coded token metadata out-of-the-box:

- userName
- comment
- enabled
- passcode
- createdBy (in case of impersonated tokens)

In Cloudera Runtime version 7.2.16, Cloudera has introduced support for a new feature that allows end-users to add accept query parameters starting with the md_ prefix and treat them as Knox Token Metadata.

Example

```
curl -iku admin:admin-password -X GET 'https://$KNOX_GATEWAY_HOST:8443/gatew
ay/sandbox/knoxtoken/api/v1/token?md_notebookName=accountantKnoxToken&md_sou
ldBeRemovedBy=31March2022&md_otherMeaningfuMetadata=KnoxIsCool'
```

When such a token is created by Knox, the following metadata should be saved:

- notebookName=accountantKnoxToken
- shouldBeRemovedBy=31March2022
- otherMeaningfulMetadata=KnoxIsCool

It will not only enable Knox to save these metadata, but will also enable Knox's existing getUserTokens API endpoint to fetch basic token information using the supplied metadata name besides the username information.

Note: The getUserTokens API returns tokens if any of the supplied metadata exists for the given token. Metadata values may or may not be matched: you can either use the * wildcard to match all metadata values with a given name or you can further filter the stored metadata information by specifying the desired value.

Example:

```
curl -iku admin:admin-password -X GET 'https://$KNOX_GATEWAY_HOST:8443/gatew
ay/sandbox/knoxtoken/api/v1/token/getUserTokens?userName=admin&md_notebookNa
me=accountantKnoxToken&md_name=*'
```

It will return all Knox tokens where metadata with notebookName exists and equals accountantKnoxToken OR metadata with name exists.

Another Sample:

1. Create token1 with md_Name=reina&md_Score=50
2. Create token2 with md_Name=mary&md_Score=100
3. Create token3 with md_Name=mary&md_Score=20&md_Grade=A

The following table shows the returned token(s) in case metadata filtering is added in the getUserTokens API:

Metadata	Token returned
md_Name=reina	token1
md_Name=mary	token2 and token3
md_Score=100	token2
md_Name=mary&md_Score=20	token2 and token3
md_Name=mary&md_Name=reina	token1, token2 and token3
md_Name=*	token1, token2 and token3
md_Uknown=*	Empty list

For more information on sample curl commands, see Managing custom Knox Token metadata.

Configuring Group Impersonation in Knox

Configure group impersonation in Knox to allow users in specific groups to impersonate other users.

About this task

Knox supports group impersonation in addition to user impersonation. This feature enables users in specific groups to impersonate other users, providing greater flexibility and control in managing user impersonation.

If both user impersonation and group impersonation are configured, user impersonation takes precedence. This is because user configurations are more specific and provide finer control.

Configure the hadoop.proxygroup.* parameters to authorize impersonation based on group membership.

Procedure

1. Edit your Knox topology file or shared provider configuration by adding group-based impersonation parameters to the identity provider.

```
                        <provider>
                        <role>identity-assertion</role>
                        <name>Default</name>
                        <param>
```

```
<name>hadoop.proxygroup.analysts.users</name>
<value>hdfs,yarn,hive</value>
</param>
<param>
<name>hadoop.proxygroup.analysts.groups</name>
<value>data-scientists,data-analysts</value>
</param>
<param>
<name>hadoop.proxygroup.analysts.hosts</name>
<value>*.company.com</value>
</param>
</provider>
```

2. Restart Knox to apply the configuration.

Results

Group impersonation is now configured for your Cloudera cluster. Users who belong to the specified groups can impersonate the configured target users based on the defined authorization rules.

How it works:

1. Knox first checks for user-specific impersonation rules by using the hadoop.proxyuser.* parameters.
2. If no user-specific rules exist or they deny access, Knox checks group-based rules by using the hadoop.proxygroup.* parameters.
3. Knox validates the user's group membership against the configured identity provider that is LDAP or Active Directory.
4. If the user belongs to an authorized group, Knox forwards the impersonation request to the appropriate service, such as HDFS, YARN, or Hive.
5. The service processes the request using the permissions of the impersonated service account.

Knox SSO SAML on FIPS-enabled clusters

Supported SAML assertion formats for Knox SSO with the pac4j provider on FIPS-enabled clusters.

Supported SAML assertion formats

Knox SSO uses the pac4j provider with OpenSAML to process SAML assertions from your identity provider (IdP). When the cluster runs in FIPS mode, Knox detects FIPS mode and adjusts how OpenSAML initializes elliptic-curve support so that non-FIPS-compliant elliptic-curve code is not used during SAML processing.

On clusters that are not FIPS-enabled, Knox uses the standard upstream OpenSAML initialization path. Pac4j SAML behavior on those clusters is unchanged.

The following table summarizes SAML assertion protection that Knox supports for pac4j SAML on FIPS-enabled clusters.

IdP assertion protection	Supported on FIPS-enabled clusters
RSA-signed assertions	Yes. Same as on non-FIPS clusters.
RSA-OAEP-encrypted assertions	Yes. Same as on non-FIPS clusters.
ECDH-ES-encrypted assertions	No. ECDH-ES-encrypted SAML assertion decryption is not supported in FIPS mode (empty NamedCurveRegistry).

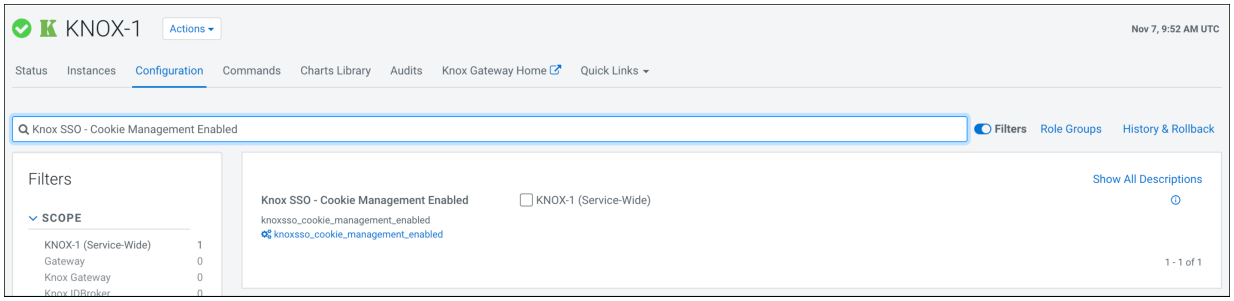
Knox SSO Cookie Invalidation

This feature allows a list of pre-configured superusers to invalidate previously issued Knox SSO tokens for (a) particular user(s) in case there is a malicious attack where one (or more) of those users' SSO tokens get compromised.

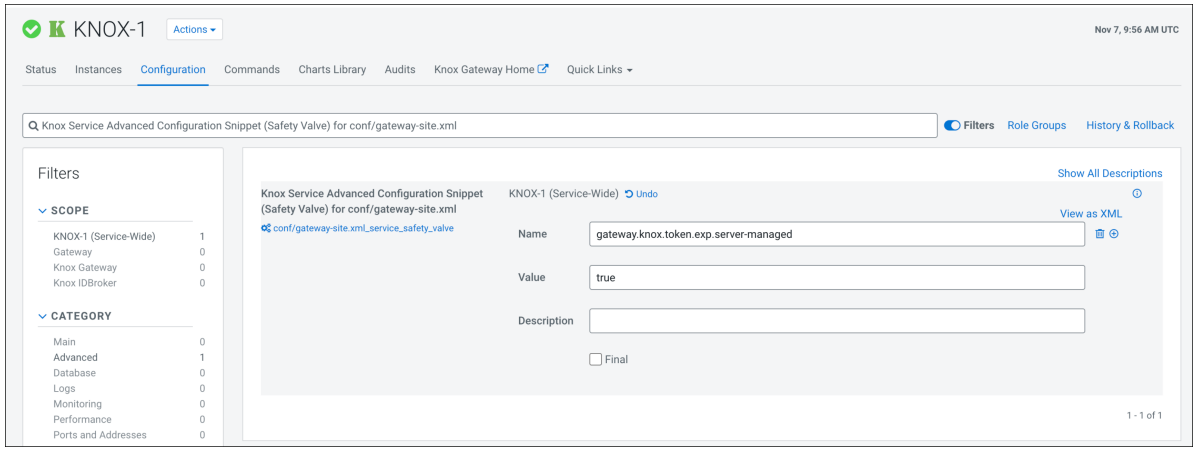
Enabling the feature

By default, the feature is disabled. There are 2 separate steps to enable it:

1. Go to Cloudera Manager Knox Configuration and enable Knox SSO - Cookie Management Enabled.



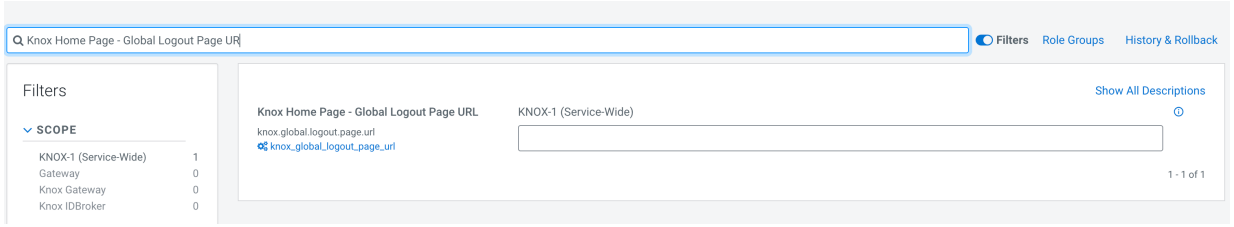
2. In Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml, press +.
   a. In Name, type gateway.knox.token.exp.server-managed.
   b. In Value, type true.
   c. Click Save Changes(CTRL+S)



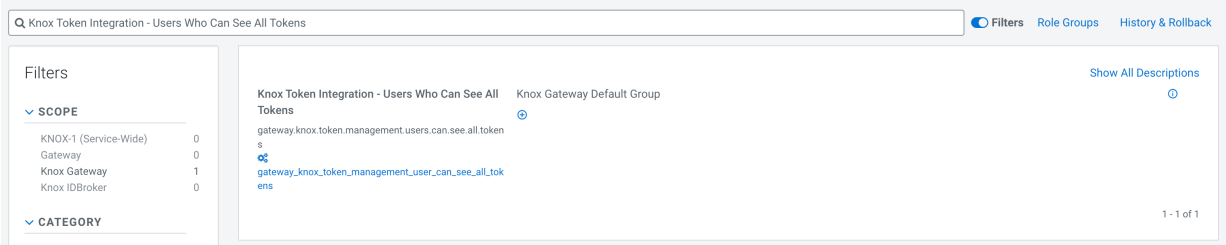
Additional configuration

In addition to enabling the feature, you should review and update the following configuration, if needed:

- Knox Home Page - Global Logout Page URL - when the knoxsso topology is configured to use the Pac4J federation filter (the default configuration in Cloudera on cloud), this configuration is an essential parameter and must not be left empty. This usually points to the logout endpoint of the pre-configured SAML/OIDC callback.



- Knox Token Integration - Users Who Can See All Tokens - A comma-separated list of usernames that can view all tokens on the Token Management page. By default, this is an empty list. Each organization should configure this property to a narrowed set of users, such as security officers, who are authorized to disable SSO cookies in the event of a security breach.



Resolving access issues after global logout

In certain scenarios, users can still access a service through Knox after performing a global logout. This issue occurs when multiple tabs of the same service are open in the browser, and the global logout is performed in one of the tabs. The issue is caused by cookies that are not invalidated across all tabs, allowing some tabs to retain access to a service or services.

To resolve this issue, add the following configuration parameters to Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml in Cloudera Manager:

```
ranger.service.inactivity.timeout=40
atlas.session.timeout.secs=40
knox.global.logout.page.url=<your-idp-logout-url>
```

Note: Replace <your-idp-logout-url> with the actual logout URL of your identity provider.

After adding the configuration parameters:

1. Restart the Knox, Ranger, and Atlas services to apply the changes.
2. Test if the global logout now redirects all tabs to the login page instead of the service home page.

How it works

After enabling the feature, every SSO cookie, the result of a login event through the Knox SSO service, will be recorded in the same database that Knox uses for token management purposes. These SSO cookies are included on the Token Management page. If the logged-in user is a configured "superuser" (added in the above-referenced Users Who Can See All Tokens list), that user is capable of narrowing down user tokens, for whom they suspect are the subject of malicious activities, and disabling the active tokens on the UI (either individually or in batches).

Once a Knox SSO cookie is disabled, it cannot be re-enabled or revoked. Knox has its own cleanup strategy to remove expired tokens from the underlying token state repository (a database in Cloudera on cloud) periodically, on a pre-configured schedule.

It is also important to emphasize that the default Time To Live (TTL) value of Knox SSO cookies is set to 1 day by default. It is highly recommended that organizations overview their own UI jobs and reduce this value to as short as possible to reduce the security risk involved here.

Related Information

Adjust the lifetime of Knox SSO session tokens

Configure custom SSO cookie name for Knox

Learn how to configure a custom SSO cookie name for Knox to enable concurrent SSO sessions across different clusters without authentication conflicts.

About this task

Knox, by default, uses hadoop-jwt as the cookie name for SSO authentication for all Hadoop services. When accessing different clusters configured for SSO authentication, using the same default cookie name causes conflicts because the web browser uses the cookie from one cluster to attempt authentication with another cluster, resulting in access issues.

To resolve this issue, you must configure a unique cookie name for each cluster's Knox SSO service.

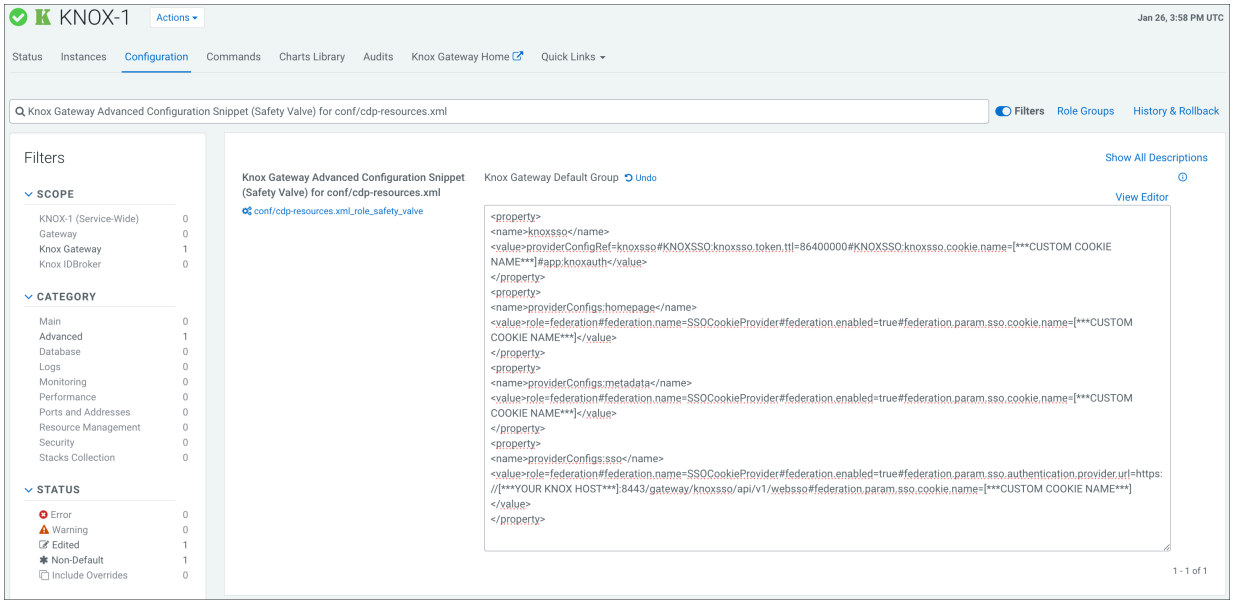
Procedure

1. In Cloudera Manager, select the Knox service.
2. Go to the Configuration tab.
3. Search for Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml.
4. Click View as XML and add the following configuration properties:

```
<property>
<name>knoxsso</name>
<value>providerConfigRef=knoxsso#KNOXSSO:knoxsso.token.ttl=86400000#KNOX
SSO:knoxsso.cookie.name=[***CUSTOM COOKIE NAME***]#app:knoxauth</value>
</property>
<property>
<name>providerConfigs:homepage</name>
<value>role=federation#federation.name=SSOCookieProvider#federation.enabl
ed=true#federation.param.sso.cookie.name=[***CUSTOM COOKIE NAME***]</va
lue>
</property>
<property>
<name>providerConfigs:metadata</name>
<value>role=federation#federation.name=SSOCookieProvider#federation.enable
d=true#federation.param.sso.cookie.name=[***CUSTOM COOKIE NAME***]</value>
</property>
<property>
<name>providerConfigs:sso</name>
<value>role=federation#federation.name=SSOCookieProvider#federation
.enabled=true#federation.param.sso.authentication.provider.url=https
://[***YOUR KNOX HOST***]:8443/gateway/knoxsso/api/v1/websso#federation.p
aram.sso.cookie.name=[***CUSTOM COOKIE NAME***]</value>
</property>
```

Note:

- Replace *[***CUSTOM COOKIE NAME***]* with a unique cookie name for your environment. Use a different cookie name for each cluster to avoid conflicts.
- Replace *[***YOUR KNOX HOST***]* in the federation.param.sso.authentication.provider.url value under providerConfigs:sso with your Knox host. If your Knox instance uses a different port than the default 8443, update the port number as well.



5. Click Save Changes(CTRL+S).
6. Refresh the Knox instances configuration by clicking the Stale Configuration: Restart needed indicator and wait until the refresh process completes.
7. Verify that the new cookie name is in use by accessing the Knox UI and checking the browser's cookie storage.

   After logging in, you should see the custom cookie name you configured instead of the default hadoop-jwt.

Results

You can now access different clusters with Knox SSO enabled simultaneously without authentication conflicts. Each cluster will use its own unique JWT cookie name for session management.

Adjust the lifetime of Knox SSO session tokens

How to change the Time To Live (TTL) of Knox SSO session tokens using Cloudera Manager.

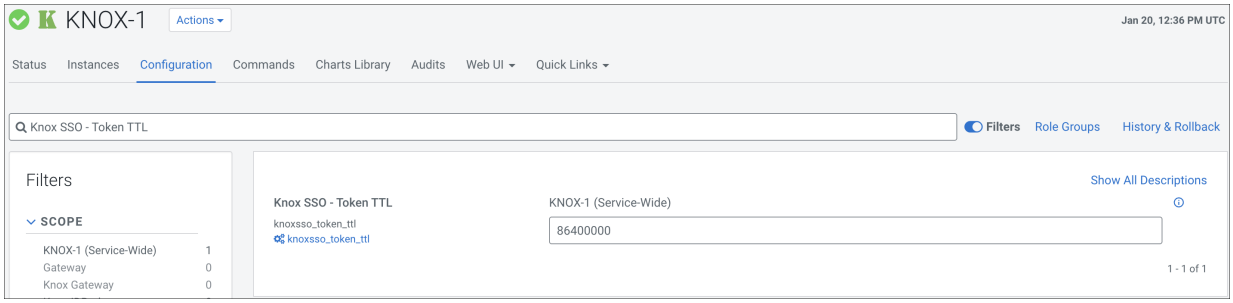
About this task

By default, Knox SSO session tokens expire after 24 hours. Depending on organization policies, you may need to reduce this value, as some organizations have security policies that require a shorter token lifetime.

The default Time To Live value is 86400000 milliseconds, which is equivalent to 24 hours. You can adjust the expiration time by updating the Knox SSO configuration in Cloudera Manager.

Procedure

1. In Cloudera Manager, select the Knox service.
2. Go to Configuration.
3. Search for the Knox SSO - Token TTL property.



4. Update the default value to match the duration that is most suitable to your organization's requirements.

   Note: The value is specified in milliseconds. For instance, a one-hour Time To Live value is 3600000 milliseconds.

5. Click the Save Changes(CTRL+S) button.
6. Refresh the Knox instances configuration by clicking the Stale Configuration: Restart needed indicator and wait until the refresh process completes.

Concurrent session verification (Tech Preview)

This feature is a security measure that enables end-users limiting the number of concurrent UI sessions the users can have. To achieve this goal the users can be sorted out into three groups: non-privileged, privileged, unlimited.

The non-privileged and privileged groups each have a configurable limit, which the members of the group can not exceed. The members of the unlimited group are able to create an unlimited number of concurrent sessions.

All of the users, who are not configured in either the privileged or in the unlimited group, shall become the member of the non-privileged group by default.

Note: Conccurent session verification feature is under Technical Preview. The technical preview feature and considered under development. Do not use this in your production systems. To share your feedback, contact Support by logging a case on our Cloudera Support Portal. Technical preview features are not guaranteed troubleshooting guidance and fixes.

Configuration

The following table shows the relevant gateway-level parameters that are essential for this feature to work:

Parameter	Description	Default
gateway.service.concurrentsessionverifier.impl	To enable the session verification feature, end-users should set this parameter to org.apache.knox.gateway.session.control.InMemoryConcurrentSessionVerifier	org.apache.knox.gateway.session.controlonVerifier
gateway.session.verification.privileged.users	Indicates a list of users that are qualified “privileged”.	Empty list
gateway.session.verification.unlimited.users	Indicates a list of (super) users that can have as many UI sessions as they want.	Empty list
gateway.session.verification.privileged.user.limit	The number of UI sessions a “privileged” user can have	3
gateway.session.verification.non.privileged.user.limit	The number of UI sessions a “non-privileged” user can have	2

How this works

If the verifier is disabled it will not do anything even if the other parameters are configured.

When the verifier is enabled all of the users are considered as a non-privileged user by default and they will not be able to create more concurrent sessions than the non-privileged limit. The same is true after you added someone in the privileged user group: that user will not be able to create more UI sessions than the configured privileged user limit.

Whereas the members of the unlimited users group are able to create an unlimited number of concurrent sessions even if they are configured in the privileged group as well.

In Cloudera, currently, there are no first-class Cloudera Manager parameters for this feature, so all of those properties have to be set through Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml configuration in Cloudera Manager.

