Cloudera Runtime 7.1.8

Apache Knox Authentication

Date published: 2020-07-28
Date modified: 2022-12-15

https://docs.cloudera.com/

Legal Notice

Contents

Apache Knox Overview 4
Securing Access to Hadoop Cluster: Apache Knox 4
Apache Knox Gateway Overview 4
Knox Supported Services Matrix 5
Load balancing for Apache Knox 6

Proxy Cloudera Manager through Apache Knox 7

Installing Apache Knox 8
Apache Knox Install Role Parameters 9

Knox Gateway token integration 11
Overview 12
Token configurations 13
Generate tokens 18
Manage Knox Gateway tokens 20
Manage Knox metadata 22

Concurrent session verification (Tech Preview) 23

Apache Knox Overview

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 CDP Private Cloud Base Cluster

- Authentication: Kerberos

  Cloudera uses Kerberos for authentication. Kerberos is an industry standard used to authenticate users and resources within a Hadoop cluster. CDP 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.
- Authorization: Ranger

  OS Security: Data Encryption and HDFS

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)	#	
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		
Hue	#	
Impala HTTP JDBC API		#
Impala UI	#	
JobHistory UI	#	
JobTracker		#
Kudu UI	#	
Livy API + UI	#	#
LogSearch		
NameNode	#	#
NiFi	#	#
NiFi Registry	#	#
Oozie API	#	#
Oozie UI	#	
Phoenix (aka Avatica)		#

Component	UI Proxy (with SSO)	API Proxy
Profiler	#	
Ranger API	#	#
Ranger UI	#	
ResourceManager API	#	#
Schema Registry API + UI	#	#
Streams Messaging Manager (SMM) API	#	#
Streams Messaging Manager (SMM) UI	#	
Solr	#	#
Spark3History UI	#	
SparkHistory UI	#	
Storm		
Storm LogViewer		
Superset		
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.

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 in the Public cloud:

- Hive
- 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

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).

Installing Apache Knox

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

About this task

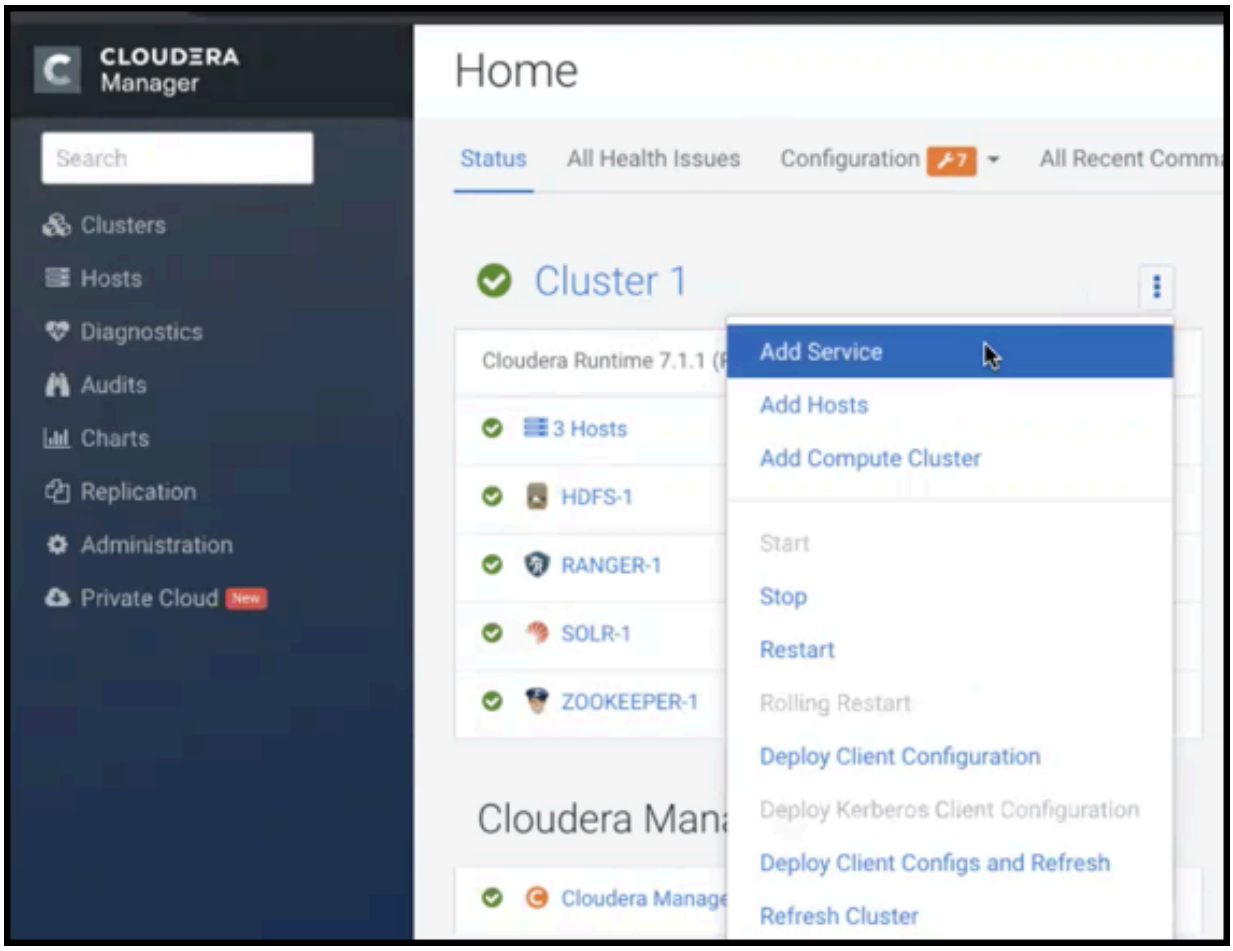
Apache Knox is an application gateway for interacting with the REST APIs and UIs. The Knox Gateway provides a single access point for all REST and HTTP interactions in your Cloudera Data Platform cluster.

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 CDP Private Cloud.
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
save_alias_command_input_password	No	Text	-

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_port	No	Port	8443
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 (Admin UI and Home Page). 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.

Knox IDBroker role parameters

Note: Knox IDBroker is not currently supported in CDP Private Cloud.

Table 5: Required parameters for Knox IDBroker role

Name	In Wizard	Type	Default Value
idbroker_master_secret	Yes	Password	-
idbroker_conf_dir	Yes	Path	/var/lib/knox/idbroker/conf
idbroker_data_dir	Yes	Path	/var/lib/knox/idbroker/data
idbroker_gateway_port	No	Port	8444
idbroker_gateway_path	No	Text	gateway
idbroker_heap_size	No	Memory	1 GB (min = 256 MB; soft min = 512 MB)

Table 6: Optional parameters for Knox IDBroker role

Name	Type	Default Value
idbroker_aws_user_mapping	Text	-
idbroker_aws_group_mapping	Text	-
idbroker_aws_user_default_group_mapping	Text	-
idbroker_aws_credentials_key	Password	-
idbroker_aws_credentials_secret	Password	-
idbroker_gcp_user_mapping	Text	-
idbroker_gcp_group_mapping	Text	-
idbroker_gcp_user_default_group_mapping	Text	-
idbroker_gcp_credential_key	Password	-
idbroker_gcp_credential_secret	Password	-
idbroker_azure_user_mapping	Text	-
idbroker_azure_group_mapping	Text	-
idbroker_azure_user_default_group_mapping	Text	-
idbroker_azure_adls2_tenant_name	Text	-
idbroker_azure_vm_assumer_identity	Text	-
idbroker_relaodable_refresh_interval_ms	Time	10 seconds (minimum = 1 second)
idbroker_kerberos_dt_proxyuser_block	Text Array	A comma-separated list of proxy user configuration used in Knox's dt topology in case Kerberos is enabled
idbroker_knox_token_ttl_ms	Time	1 hour (minimum = 1 second)

Knox Gateway token integration

As of CDP 7.2.14, you can use Apache Knox homepage to generate and manage Knox Gateway tokens for CDP Public Cloud.

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).

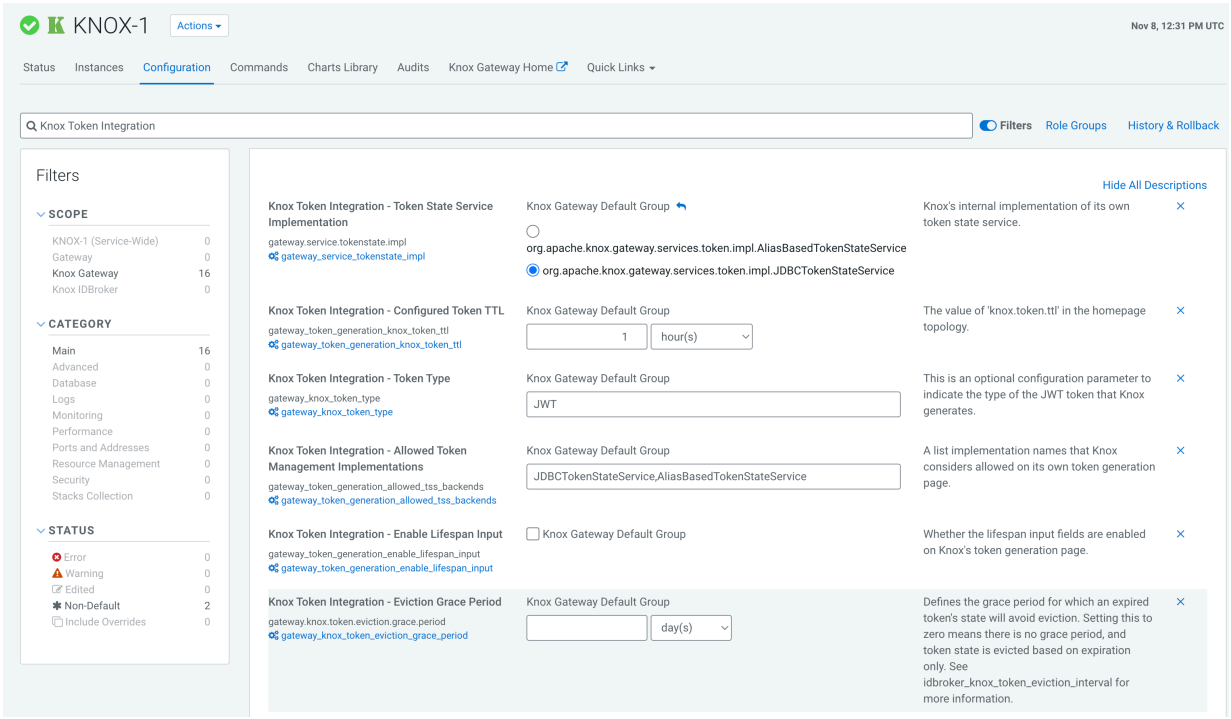
About Knox gateway tokens

Before CDP 7.2.14, Knox on CDP Public Cloud 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.

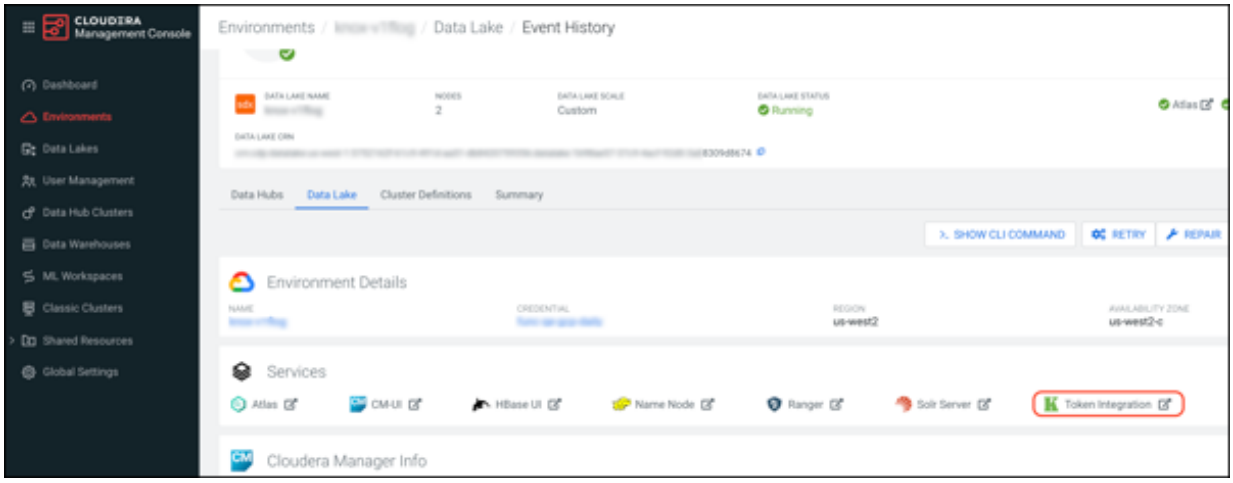
View Knox token integration

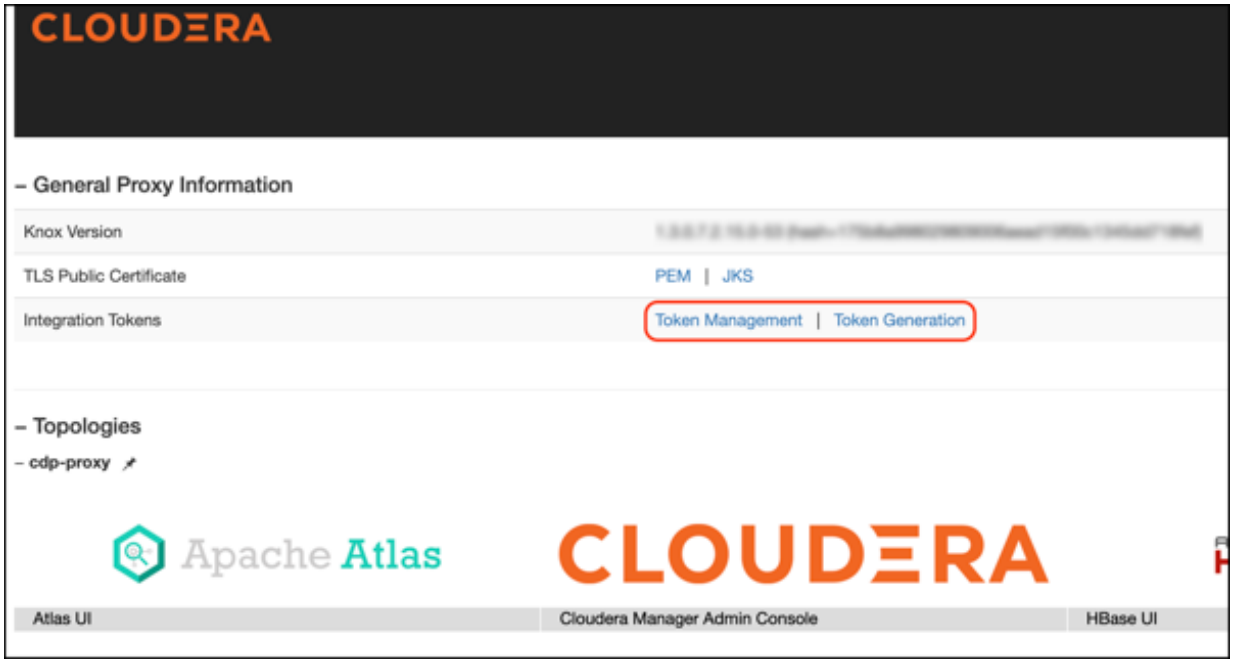
Knox token integration can be accessed via Cloudera Manager or the Knox homepage:

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



- Navigate to the Management Console service > Data Lakes > (Your cluster) > Token Integration (under the Services tab). This will bring you to the Knox homepage. There are two new links on your Knox homepage homepage: Token Management and Token Generation.





Knox token integration in CDP 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.

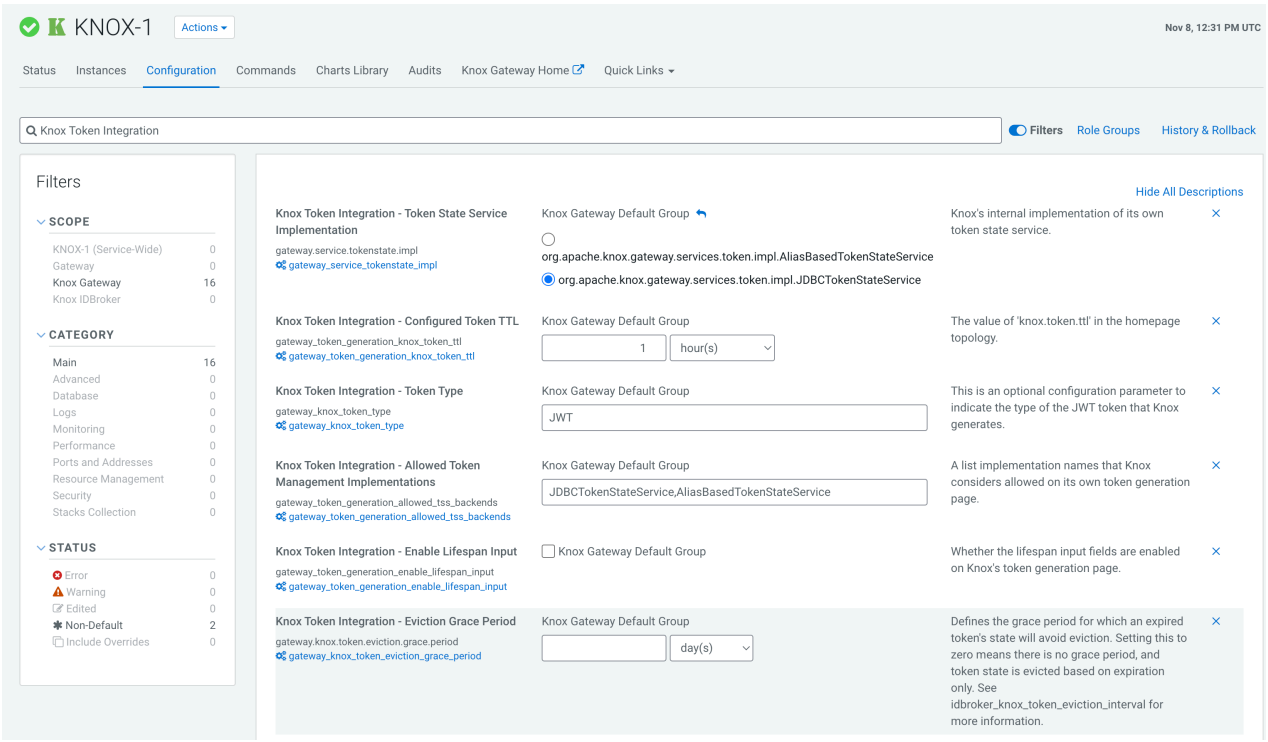
Default configurations

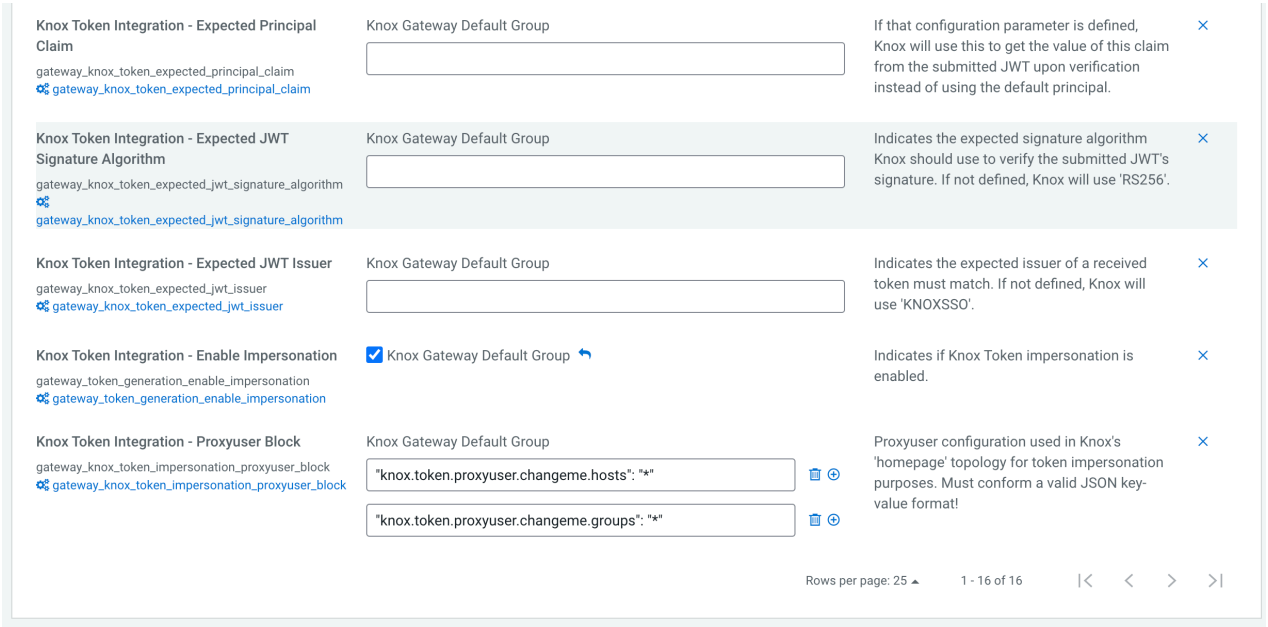
Table 7: Default token configurations

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
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

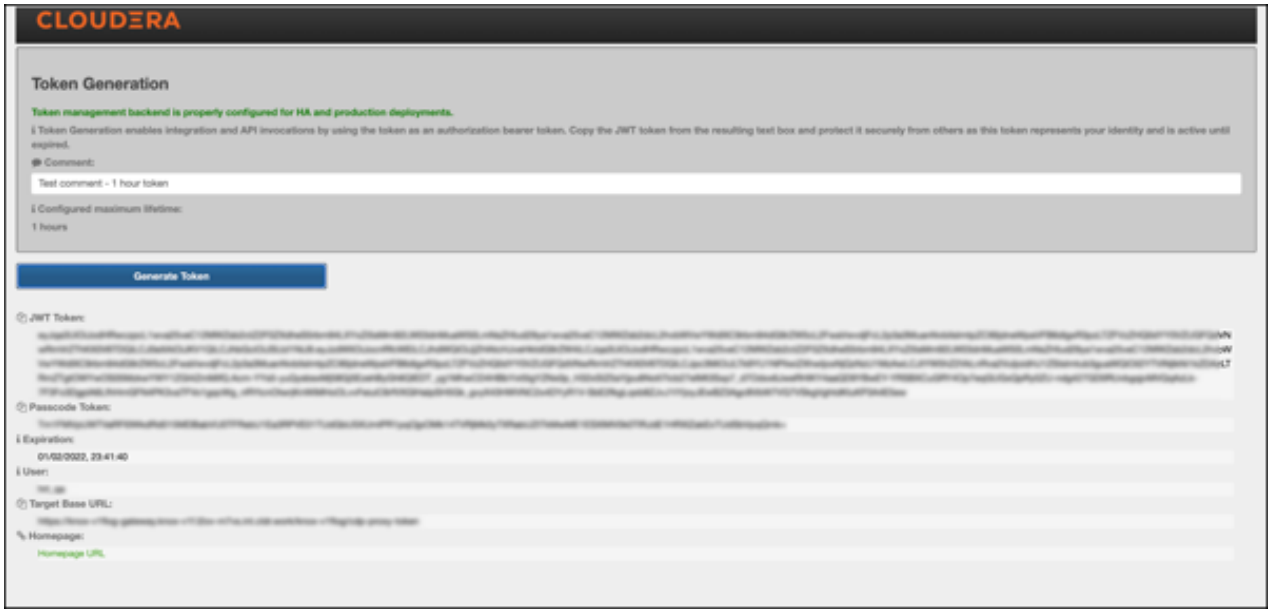
Property	Sample values	Default
Proxyuser Block	Proxyuser configuration used in Knox's 'homepage' topology for token impersonation purposes. Must conform a valid JSON key-value format!	"knox.token.proxyuser.changeme.hosts": "*" "knox.token.proxyuser.changeme.groups": "*"

Default configurations seen from Cloudera Manager:





Default configurations seen from the Knox homepage UI:



Database connection properties

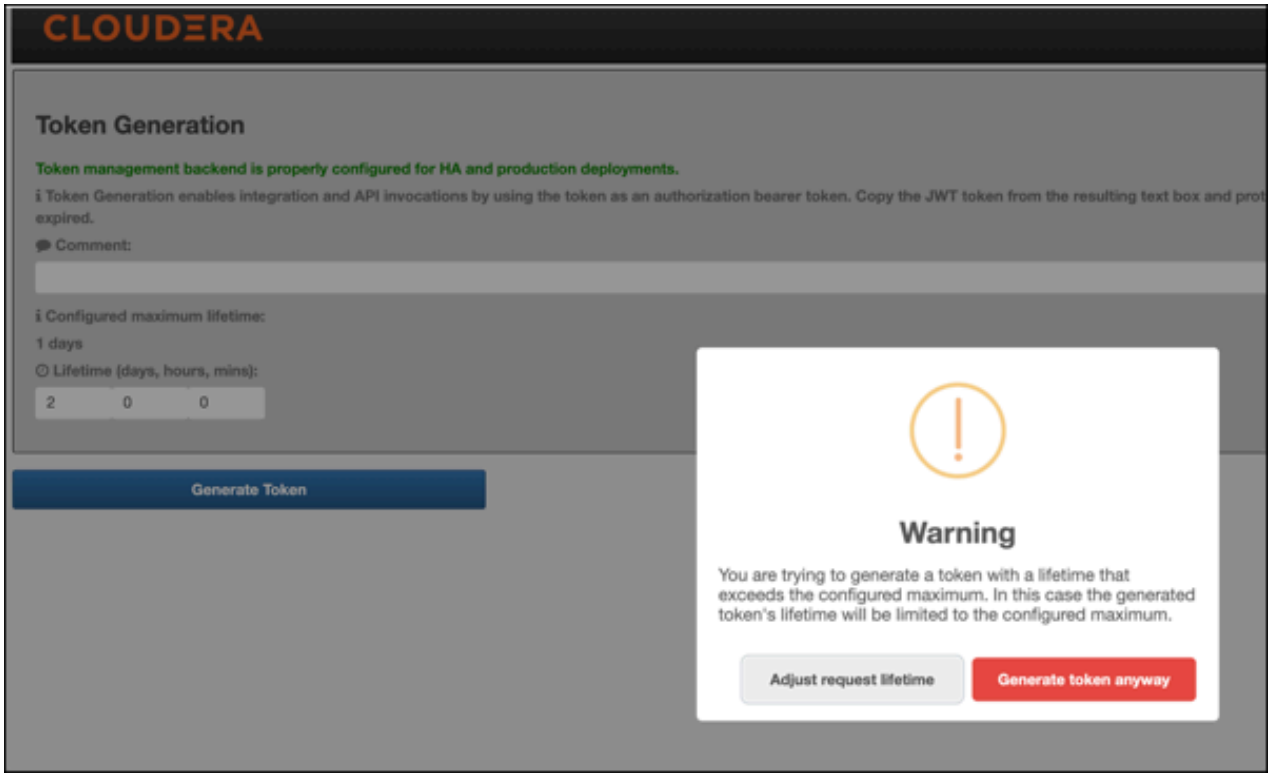
Optional database connection properties that you can declare individually:

- 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.

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

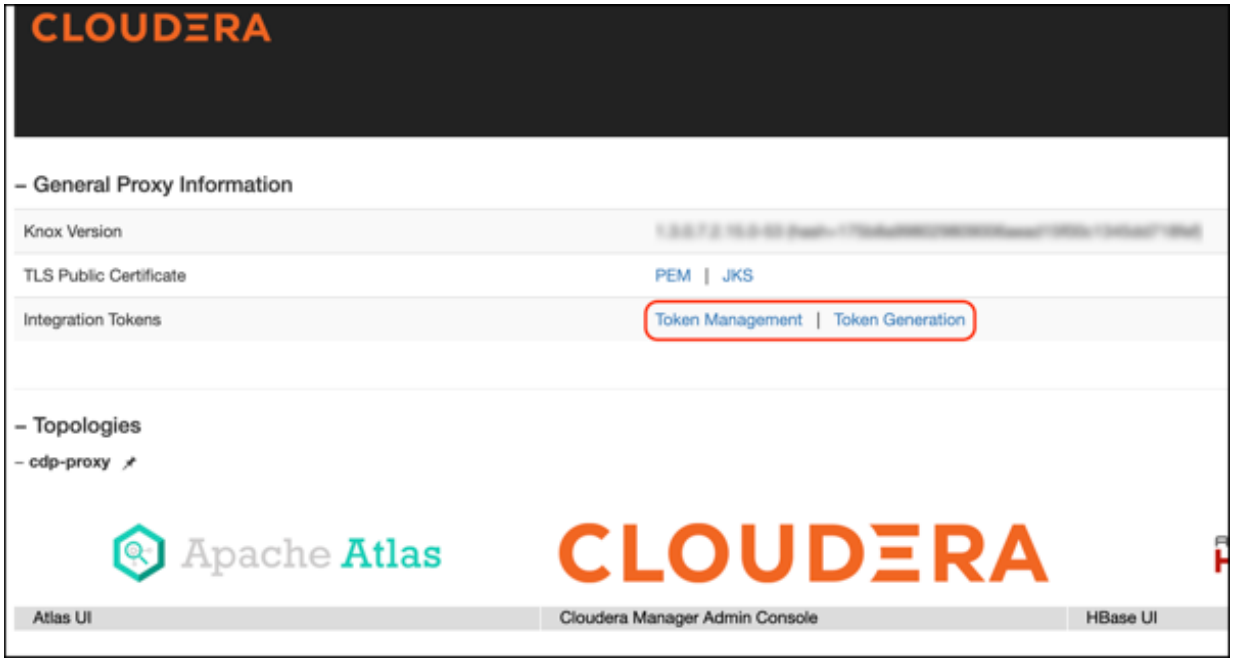
Option	Description	Sample values
jwkAlg	(Optional) The desired JSON Web Signature algorithm name. Determines if the gateway-level alias is configured with a 256, 384, or 512-bit length JWK.	HS256 (Default) HS384 HS512
saveAlias	(Optional, Recommended) Given alias name used to save the generated JWK,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

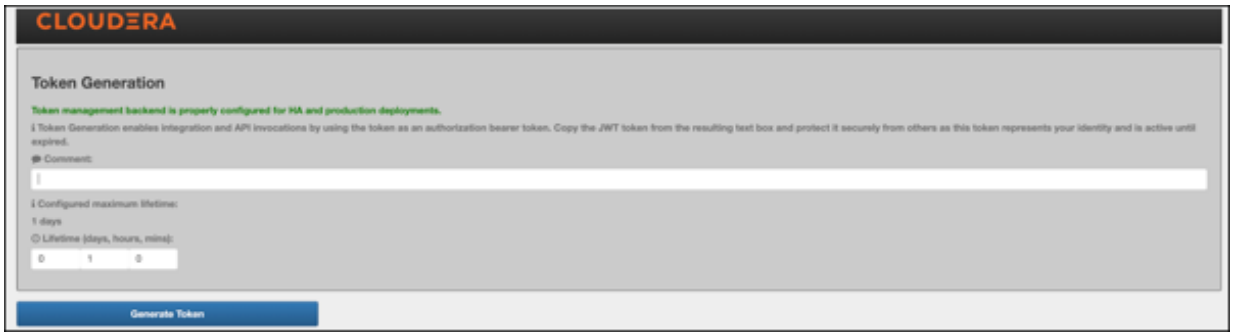
How to generate Knox gateway tokens from the Knox homepage.

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.



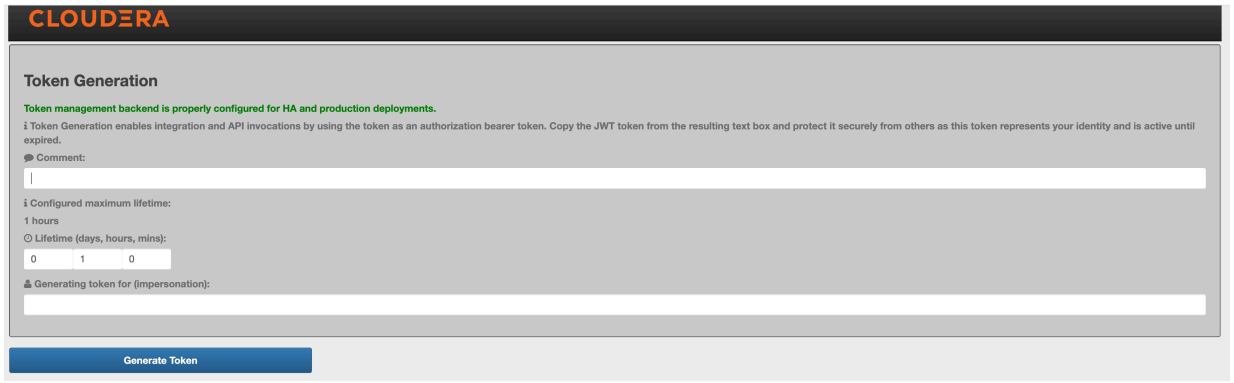
2. The following sections are displayed on the page:
   - Status bar: Message about the configured token state backend. There are 3 different statuses:
     - ERROR: Displayed in red. Indicates a problem with the service backend which makes the feature not work. Usually, this is visible when end-users 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).

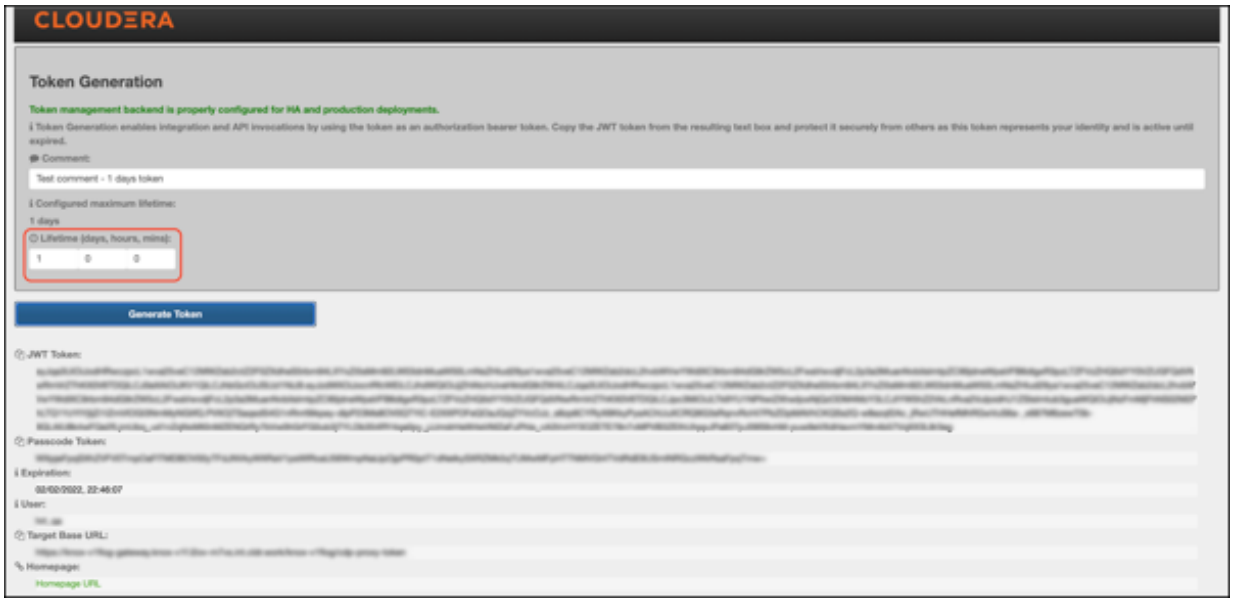
Using that input field our customers should be able to generate tokens on behalf of other users. For this to work, the Knox Token Integration - Proxyuser Block property has to be configured properly.

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:

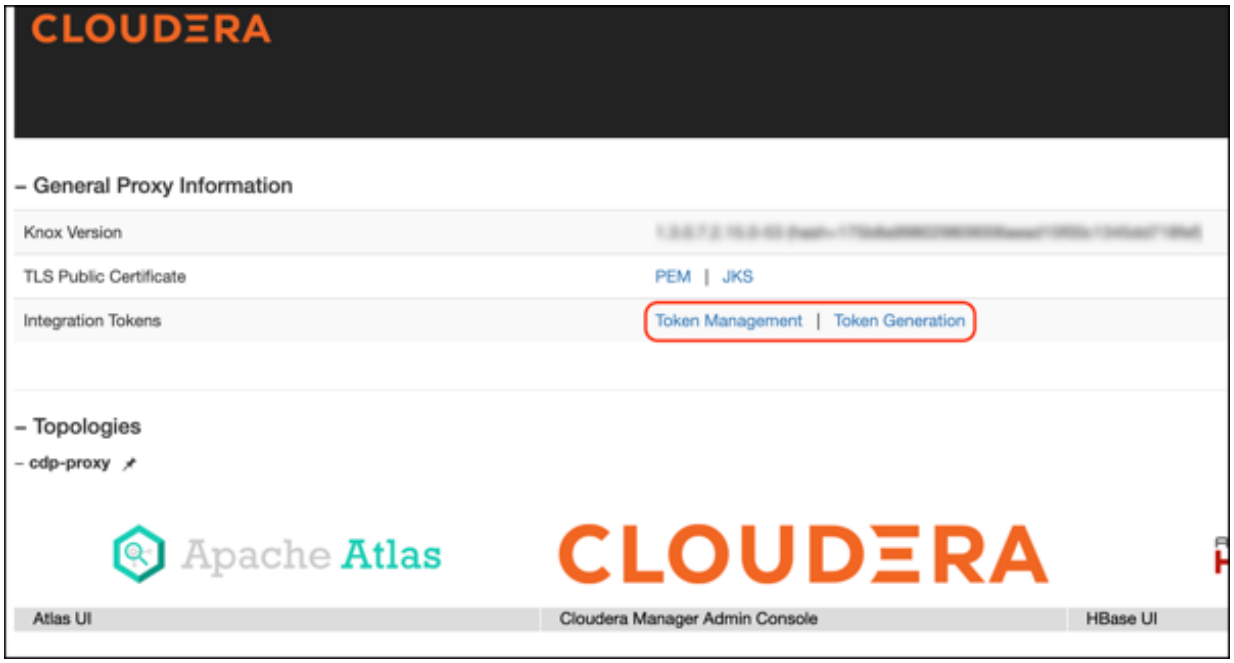
     ```
     $ curl -ku Passcode:WkRFMk1XTmh[...]RVNFpXRTA= https://localhost:8443/ga
     teway/cdp-proxy-token/webhdfs/v1?op=LISTSTATUS

     {"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.

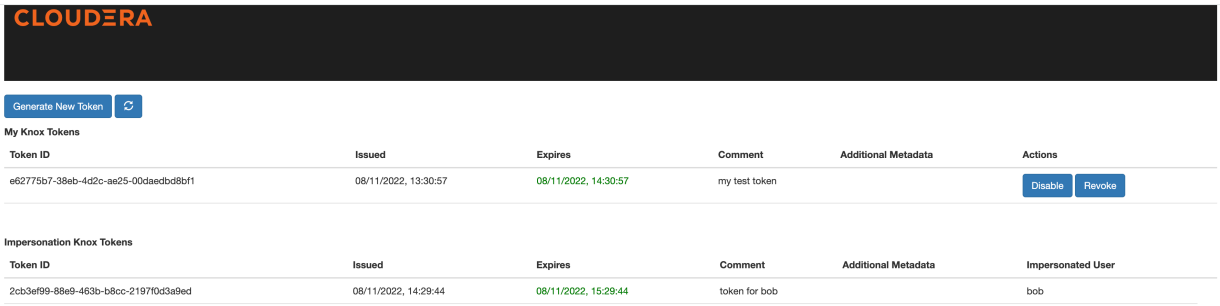


Active token will be displayed in green; expired tokens are red.

If Knox Token Integration - Enable Impersonation is set to true, the logged user will see two tables:

a. Tokens of the logged-in user
b. Tokens the logged-in user has generated for other users (impersonation)

The tables also display information about additional metadata



2. On this page, you will see basic information about your generated token(s) and you can execute the following actions:

   - Enable/Disable: Temporarily enable/disable a token.

     Note: Disabled tokens are not allowed to be use for authentication purposes.

   - Revoke: Permanently remove the token from the persistent store.

     Caution: This action cannot be undone; once you revoke a token, Knox will delete it from the in-memory cache and the underlying persistent token storage.

3. Click the Refresh icon above the table.

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.

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:

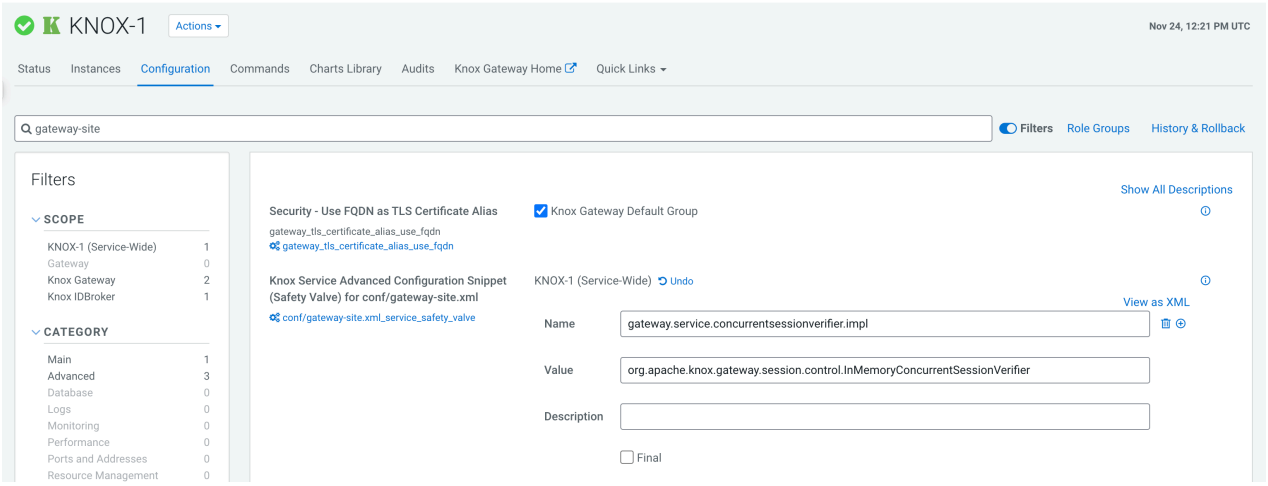
Parameter	Description	Default
gateway.service.concurrentsessionverifier.impl	To enable the session verification feature, end-users should set this parameter to org.apache.knox.gateway.session.control.InMemoryConcurrentSessionVerifier	org.apache.knox.gateway.session.contronVerifier
gateway.session.verification.privileged.users	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 Data Platform, 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.


KNOX-1
Actions
Nov 24, 12:21 PM UTC
Status
Instances
Configuration
Commands
Charts Library
Audits
Knox Gateway Home
Quick Links
gateway-site
Filters
Role Groups
History & Rollback
Filters
SCOPE
KNOX-1 (Service-Wide) 1
Gateway 0
Knox Gateway 2
Knox IDBroker 1
CATEGORY
Main 1
Advanced 3
Database 0
Logs 0
Monitoring 0
Performance 0
Ports and Addresses 0
Resource Management 0
Show All Descriptions
Security - Use FQDN as TLS Certificate Alias
gateway_tls_certificate_alias_use_fqdn
gateway_tls_certificate_alias_use_fqdn
Knox Gateway Default Group
Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml
conf/gateway-site.xml_service_safety_valve
KNOX-1 (Service-Wide)
Undo
View as XML
Name
gateway.service.concurrentsessionverifier.impl
Value
org.apache.knox.gateway.session.control.InMemoryConcurrentSessionVerifier
Description
Final