

Cloudera Runtime 7.1.8

Atlas Import Utility

Date published: 2019-09-23

Date modified: 2022-08-30

CLOUDERA

<https://docs.cloudera.com/>

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Importing Hive Metadata using Command-Line (CLI) utility.....	4
Bulk and migration import of Hive metadata.....	5
Using Atlas-Hive import utility with Ozone entities.....	6
 Setting up Atlas Kafka import tool.....	 6

Importing Hive Metadata using Command-Line (CLI) utility

You can use the Atlas-Hive import command-line utility to load Atlas with databases and tables present in Hive Metastore.

This utility supports importing metadata of a specific table, tables from a specific database or all databases and tables.



Important: You must add Atlas Gateway role to all hosts which have the Hook-based services like Hive / Hive on Tez / Impala / HBase / Spark / Kafka.

Consider a scenario where Hive has databases and tables prior to enabling Hive hook for Atlas. In such a situation, the Atlas-Hive import utility can be employed to ensure Hive and Atlas are in sync.

Also, the utility can be used in a scenario where Atlas is unable to process specific messages due to some errors that could possibly occur with Kafka.



Important: You must run the tool on the Hive node.



Attention: Ensure that the end user (e.g. hive) running the import-hive script has permissions in Ranger to create, update, and delete Atlas entities. The user must also have sufficient permissions to access Hive databases, tables, and columns that need to be imported into Atlas.

Supported Hive Metadata import options:

Atlas-Hive utility supports various options which can be used while importing Hive Metadata:

- `-d <database regex>` OR `--database <database regex>`

Specify the database name pattern which has to be synced with Atlas.

- `-t <table regex>` OR `--table <table regex>`

Specify the table name pattern which has to be synced with Atlas. It must be used along with `-d`.

- `-f <filename>`

Imports all databases and tables in the specified file. The file must have one entry on each line where the entry is in the form of `<database_name>:<table_name>`.

For example:

A scenario where the user wants to import two tables named t11 from database db1 and t21 from db2 and all tables from db3. The file content must be:

- db1:t11

- db2:t21

- db3

- No options

Does not specify any option to import all databases from Hive into Atlas.

A sample usage of the script Atlas hook in Hive is not configured and hence no “Live” data gets reflected into Atlas.

Later, you configure the Atlas hook but it is observed that the Hive database already contains entities that need to reflect in Atlas. In such cases, an Atlas-hive import script reads the database and tables from Hive Metadata and creates entities in Atlas.

An example of Atlas-hive script:

Usage 1: <atlas bundle>/hook-bin/import-hive.sh

Usage 2: <atlas bundle>/hook-bin/import-hive.sh [-d <database regex> OR --database <database regex>] [-t <table regex> OR --table <table regex>]

Usage 3: <atlas bundle>/hook-bin/import-hive.sh [-f <filename>]

File Format:

database1:tbl1

database1:tbl2

database2:tbl1

Limitations of using Atlas-Hive import script

The Atlas-Hive import utility has the following limitations:

- Cannot delete entities which are dropped from Hive but do exist in Atlas.
- Cannot create lineages.



Note: If a specified entity is not available in Atlas, the entry gets created in Atlas. If a specified entity exists in Atlas, it gets updated with the latest metadata state present in Hive.

Bulk and migration import of Hive metadata

The existing sequential import of Hive metadata into Apache Atlas is time consuming. The new bulk and migration import process which creates a ZIP file first and then runs the Atlas bulk import using the ZIP file, improves performance and is faster. An improvised method to import Hive metadata into Atlas is now available.

The [previous version](#) of Hive Metadata import was a sequential import into Atlas. The time taken for manual import of databases to the Atlas system consumes more time. With the need to import more databases, additional time is consumed in completing the import process and the process takes longer to complete.

Instead of Atlas sequential import, this update enables you to import without the Atlas interaction, where the existing Hive import is updated to import the Hive databases and/or tables into ZIP files. And later, you can import the ZIP file into Atlas using a bulk or migration import process.

By default, the enhanced Hive metadata import creates the ZIP file and then automatically runs Atlas bulk import using the same ZIP file.

As an example, use the following commands to perform the enhanced new Hive metadata import.

```
./import-hive.sh -t hive_db_vxofz.hive_table_tmwos -o /tmp/import.zip
```

Optionally, you can perform Hive metadata import using the following two phases separately to use either the bulk or migration import process:

1. Create the Hive metadata export ZIP file.
2. Use the exported ZIP file to run the bulk or migration import process for Atlas.



Important: [Migration import](#) is performant than bulk import (The default without option -i) . But during the migration import process, Atlas blocks all the REST API calls and Atlas Hook notification processing.

As an example, use the following commands to perform the new Hive metadata import in phases.

1. To create the Hive metadata export ZIP file, run the following command:

```
./import-hive.sh -i -t hive_db_vxofz.hive_table_tmwos -o /tmp/import.zip
```

Option	Description
--------	-------------

-o or --output	Used to create the ZIP file. <ul style="list-style-type: none"> If you specify a ZIP file followed by the -o option, it creates the ZIP file. If you provide only a path, then by default this option tries to create a import-hive-output.zip file inside the path. If the ZIP file already exists, the import process fails.
-i or --ignorebulkImport	Used to ignore the second phase, that is the bulk import execution (the default behavior is to invoke the bulk import process)

2. To Import the create a ZIP file using bulk import, use the CURL command:

```
cat > /tmp/importOptions.json
```

```
{ "options": { "updateTypeDefinition": "false", "format": "zipDirect", "size": "1" } } kinit -kt /cdep/keytabs/atlas.keytab atlas
```

```
curl -k --negotiate -u : -X POST
```

```
'https://quasar-cgfwjs-1.quasar-cgfwjs.root.hwx.site:31443/api/atlas/admin/import' --form 'request=@/tmp/importOptions.json' --form 'data=@/tmp/import.zip'
```

Instead of the bulk import, run the migration import process and follow the [Migration import](#) documentation.



Note: While performing the import operation using the API with this enhancement, the deleteNonExisting option is not supported. For example, when you create tables in a database, perform an import, and drop the table, and later perform an import with deleteNonExisting, the dropped table status shows as “Active” in Atlas. You must perform this operation using the [previous version](#) of Hive Metadata import.

Using Atlas-Hive import utility with Ozone entities

The Atlas-Hive import script also supports creating the corresponding Ozone entities into Atlas, if the Hive entity has an Ozone path associated with it.

For example, if a table was created using an Ozone path, the Atlas-Hive import script creates Ozone related entities into Atlas.



Attention: If a table was loaded using a file located into an Ozone path, Ozone entities will not be created into Atlas as HMS does not furnish the details.

Setting up Atlas Kafka import tool

You can use the Atlas-Kafka import tool to load Atlas with Kafka topics.

You must first set up the tool and later run the tool manually to create or update the Kafka topic entities in Atlas. The tool uses client configuration to fetch the required configuration, like the Atlas endpoint and Zookeeper.



Important: You must run the tool on the Kafka node.

The Kafka type definition in Atlas contains multiple fields. The import tool fills out the following field values:

- clusterName - Contains the value provided by the atlas.metadata.namespace property in the application configuration file. The default value is cm.
- topic, name, description, URI - Contains the topic name.
- qualifiedName - Contains the topic name and the clusterName which is joined by '@'. For instance, my-topic@cm. This serves as the unique identifier for topics.
- partitionCount - Displays the number of partitions of the topic.

To set up the Atlas - Kafka import tool, follow these steps:

1. Select the Atlas service in Cloudera Manager.
2. Deploy client configs: Actions > Deploy Client Configuration.



Note: If kerberos authentication is enabled in Zookeeper, you must run the tool as the Kafka kerberos user and set up the Ranger policies for the Kafka user.

3. If Ranger is enabled, create a Ranger policy with the user running the tool in cm_atlas to allow the <user> to create, update, delete, and read Kafka entities:
 - Log into Ranger UI.
 - Navigate to cm_atlas policies.
 - Select the policy which has create_entity and delete_entity permissions.
 - Add the kerberos user that is used with the import-kafka.sh tool.
4. SSH into a host where the client configuration is deployed.
5. Run kinit as kafka if Kerberos is used to set up the Ticket Granting Ticket (TGT).
6. Set JAVA_HOME.



Note: Use export JAVA_HOME=/usr/java/default in this scenario.

7. Run the command `/opt/cloudera/parcels/CDH/lib/atlas/hook-bin/import-kafka.sh` to import Kafka topics into Atlas.