Cloudera Runtime 7.2.16

Atlas Lineage

Date published: 2019-09-23
Date modified: 2023-01-03

https://docs.cloudera.com/

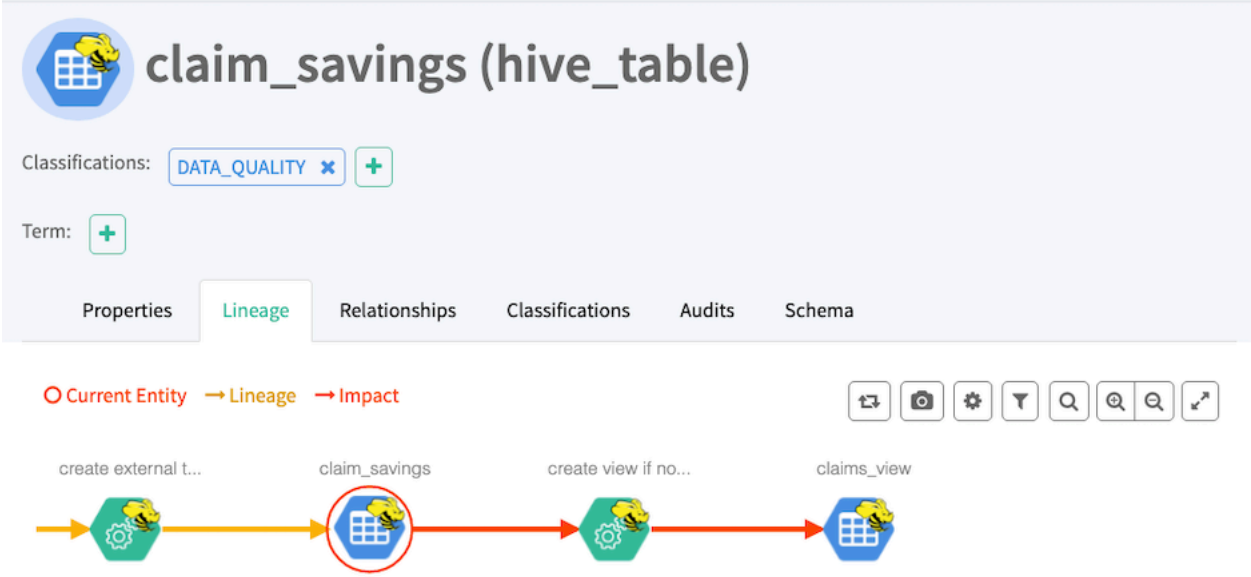
Legal Notice

Contents

Lineage overview

Atlas lineage helps you understand the source and impact of data and changes to data over time and across all your data.

Lineage information helps you understand the origin of data and the transformations it may have gone through before arriving in a file or table. In Atlas, if transformations occurred in services that provide process metadata, a lineage graph shows how data in a given column was generated. When a column appears in the output of a process, Atlas reads the content of the process and links the input column or columns to the output asset. This relationship is stored as a vertex in Atlas's graph database. It is displayed as a lineage graph in the details of each entity.



By default, Atlas can collect lineage information from the following sources:

- HiveServer
- Impala
- Spark

The lineage metadata produced by these sources may refer to additional asset entities. For example, when a Hive operation moves data from a Hive table to an HBase table or an HDFS file, Atlas includes an entity to represent the HBase table or HDFS file, and the entity is connected to the Hive table through lineage. The following sources may appear in lineage graphs when referenced:

- HBase
- HDFS
- S3

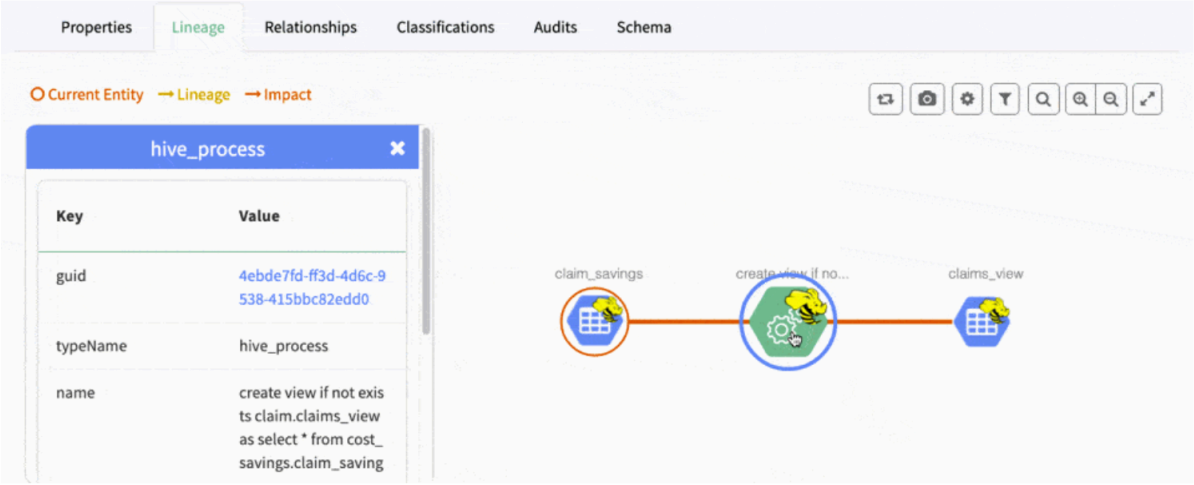
Data flow lineage from Cloudera Flow Management (NiFi) can be included as well by configuring the appropriate reporting task.

Viewing lineage

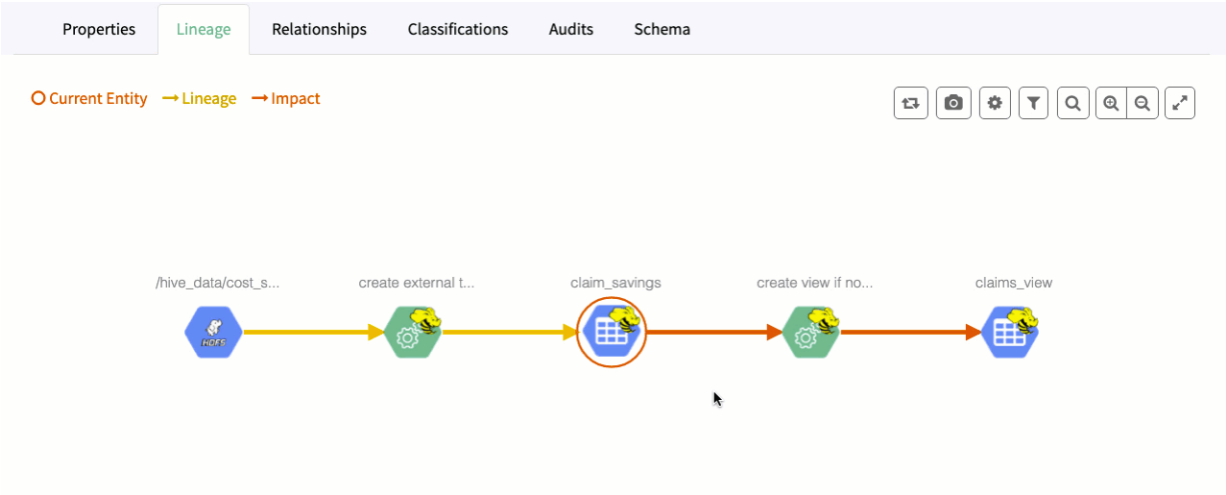
Atlas lineage graphs include lots of detail that you can reveal and configure.

Use your cursor to explore a lineage graph:

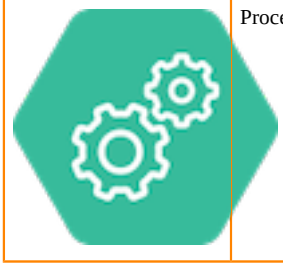
- Click to show details for an entity

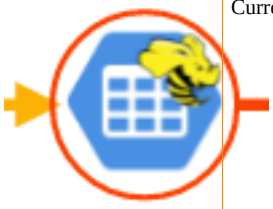
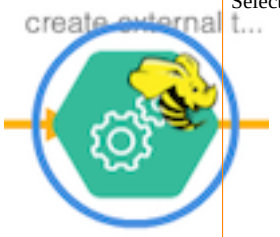


- Hover over an entity to show only one ancestor and descendant

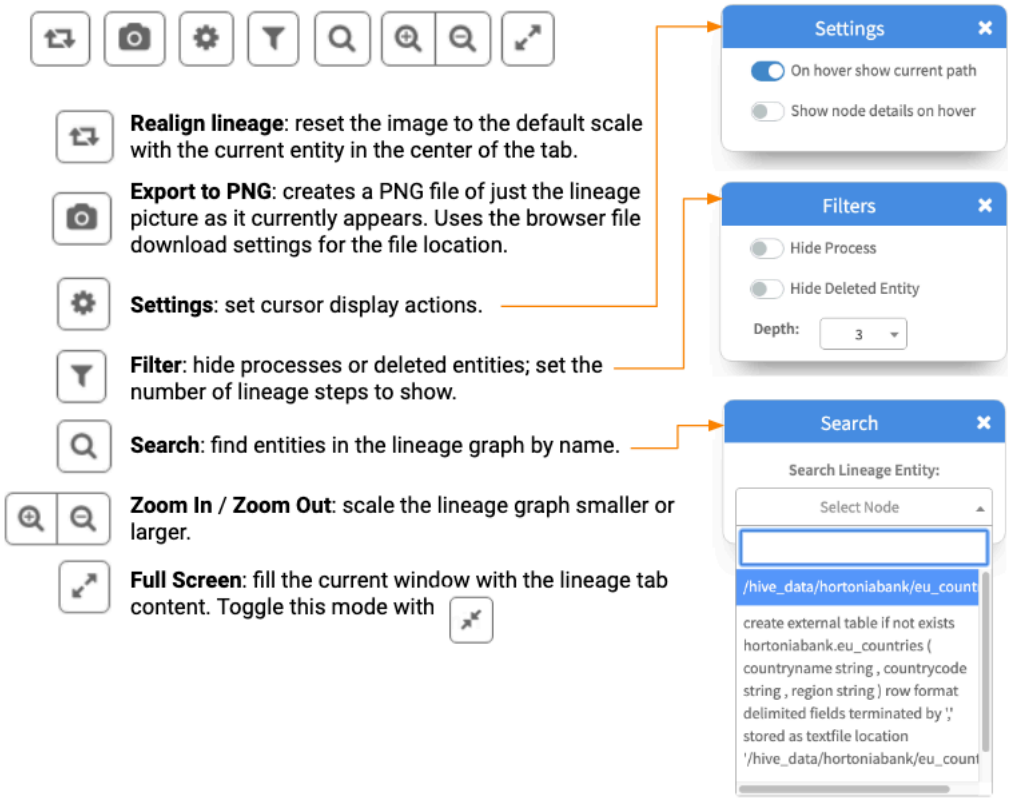


The following symbols can appear in lineage graphs:

Symbols	Name	Description and Actions
	Data Set entity	Represents a column, table, view, file, database, or other physical or logical data asset. While all data set entities are shown as a blue hexagon, the icon may vary based on the type of data asset and the source system.
	Process entity	Represents a query, job, or other process that applied to data assets. While all process entities are shown as a green hexagon, the icon may vary based on the type of process and the source system.

Symbols	Name	Description and Actions
	Current entity	A red circle indicates the current entity in the lineage graph.
	Selected entity	A blue circle indicates the entity selected in the lineage graph.
	Lineage	Connects the current entity with entities that provided input data. Entities connected with yellow lines are ancestors of the current entity.
	Impact	Connects the current entity with entities that could have an impact if data or schema information is changed in the current entity. Entities connected with red lines are descendants of the current entity.

The upper right corner of the lineage picture includes the following controls:



Related Information

Propagating classifications through lineage

Lineage lifecycle

Tables are dropped, schemas change, views are created: lineage tracks these changes over time.

Atlas reads the content of the metadata it collects to build relationships among data assets. When Atlas receives query information, it notes the input and output of the query at the column level: Atlas generates a lineage map that traces how data is used and transformed over time. This visualization of data transformations allows governance teams to quickly identify the source of data and to understand the impact of data changes.

Atlas processes contain lineage info; data assets by themselves do not. Impala queries are represented as processes and have lineage information; the data asset affected by Impala queries appear as Hive entities.

HDFS, S3, ADLS files appear when they are referenced by Hive, Impala, or Spark queries; operations that occur on the file system are not reflected in Atlas lineage.

The contents of a lineage graph are determined by what metadata is collected from services. If a process refers to a data asset but Atlas doesn't have an entity for that data asset, Atlas isn’t able to create an entity for the process and the lineage defined by that process won’t appear in Atlas.

Deleted data assets

Entities that represent data assets that have been deleted (such as after a DROP TABLE command in Hive) are marked as deleted. They show up in search results only if the checkbox to Show historical entities is checked. Deleted entities appear in lineage graph dimmed-out.

Historical entities are never automatically removed or archived from Atlas' metadata. If you find you need to remove specific deleted entities, you can purge specific entities by their GUIDs through REST API calls.

Temporary data assets

Sometimes operations include data assets that are created and then deleted as part of the operation (or as part of a series of operations that occur close together in time). Atlas collects metadata for these temporary objects. The technical metadata for the operation, such as query text, includes a reference to the temporary object; the object itself will show in the Atlas lineage diagrams.

For example, consider a Hive pipeline that writes data to a table, transforms the data and writes it to a second table, then removes the first table. The lineage graph shows the source file, the process that creates the first table, the first table, the process that transforms the data and loads it into the second table, and the second table. Atlas also collects the process where the first table is dropped. When you view the lineage graph, you can choose to show the first table or to exclude it by setting the filter option Hide Deleted Entity.

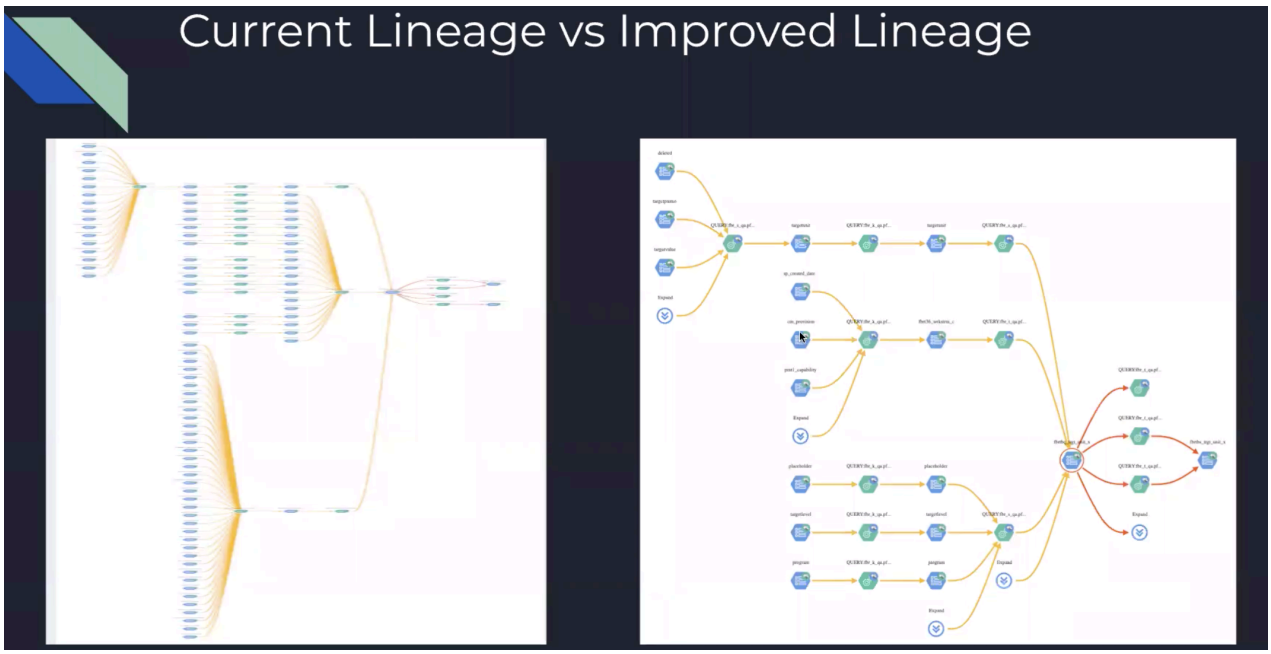
Support for On-Demand lineage

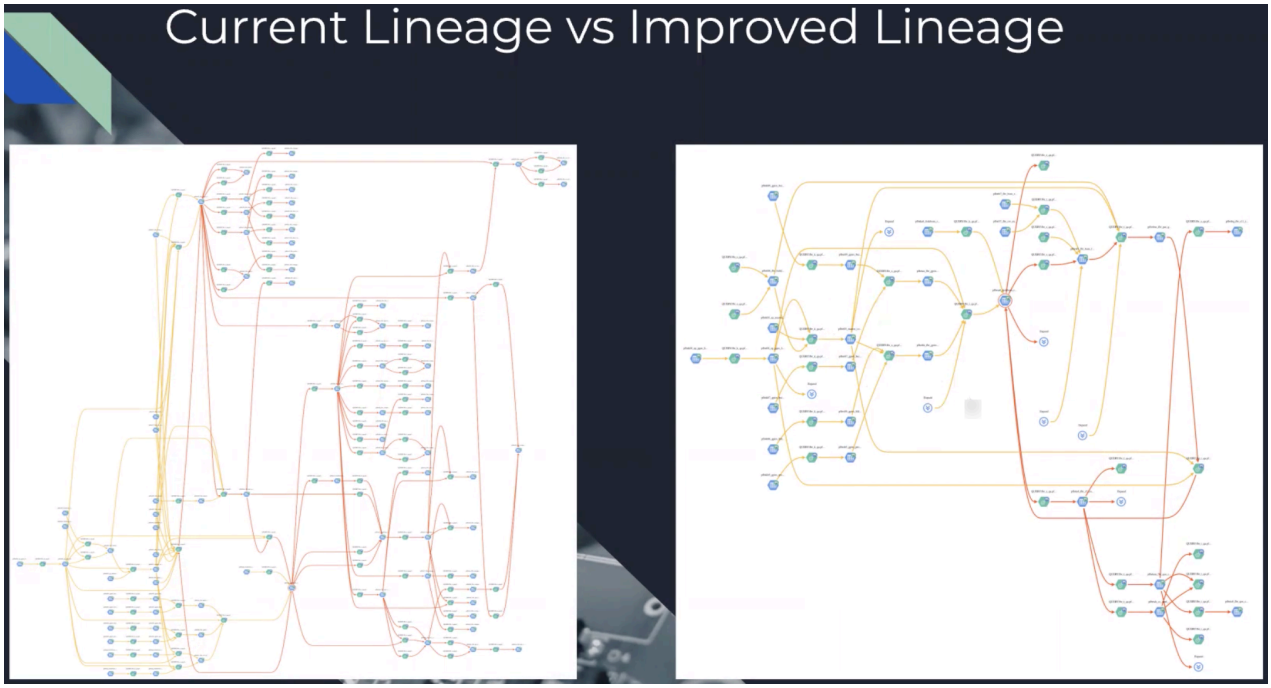
The on-demand lineage provides improved end user experience to handle data flow and related entities.

Currently, the lineage works with minimal basic data to return all the connected entities for any single request. The Atlas UI consumes additional time to load when the lineage response is larger than usual and also when the lineage is multi-level, the cost of processing the data could be higher.

To handle these conditions, the concept of on-demand lineage is introduced. Using the on-demand lineage, you can provide more control to the user for depth and height of specific lineage, apart from increasing the number of nodes from three (which is default with three input and output nodes respectively) and finally displaying the maximum number of nodes. On-demand lineage also processes limited lineage data that helps in the optimization of the UI.

In hindsight, the lineage data is broken down into parts and when the user clicks on the specific “expandable” button in the lineage, only then the lineage shows up fully (or as the lineage data is broken) to provide additional information.





Attention: The existing lineage setup is also supported but the user has to configure additional settings to bring up the on-demand lineage feature.

You can set the following parameters to use the on-demand lineage:

- atlas.lineage.on.demand.default.enabled = true
- atlas.lineage.on.demand.default.node.count = 3 (Default value is 3)
- atlas.lineage.max.node.count = 5k (Atlas supports upto 9k nodes)

Note the following:

- If you pass an inputRelationsLimit / outputRelationsLimit in the payload, the entered values are applicable only for that specific GUID, and for other nodes Atlas considers the default value as input/output Relations Limit.
- If you do not provide any limits or if you have 0 as the limit, Atlas considers default configurations.

- Order nodes in lineage are based on how the nodes are stored in JanusGraph.

The lineage data is retrieved from JanusGraph (retrieving the edges and vertices). The order in which the data availability is provided solely depends on the order JanusGraph stores and returns vertices. And the order of vertices in lineage is generally consistent all the time. In other words, every time a request is initiated for lineage data, the result is in the same order and the lineage on-demand feature is designed based on the consistent ordering as available in the JanusGraph database.

- You can update the node count value using the Settings > Nodes On Demand option in Atlas UI. This, apart from updating the same using Cloudera Manager.

HDFS lineage data extraction in Atlas

Atlas supports the HDFS lineage data extraction mechanism. Learn how it works, which HDFS shell script commands are supported, how to use these commands, and what the supported HDFS entities and hierarchies are.

Previously, HDFS data extraction ensured that only partial data was captured. Data stewards and platform owners track lineage of their ingest and ETL processes that depend on custom programs that convert copy and convert data when moving from one directory in HDFS to another.

The data descends into a staging directory in HDFS and later, periodically moves to an input directory in HDFS. Hive External tables are constructed on top of the input directory and Atlas does not support capturing HDFS locations without the Hive External table being created on top of the HDFS location(s). Typically, the move from staging to input directory is captured in Atlas lineage. The connectivity between staging and input directories does not generate the lineage automatically.

With the enhancement, lineage would automatically be captured and connected for the data ingest processes. This development allows the data architect and data steward to understand and govern data from ingest, ETL to consumption.

How HDFS lineage extraction works

The enhanced implementation consists of a wrapper, which is based around the HDFS commands that creates a new lineage for filesystem operations.

The lineage captures the directory / file specified in the scripting tool. The lineage also creates a parent-child relationship of directories / files and its contents upto the NameNode.

A filtering mechanism is also built to exclude or include paths and directories.

Currently, the following HDFS shell script commands are supported:

- -put and -copyFromLocal - Copy the contents from the local machine into HDFS.
- -cp - Copies the contents from one path to another HDFS path.
- -mv - Moves the contents from one path to another HDFS path.
- -rm - Removes the contents from HDFS.

Related Information

HDFS lineage commands

Prerequisites for HDFS lineage extraction

Before you start the extraction mechanism ensure you review the activities that need to be performed in Cloudera Manager.

You must enable the gateway role in Cloudera Manager. The following properties are visible in the Atlas configuration.

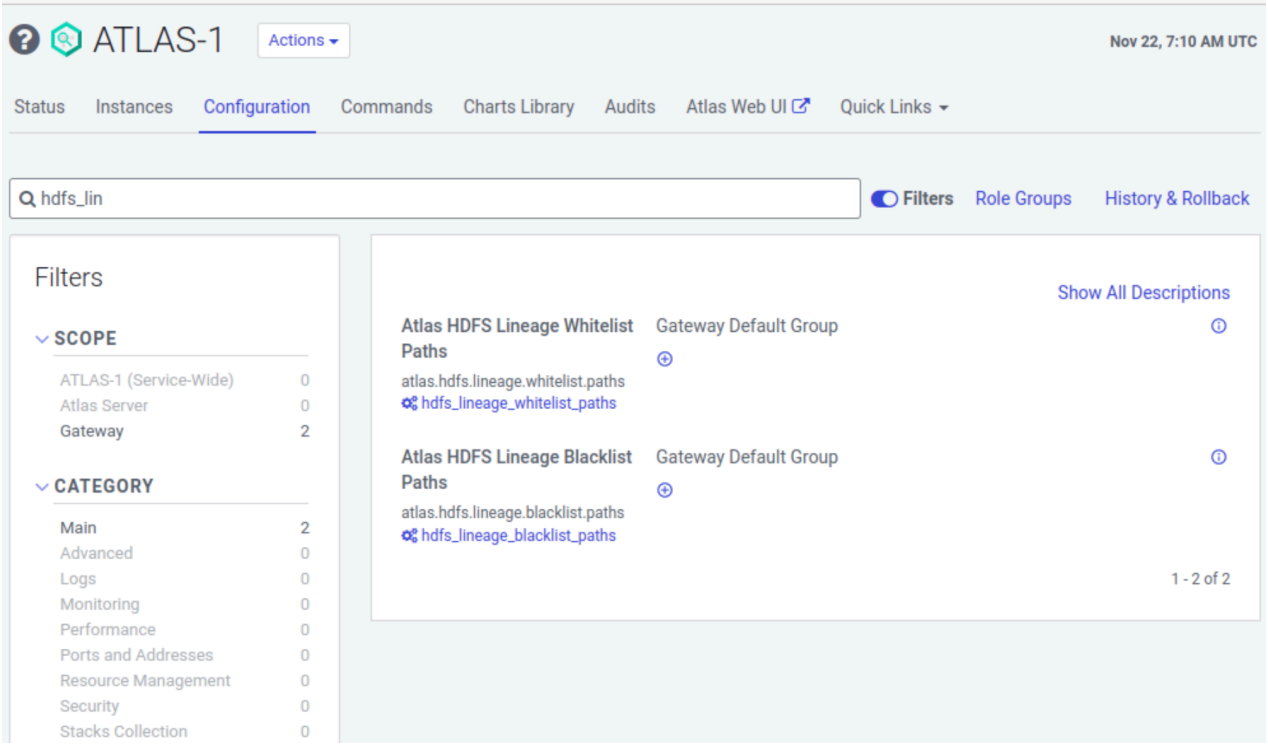
- atlas.hdfs.lineage.blacklist.paths

- atlas.hdfs.lineage.whitelist.paths

For example:

atlas.hdfs.lineage.blacklist.paths=/tmp/blacklist/./tmp/whitelist/blacklist/

atlas.hdfs.lineage.whitelist.paths=/tmp/whitelist/



Adding additional configuration

Before you commence the HDFS extraction, you must include the following properties in your Atlas setup to ensure that the HDFS entities reflect at the Atlas endpoint.

- Set the following to atlas-application.properties manually (NOT in Cloudera Manager) using the path: /etc/atlas/conf/atlas-application.properties

  Note: The user has to update the Atlas properties (/etc/atlas/conf/atlas-application.properties) in the Gateway node and run the extractor script from there. Any user who is allowed or permitted or has the required access to publish messages to Kafka can be used. For example: "atlas"

  - atlas.jaas.KafkaClient.option.keyTab
  - atlas.jaas.KafkaClient.option.principal

If you do not add these properties, the HDFS lineage entities do not reflect at the Atlas endpoint.

Attention: Entities created through Hook do not get consumed by Atlas and are specifically observed while running the HDFS Lineage script. Once the process to run the hdfs-lineage.sh script is completed, it is seen that in a few instances the entity is not created in Atlas. This scenario is observed on an intermittent basis and few entities are not viewed in Atlas. In the case where this issue is observed, the publishing of messages to Kafka topics consumes more than three seconds. This additional time consumption could be because of:

- Logging into Kerberos took more time to complete
- Connecting to Kafka topic took more than three seconds

To overcome this scenario, the Async message processing (atlas.notification.hook.asynchronous) must be disabled. You must manually set this flag in /etc/atlas/conf/atlas-application.properties to false

HDFS lineage commands

You must run the following commands to complete the HDFS lineage extraction into Atlas.

The name of the HDFS lineage capture script is called hdfs-lineage.sh. The extractor script is located in the extracto rs/bin directory in $ATLAS_HOME. The default location path is: /opt/cloudera/parcels/CDH/lib/atlas/extractors/ bin/hdfs-lineage.sh

Attention: Any HDFS lineage commands that are run outside of the script does not capture the lineage in Atlas.

As noted previously in earlier about the way HDFS lineage extraction is performed, the HDFS shell script commands are used to extract lineage in Atlas. The commands are run in the order of extraction.

Related Information

Prerequisites for HDFS lineage extraction

Running HDFS lineage commands

To support HDFS lineage extraction, you must execute the lineage shell script commands.

About this task

Running the HDFS commands:

Running the -put command:

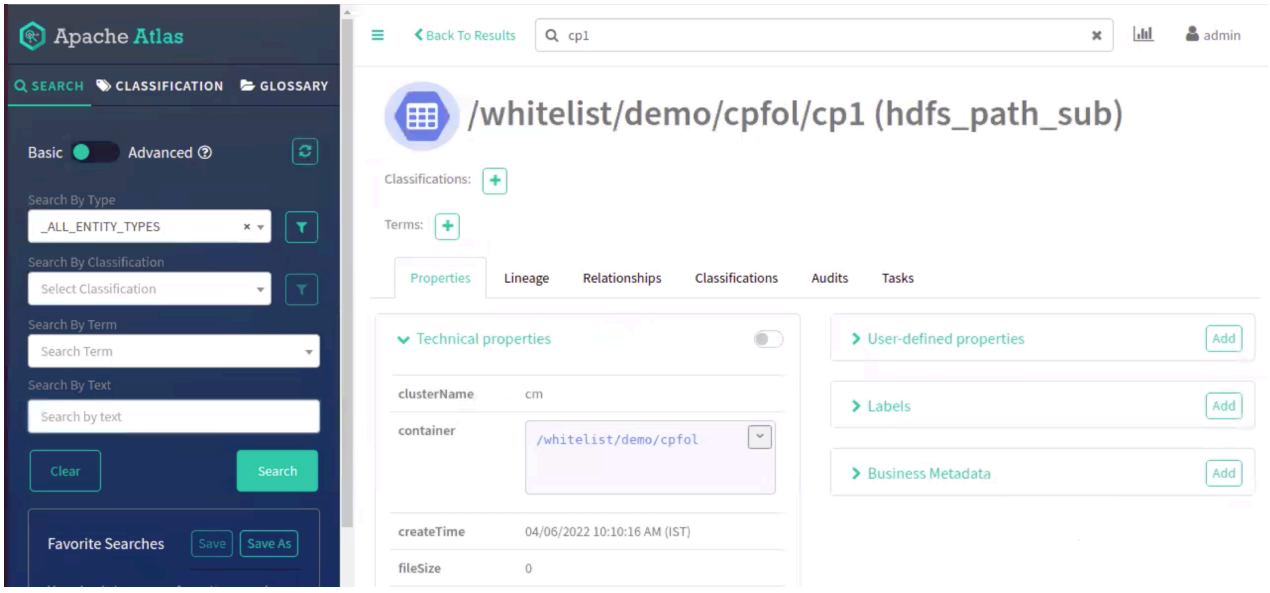
./hdfs-lineage.sh -put ~/csvfolder/ /tmp/whitelist/demo/cpfol/cp1

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

```
Hdfs dfs -put /root/csvfolder/ /tmp/whitelist/demo/cpfol/cp1 command was
          successful
```

HDFS Meta Data extracted successfully!!!

You must verify if the specific folder is moved to Atlas.



Running the -cp command:

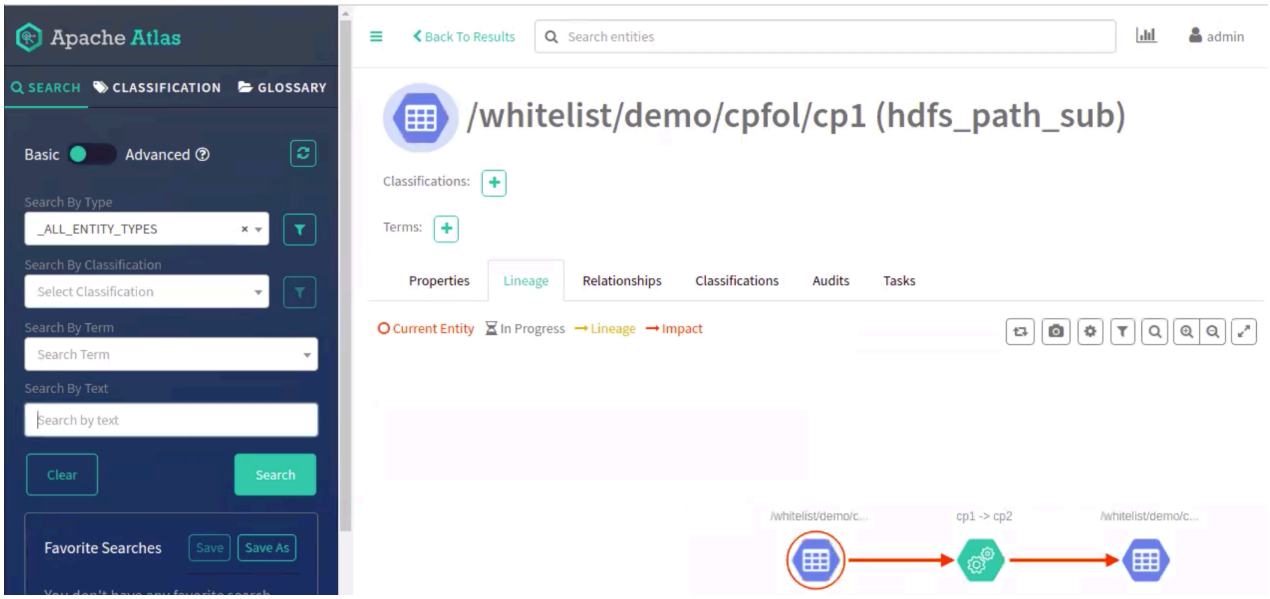
./hdfs-lineage.sh -cp /tmp/whitelist/demo/cpfol/cp1 /tmp/whitelist/demo/cpfol/cp2

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -cp /tmp/whitelist/demo/cpfol/cp1 /tmp/whitelist/demo/cpfol/cp2 command was successful

HDFS Meta Data extracted successfully!!!

You must verify if the contents of the folder is moved to another folder.



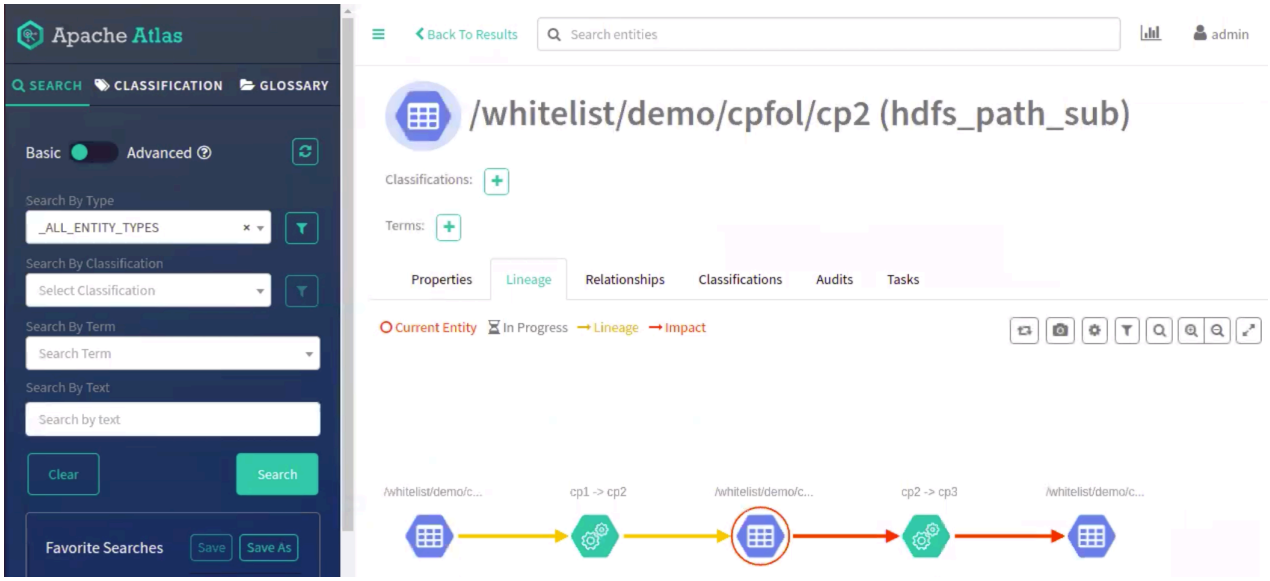
Similarly, you can move the contents of the copied folder to another folder.

./hdfs-lineage.sh -cp /tmp/whitelist/demo/cpfol/cp2 /tmp/whitelist/demo/cpfol/cp3

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -cp /tmp/whitelist/demo/cpfol/cp2 /tmp/whitelist/demo/cpfol/cp3 command was successful

HDFS Meta Data extracted successfully!!!



You must create an External Hive table on top of the HDFS location. For example, a Hive table on top of the HDFS location: cp3

create external table if not exists countries_table_11 (

CountryID int, CountryName string, Capital string, Population string)

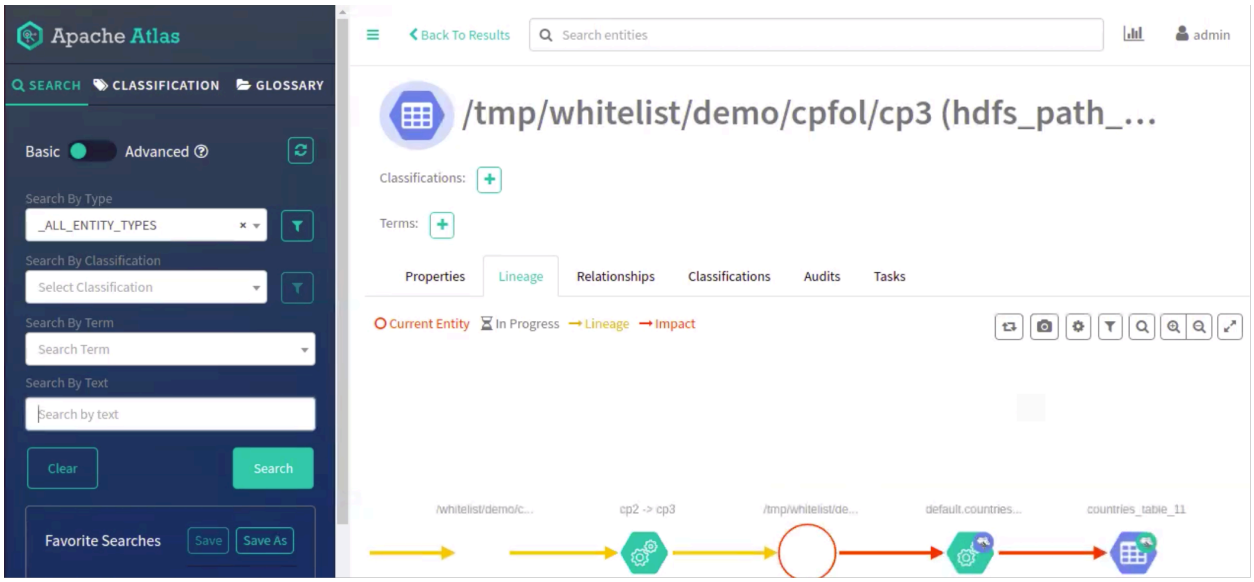
comment 'List of Countries'

row format delimited

fields terminated by ','

stored as textfile

location '/tmp/whitelist/demo/cpfol/cp3';



Running the mv command:

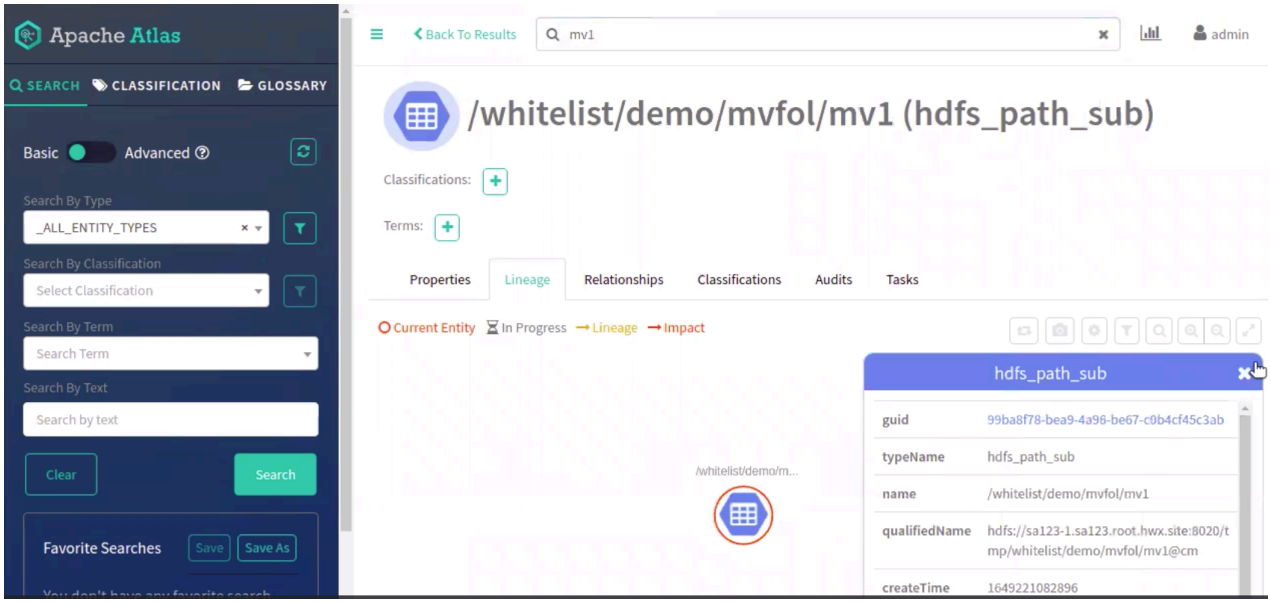
./hdfs-lineage.sh -put ~/csvfolder/ /tmp/whitelist/demo/mvfol/mv1

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

```
Hdfs dfs -put /root/csvfolder/ /tmp/whitelist/demo/mvfol/mv1 command was
          successful
```

HDFS Meta Data extracted successfully!!!

A single node is generated in the Atlas UI



You can move the contents of the folder mv1 to mv2

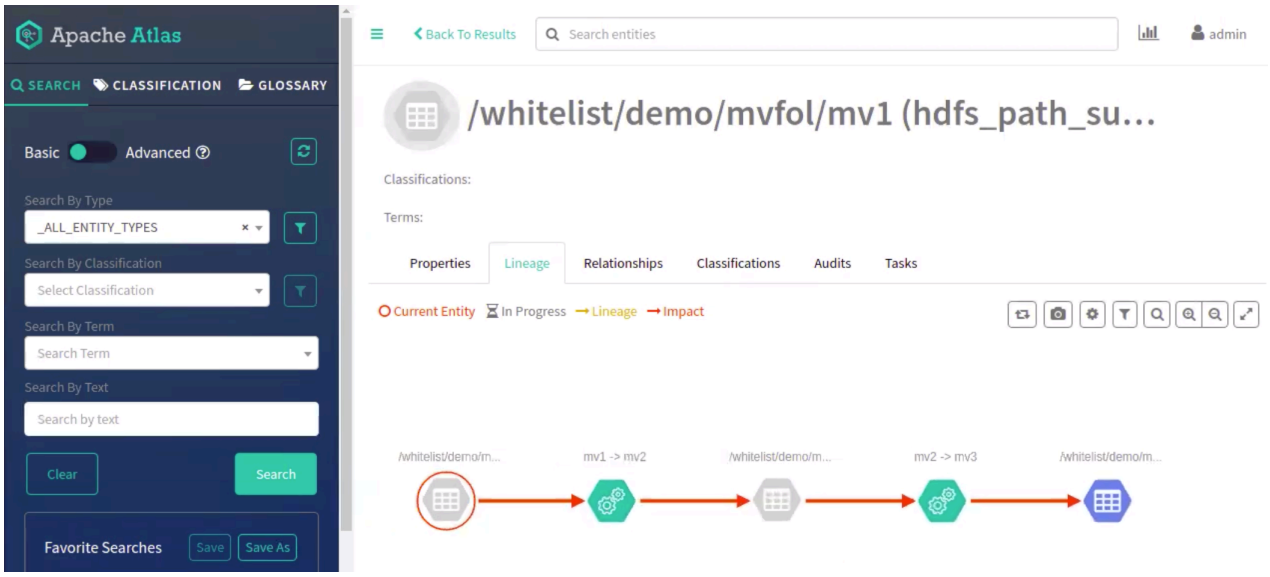
./hdfs-lineage.sh -mv /tmp/whitelist/demo/mvfol/mv1 /tmp/whitelist/demo/mvfol/mv2

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -mv /tmp/whitelist/demo/mvfol/mv1 /tmp/whitelist/demo/mvfol/mv2 command was successful

HDFS Meta Data extracted successfully!!!

In Atlas UI, you can view the contents of mv1 being moved to mv2 and the old entity is deleted (grayed out in the image) and likewise it continues for every operation where the previously created entity is deleted.



You must create an External Hive table on top of the HDFS location. For example, a Hive table on top of the HDFS location: mv3

create external table if not exists countries_table_12 (

CountryID int, CountryName string, Capital string, Population string)

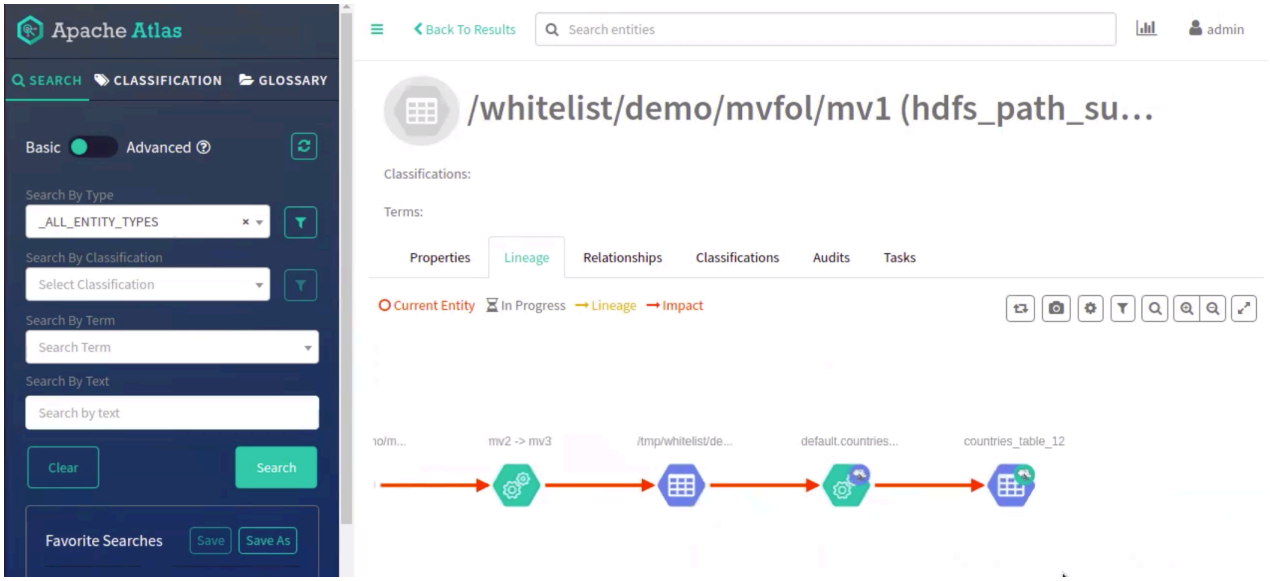
comment 'List of Countries'

row format delimited

fields terminated by ','

stored as textfile

location '/tmp/whitelist/demo/mvfol/mv3';



Running the rm command:

Create a sample file in HDFS

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/rmtest.txt

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -put /root/sample.txt /tmp/whitelist/rmtest.txt command was successful

HDFS Meta Data extracted successfully!!!

Use the rm command to delete the rmtest.txt file.

./hdfs-lineage.sh -rm -skipTrash /tmp/whitelist/rmtest.txt

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -rm -skipTrash /tmp/whitelist/rmtest.txt command was successful

HDFS Meta Data extracted successfully!!!

Note: If a folder contains multiple sub-folders and if the parent folder is deleted, all the contents of the parent folder are deleted.

Example

./hdfs-lineage.sh -rm -r -skipTrash /tmp/whitelist/demo/delfol

Related Information

Supported HDFS entities and their hierarchies

Inclusion and exclusion operation for HDFS files

The content or the files available in HDFS are either captured or ignored as lineage in Atlas. This scenario is achieved by including and excluding files and folders that Atlas checks for creating a lineage.

The hdfs.conf file contains the configuration for inclusion and exclusion options as described in Prerequisites for HDFS lineage extraction on page 9.

Some use cases where the including and excluding of HDFS files can be achieved are as follows

- When a simple file is created in the “whitelist” folder, a lineage is created in Atlas.

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/sample2.txt

- Similarly, when you try to create a file in the “blacklist” folder, Atlas does not create a lineage.

./hdfs-lineage.sh -put ~/sample.txt /tmp/blacklist/sample3.txt

- If a folder is not listed for any specific operation (Whitelist or Blacklist), Atlas does not capture the lineage.

./hdfs-lineage.sh -put ~/sample.txt /tmp/non/folder1/folder1file.txt

Note: Atlas does not capture the lineage even if the parent folder is included but the sub-folder / directories are excluded.

For example:

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/blacklist/blacklistfile.txt

- Atlas captures the lineage of the sub-folder if the parent directory or folder is included.

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/whitesub/whitelistsubfile.txt

- Atlas does not create a lineage to an excluded folder but a lineage is created to the same excluded folder if there is an included path available.

./hdfs-lineage.sh -put ~/sample.txt /tmp/blacklist/blToWl.txt

./hdfs-lineage.sh -cp /tmp/blacklist/blToWl.txt /tmp/whitelist/blToWl.txt

Atlas captures the lineage when folder(s) are included and later the lineage is captured from the excluded entity.

Important: The included folder takes precedence over the excluded folder, always.

The included folder takes precedence over the excluded folder, always.

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/wlToBl.txt

./hdfs-lineage.sh -cp /tmp/whitelist/wlToBl.txt /tmp/blacklist/wlToBl.txt

Supported HDFS entities and their hierarchies

When you perform basic search with the sub-type "hdfs_path" through Atlas UI or API, the search results include other HDFS types like "hdfs_path_v2", "hdfs_path_namenode", and other related entities.

The following HDFS lineage types are available when you perform the search using hdfs_path:

- hdfs_path_v2

  - hdfs_container

  - hdfs_contained

  - hdfs_path_namenode

  - hdfs_process

- hdfs_path_v2 is subtype of - hdfs_path - hdfs_container - hdfs_contained
- hdfs_path_namenode is subtype of - hdfs_path - hdfs_container

Note: Searching for the entity hdfs_path renders results for hdfs_path_v2 and hdfs_path_namenode. When the 'Exclude subtypes' option is selected, only the hdfs_path results are available.