DSS Administration

Date of Publish: 2018-05-16

http://docs.hortonworks.com

Contents

Managing Asset Collections

You can create, edit, and delete Asset Collections.

Create Asset Collections

You can group data assets into Asset Collections. This enables you to organize data based on business classifications, purpose, protection requirements, or more. Examples of Asset Collections are: customer profiles, sales assets, financials, PII, and HR data.

Procedure

1. From the **Asset Collection** page, click **Add Asset Collection**.

   The **Add** page appears.
2. Enter the following information.

Field Name	Description	Example Values
Name	Enter an appropriate Asset Collection name. This name cannot be duplicated across the system. (Mandatory)	Customer Profiles, Sales Assets, Financials
Description	Describe the purpose or intent of the Asset Collection. (Mandatory)	Contains customer profiles: data assets for US and WW.
Datalake	Assign the Asset Collection to one Datalake. Choose from a list of available Datalakes. (Mandatory)	dss_bbsh_clust3
Tags	Add tags to your asset collection for context and subsequent lookup. Tags enable your to quickly catalog, search and retrieve asset collections as well as share such information with others in the future. (Optional)	se, pii, geo, finance
Public/Private	Select public if you want other users to have access to this asset collection. Select private if only you want to have access to this asset collection. **Note:** You can later change the status of the asset collection. Click the lock icon on the Asset Collection Details page to change the access state of the asset collection.	

3. Click **Next**.

   The Asset Collection Details page appears for the new asset collection.
4. Click **Add Assets** to add related data assets into your asset collection.

   The **Asset Search** page appears.
5. Search for assets using Basic Search.
   a) Search using the name of the asset by entering the name in the search bar.
   b) Use filters to search for specific assets based on the attributes of assets. Click **Filter** to display the filters available.
      - Created Time: From the dropdown list, select the time to refine the search on the basis of when the asset has been created.

- Owner: Enter the name of the owner to refine the search on the basis of the owners of the assets.
- DB Name: Enter the name of the database.
- Tag: Enter the names of the tags.

c) Select one more than one filter if needed.
d) Click **Search** to view the assets. The Results appear.
e) Click **Reset** to reset the filters and search again.
f) From the list, click to select the assets that you like to add to your asset collection.

6. Search for assets using Advanced Search, if needed. Advanced search uses facets of technical and business metadata about the assets, such as those captured in Apache Atlas, to help users define and build collections of interest. Advanced search conditions are a subset of attributes for the Apache Atlas type hive_table.
7. Click **Done**.

   The assets are added to the asset collection and the Search page is refreshed.
8. Close the Search tab.

   The **Asset Collection Details** page appears.
9. Click **Save**.

Edit Asset Collections

You can edit asset collections by adding or removing assets and changing the access state of the asset collection.

Procedure

1. Click an asset collection in the list to edit it. The Details page of that Asset Collection appears.
2. On the Assets tab, click **Edit** to edit the content of this asset collection. The assets collection appears in edit mode. If another user is editing this asset collection, an error message will appear saying that this asset collection is being edited by another user and you cannot edit it.
3. Add or remove assets in the asset collection.
   a) Click **Add** to add new assets to this asset collection.
   b) Select one or more assets and click **Remove** to remove assets from this asset collection.
4. Click **Save** to save the changes that you made to the asset collection.
5. Click **Cancel** to undo any changes that you made to this asset collection.

Delete Asset Collections

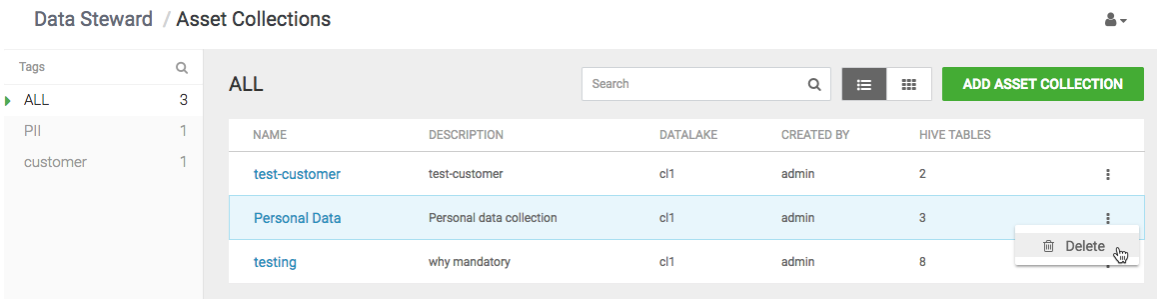
You might want to delete an Asset Collection if you no longer need to track those assets in that collection, or if you want to reassign those assets to another collection. You can delete Asset Collections at any time. Deleting an Asset Collection does not delete the assets contained therein, it only disassembles the collection of assets. You can re-create Asset Collections or reassign assets to new Asset Collections.

Procedure

1. From **Data Steward** > **Asset Collections** page, click the **More Options** icon

   ( )

   beside the name of the Asset Collection you want to delete.
2. Click **Delete**:



3. Click **Confirm**.
   You are returned to the **Asset Collections** home page.

Collaborate with other users

Data Stewards can collaborate and share insights with other users in the enterprise regarding various asset collections.

As a data steward, you can rate asset collections and view the average rating of an asset collection. This can help other users to find asset collections with higher ratings easily. You can also add your knowledge and insights about the asset collection by adding comments. Other users can respond to your comments or add their comments about each data asset collection.

On the right hand side of the asset collection page, you can see additional details about the asset collection. The collaboration details are also displayed in this tab. The tab displays the following details - average rating for the asset collection, the number of likes, the number of comments, and the bookmark icon indicating if the asset collection is bookmarked by the current user or not.

You can perform the following collaboration actions for each asset collection.

Like an asset collection

You can let other users know that you like an asset collection. The like icon on the asset collection page displays the total number of likes received by this asset collection.

Click the like icon to add the Asset Collection to your list of liked collections.

Comment and discuss about an asset collection

You might want to share your knowledge or insights about this asset collection with other users. Data Steward Studio allows you to collaborate with other users by adding comments.

Click the comment icon to add a comment about this asset collection. The Collaborate tab expands. Click **Actions** menu to reply to an existing comment. You can continue to add comments for each asset collection.

Bookmark the asset collection

In addition to sharing with other users, you can also bookmark asset collections for easy access in the future.

Click the bookmark icon to add the asset collection to your list of bookmarks. This asset collection with appear in the list of bookmarks when you click the Bookmarks link on the left navigation menu.

Rate the asset collection

You can also rate the asset collections on a scale of one to five. Click the star icon to rate the open asset collection. The Collaborate tab expands.

Click the stars to provide your own rating. The rating on the Asset Collections page shows the average of the rating provided by various users. The Rating section also displays the number of votes given for this asset collection.

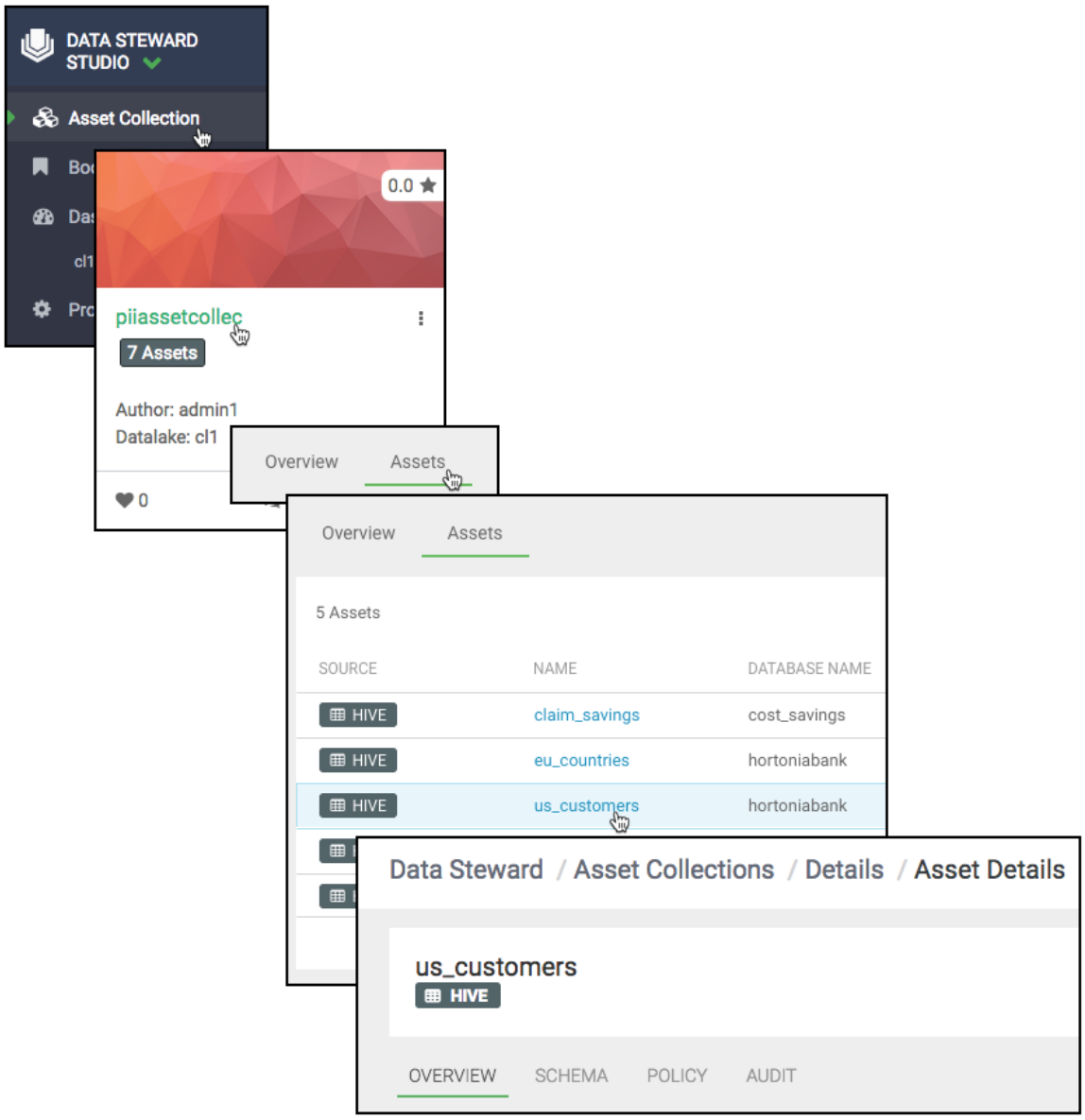
View the tags of an asset collection

You can add tags while creating the asset collection. You can also click on the tags to search for asset collections with similar tags. There are two types of tags. System tags are automatically generated based on the details of the assets in the asset collection. You can add more tags that appear in the list of user generated tags.

Viewing Data Asset Details (Asset 360)

The Asset 360 page comprises four tabs (Overview, Schema, Policy, and Audit). These tabs contain dashboards that provide an overview of your asset collection.

The Asset 360 page can be accessed from **Asset Collection** > **Select one asset collection** > **Assets** > **Select one data asset**. This brings you to the Overview tab, the first of the four tabs that form the Asset 360 page.



- Overview: Displays an overview for the data asset.
  - Table properties: Number of rows, number of columns, sensitive columns, number of partitions, owner, tags, profilers

    - Lineage: Shows the chain of custody for the data from relevant metadata repositories such as Apache Atlas. Lineage shows both upstream paths (lineage) into and downstream paths (impact) out of a given asset.
    - Users: Displays top 10 users for the data asset.
    - Access types: By action and operation type.
- Schema: Displays the schema of the data asset for structured data (such as Hive tables) from the relevant metadata repositories (such as Atlas). You can also view the shape or distribution characteristics of the columnar data within a schema based on the Hive column profiler.
- Policy: The policy view shows security (authorization) policies defined on assets such as those present in Apache Ranger. It includes both resource (physical asset based) as well as classification based policies
- Audit: The data asset audit logs page shows both most recent access audits from Apache Ranger and also summarized views of audits by type, user, and time window based on profiling of audit data.

View Data Asset Overview

Asset 360 > **Overview** displays all the Apache Atlas metadata associated with a particular data asset.

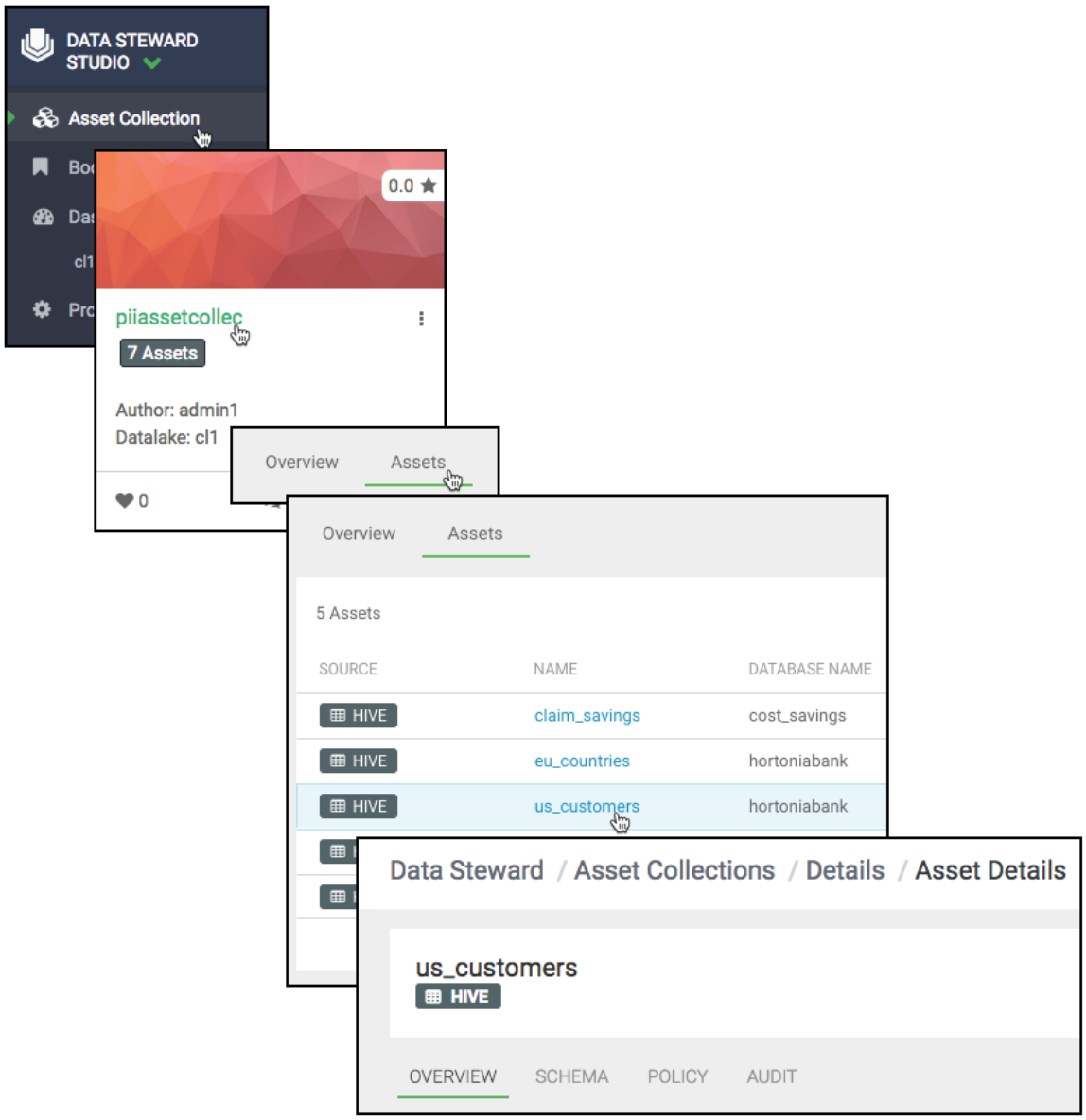
About this task

The Data Asset Overview page shows:

- Table properties: Number of rows, number of columns, sensitive columns, number of partitions, owner.
- System Tags: Displays tags associated with your asset to help with cataloging, searching, and retrieving.
- Profilers: Shows the status of profilers: active/inactive and time last run.
- Lineage: Shows the chain of custody for the data from relevant metadata repositories such as Apache Atlas. Lineage shows both upstream paths (lineage) into and downstream paths (impact) out of a given asset.
- Users: Displays top 10 users for the data asset.
- Access types: By action and operation type.

Procedure

From Data Steward, click: **Asset Collection** > **Select one asset collection** > **Assets** > **Select one data asset**:



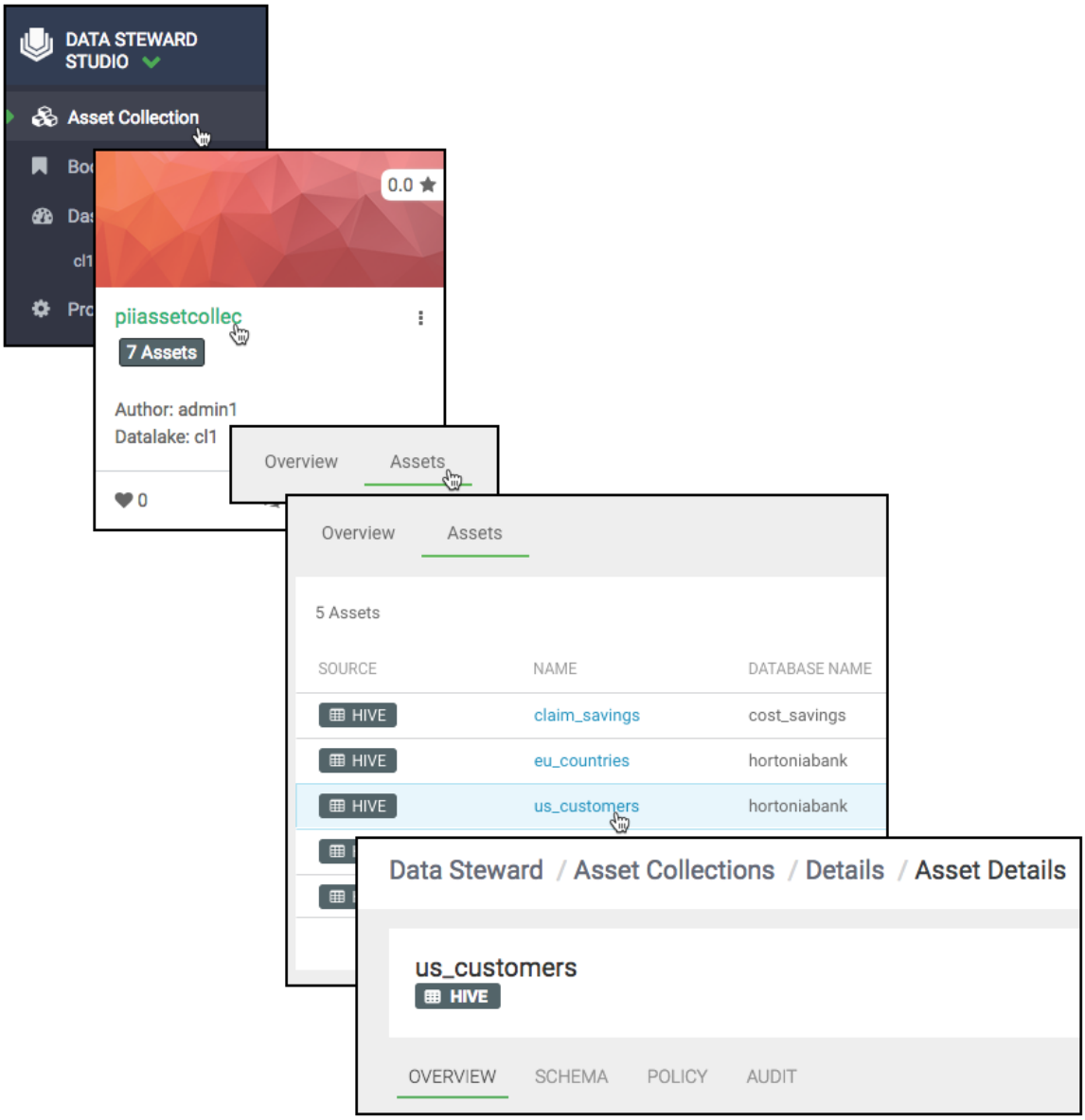
The Asset **Overview** window opens.

View Data Asset Schema

From **Asset 360** > **Schema**, you can view the schema of the data asset for structured data (such as Hive tables) from the relevant metadata repositories (such as Atlas).

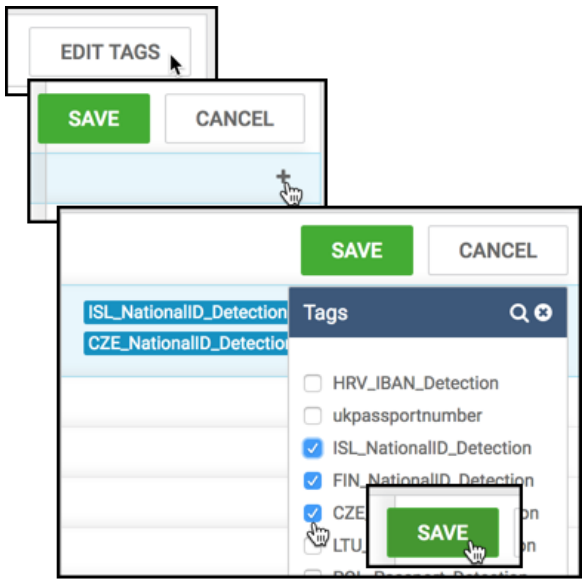
Procedure

1. From Data Steward, click: **Asset Collection** > **Select one asset collection** > **Assets** > **Select one data asset**.



The Asset **Overview** window opens.

2. Click **Schema**.
   The **Schema** table shows the data asset schema as retrieved from Apache Atlas.
3. (Optional) To edit tags:
   a) Click **Edit Tags**.
   b) Click the (+) icon.
   c) Select or deselect the tags you choose, then click **Save**.

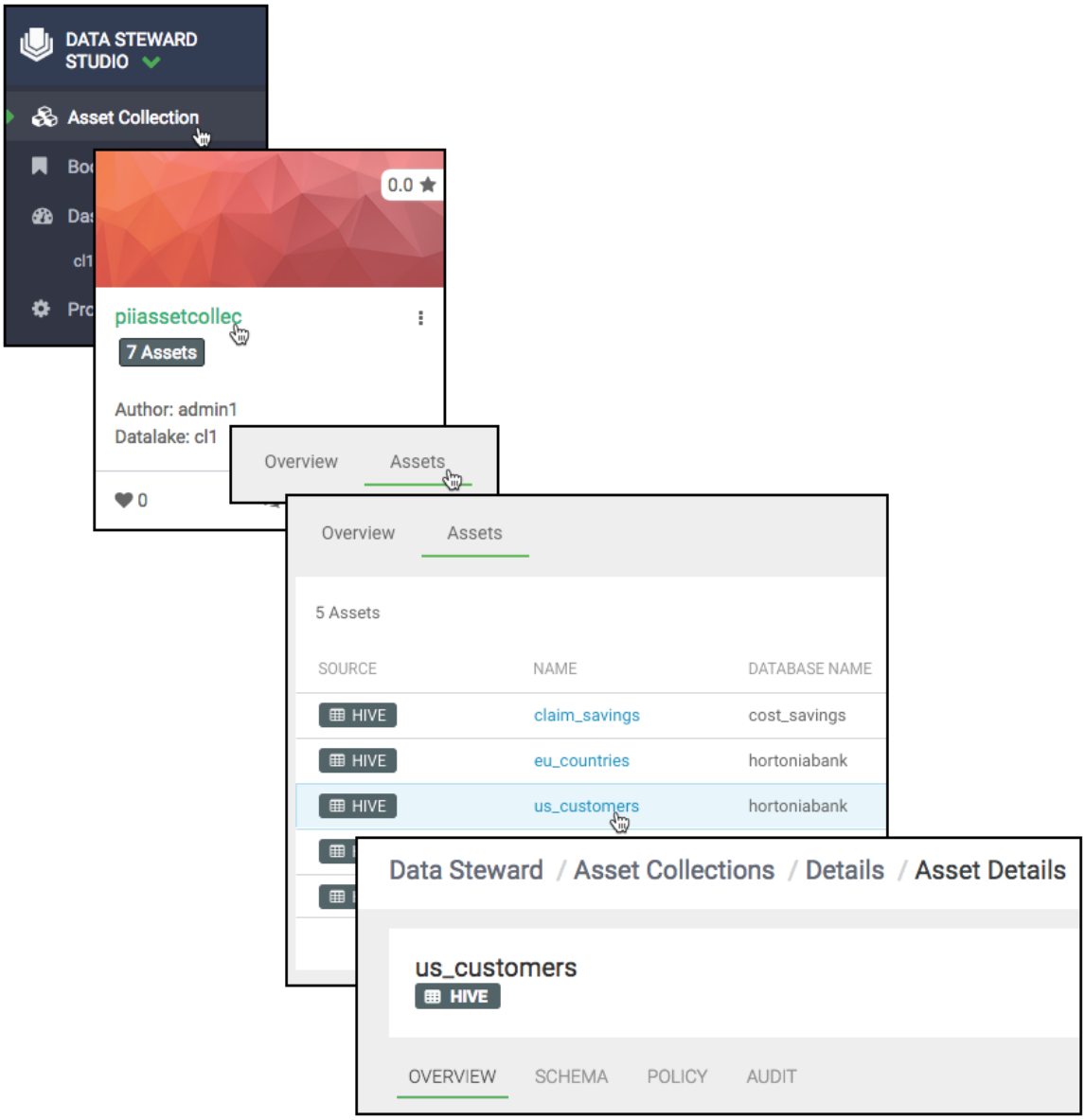


View Authorization Policies on a Data Asset

Asset 360 > **Policy** displays all the Apache Ranger policy details associated with a particular data asset. This helps you understand how data access is secured and protected: what users can see what data (or metadata) under what conditions (security policies, data protection, and anonymization).

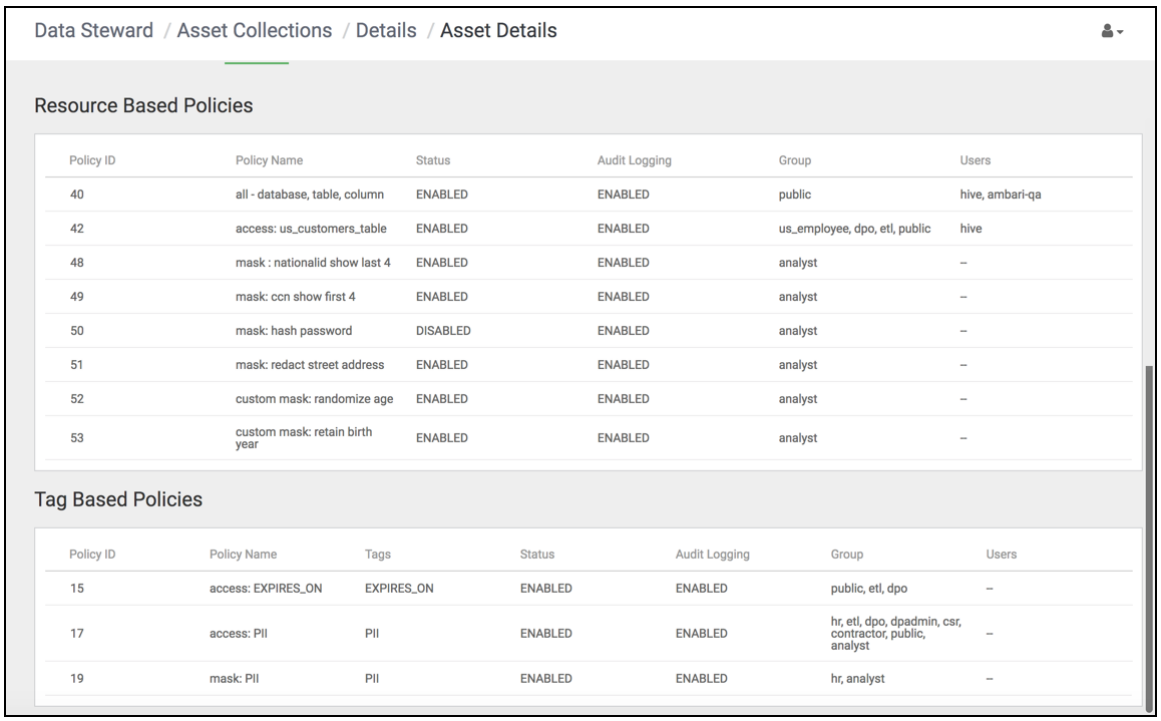
Procedure

1. From Data Steward, click: **Asset Collection** > **Select one asset collection** > **Assets** > **Select one data asset**:



The Asset **Overview** window opens.

2. Click the **Policy** tab.
   The **Policy** table shows the data asset policies as retrieved from Apache Ranger.

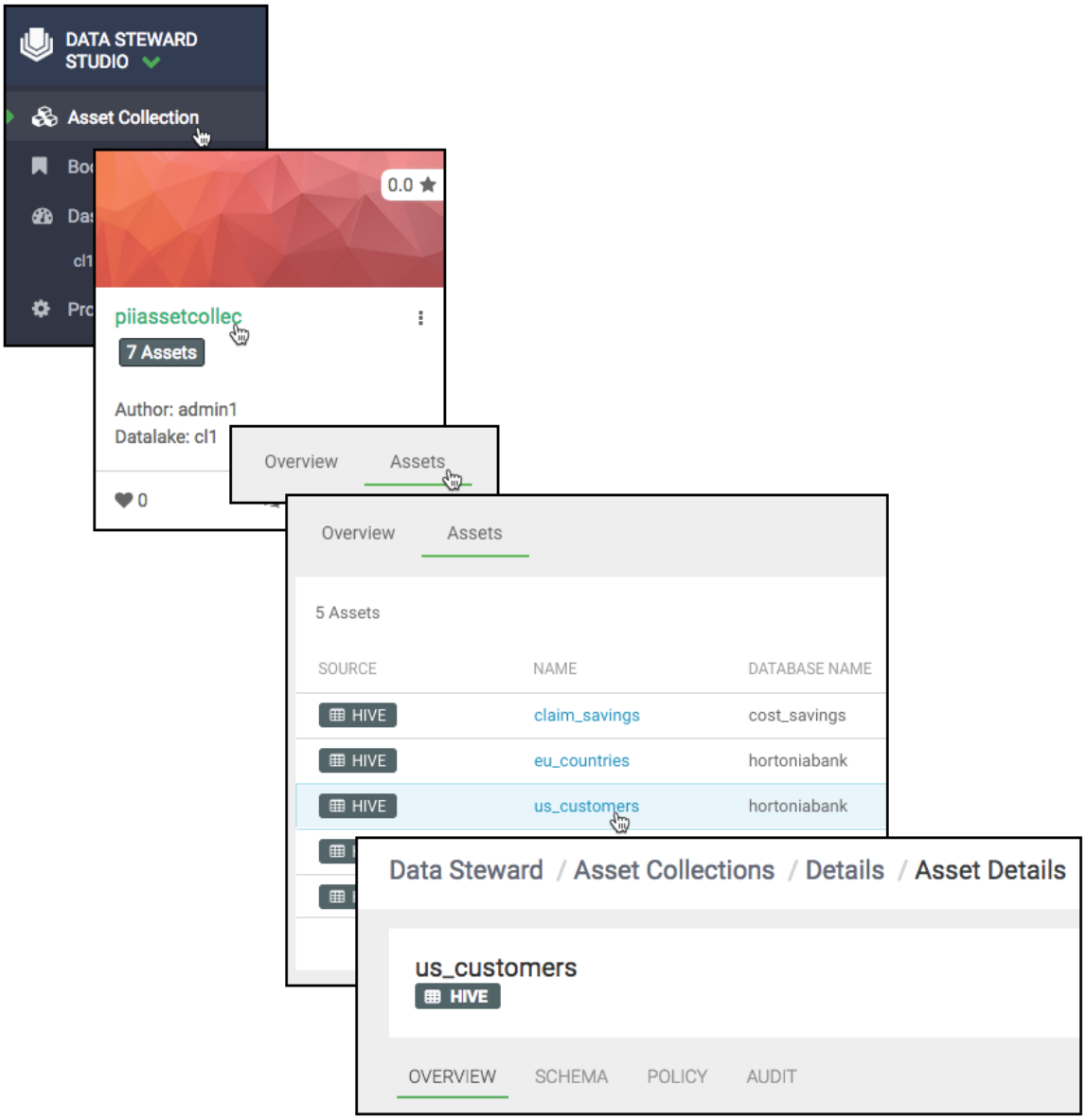


View Data Asset Audit Logs

Asset 360 > **Audit** displays all the Apache Ranger audit events associated with a particular data asset. This helps you to view who has accessed what data from a forensic audit or compliance perspective, and to visualize access patterns and identify anomalies.

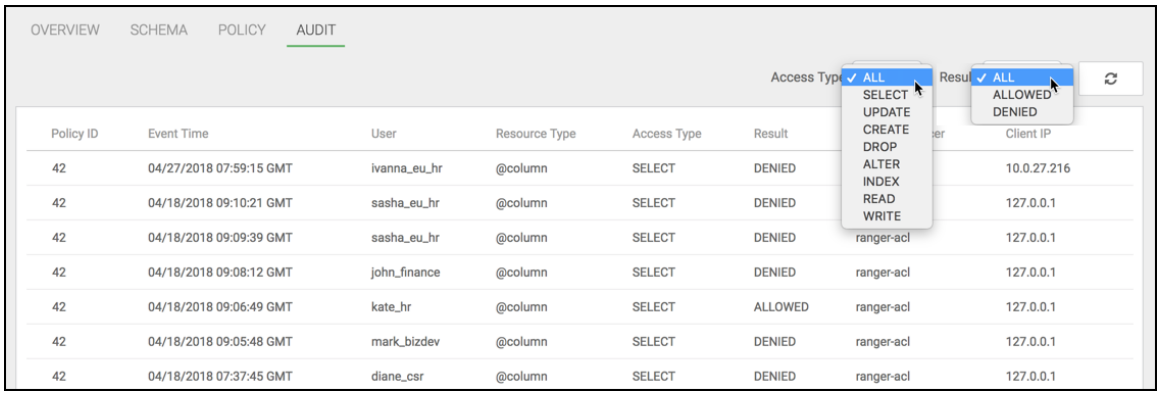
Procedure

1. From Data Steward, click: **Asset Collection** > **Select one asset collection** > **Assets** > **Select one data asset**:



The Asset **Overview** window opens.

2. Click the **Audit** tab.
   The Audit table shows the most recent raw audit event data as well as summarized views of audits by type of access and access outcome (authorized/unauthorized). Such summarized views are obtained by profiling audit records in the data lake with the audit profiler.



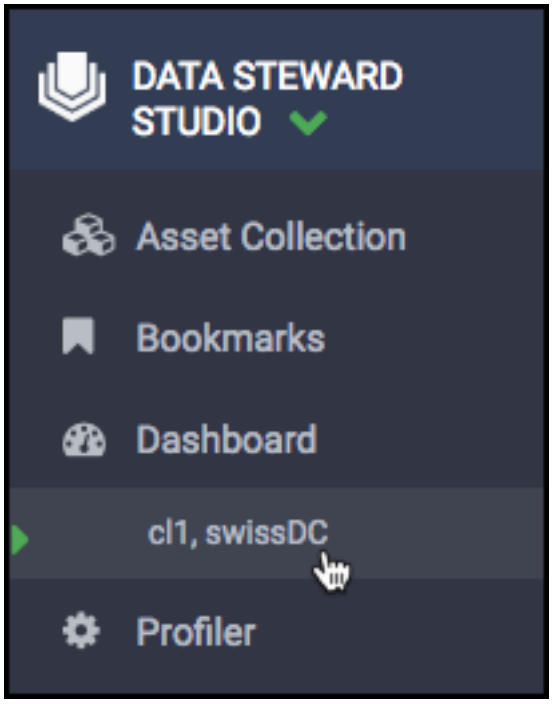
3. (Optional) You can filter the audit results by Access Type or Result.

Access type: SELECT, UPDATE, CREATE, DROP, ALTER, INDEX, READ, WRITE.

Result: ALLOWED, DENIED.

Viewing a Data Lake Dashboard

The Data Steward Studio Dashboard gives you an overview of your data lake's profiles and assets: Hive tables, execution, sensitivity, and access. This helps you understand asset profile coverage, access data, and asset sensitivity proportion (for example PII, PCI, and HIPAA), at a glance.



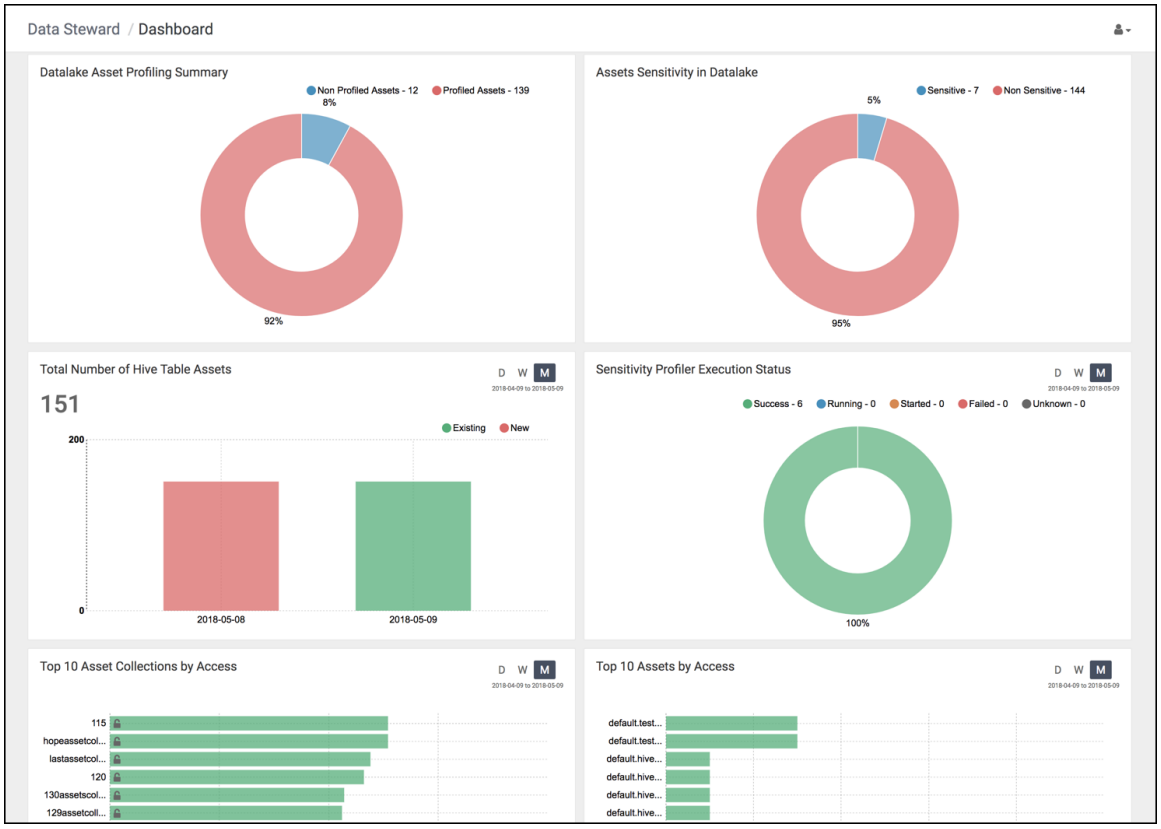


Table 1: Dashboard graphs

Graph Title	Description
Datalake Asset Profiling Summary	The number and percentage of assets covered by data profiling operations.
Asset Sensitivity in Datalake	The number and percentage of assets that are considered sensitive (for example, PII, PCI, and HIPAA). Based on a defined set of regular expressions, DSS runs a profiler job against Hive columns to determine whether values of the column satisfy the criteria for specific types of sensitive data and classify the columns accordingly.
Total Number of Hive Table Assets	Shows how your Hive table assets are growing over time.
Sensitivity Profiler Execution Status	This graph provides information about the monthly status of a particular profiler's execution: How many assets were run on that day, and how many completed successfully.
Top 10 Asset Collections by Access	Most accessed Asset Collections and how many times they were accessed.
Top 10 Assets by Access	Most accessed assets, who is accessing them, and how many times.

Related Information

Sensitive information types in Exchange 2016

Managing Profilers

The DSS profiler engine runs data profiling operations as a pipeline on data located in multiple data lakes. These profilers create metadata annotations that summarize the content and shape characteristics of the data assets.

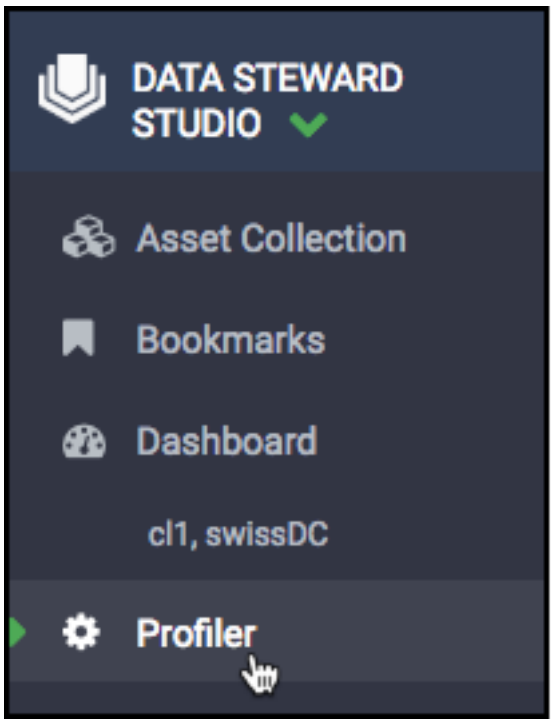


Table 2: List of built-in profilers

Name	Profiler	Description
Hive Column	tablestats hivecolumn	A Hive column univariate statistical profiler.
Hive Metastore	hive_metastore_profiler	Retrieves information about the number of hive tables that have been added every day.
Sensitive	sensitiveinfo	A sensitive data profiler- PII, PCI, HIPAA, etc.
Ranger Audit	audit	A Ranger audit log summarizer.

You can edit some of the profiler configurations in Ambari via the Datalake Profiler component. Currently, you can only use pre-built profilers. You can only schedule profilers during installation.

Related reference

Ambari Dataplane Profiler Configs

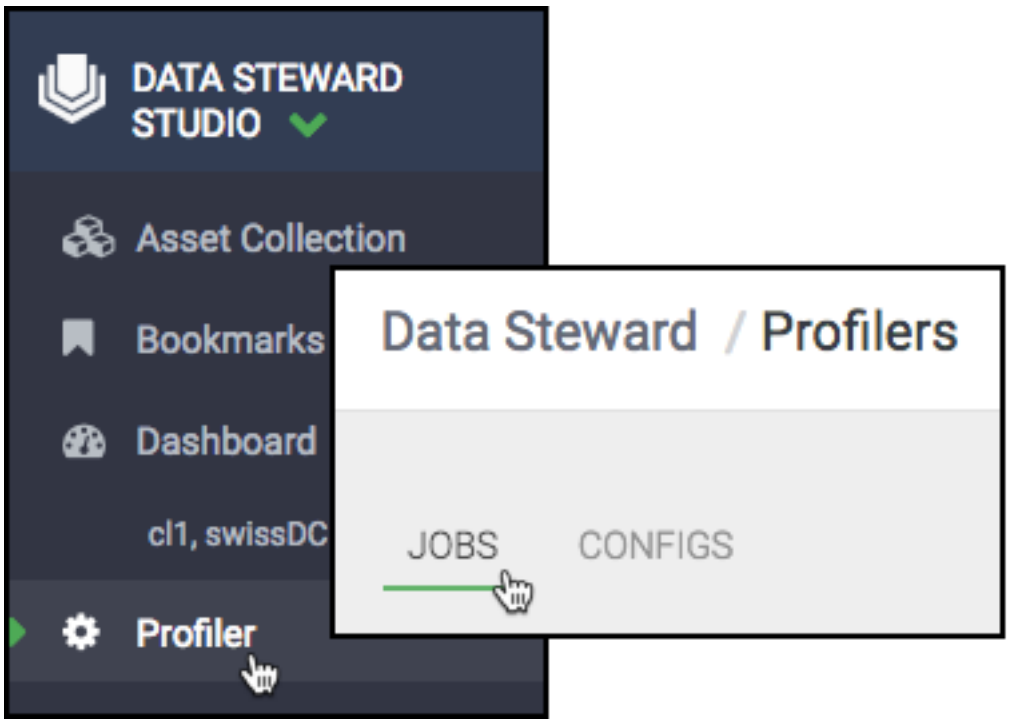
Viewing Profiler Jobs

You can monitor the overall health of your profiler jobs by viewing their status on the **Profiler** > **Jobs**.

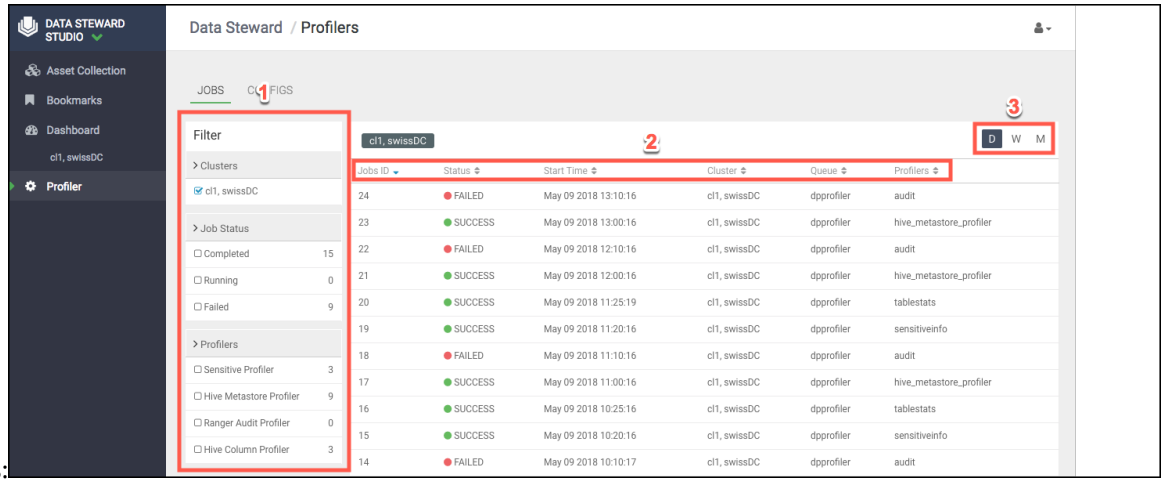
Each profiler, such as the Sensitive Profiler or Hive Column Profiler, runs a Spark job on a user-defined schedule in a user-defined queue. The queue is defined via the profiler configuration. You can view the status of each of those jobs for all your clusters.

Monitoring the profiler jobs has the following uses:

- By seeing long-term trends in job execution, you can determine the overall health of your profilers.
- If you do a data ingest, you can find out if the profiling has completed.
- Knowing when jobs first failed can help when troubleshooting problems with profilers.



You can take the following actions:



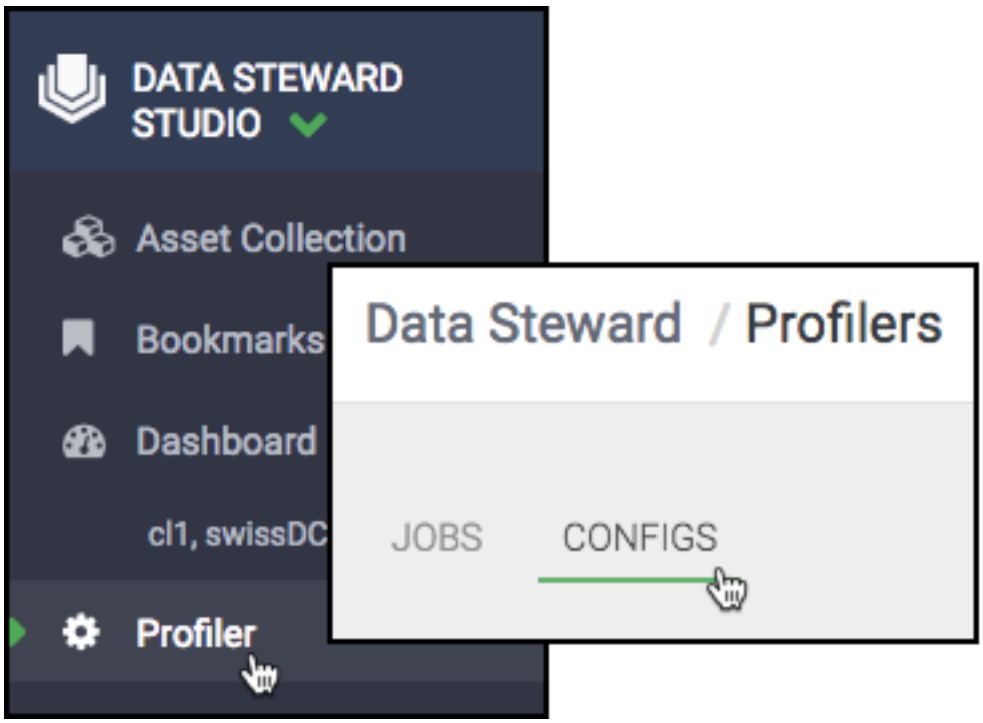
1. Filter by cluster, job status, or profiler.
2. Sort by jobs ID, status, start time, cluster, queue, or profilers.
3. Expand or narrow to show a day, week, or month of jobs.

Viewing Profiler Configurations

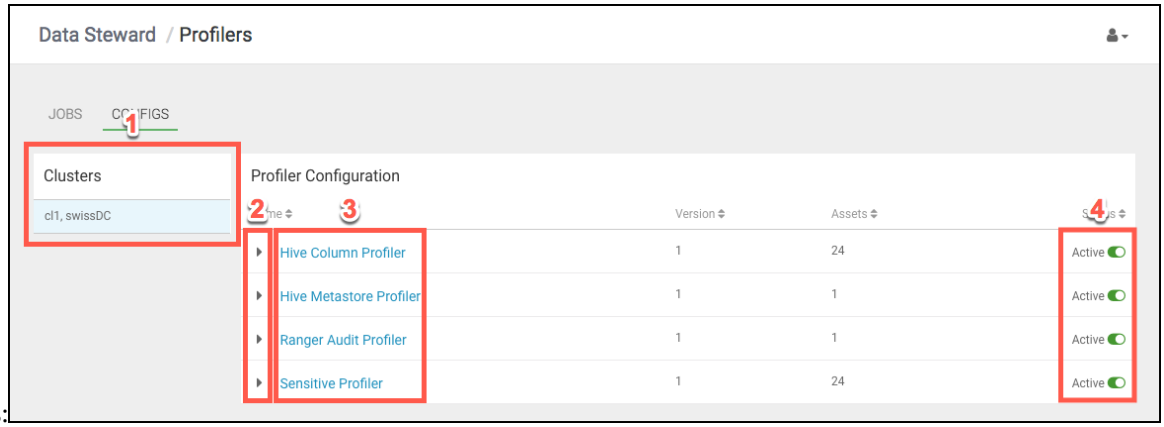
You can monitor the overall health of individual profilers by viewing their status on **Profiler** > **Configs**.

Monitoring the profiler configurations has the following uses:

- See which profilers are active and inactive.
- View asset coverage for a particular profiler over time- for instance, if you change a configuration for a profiler, you can see if new assets become covered.



You can take the following actions:



1. Filter by cluster.
2. Expand the execution status of an individual profiler. The percentage specifies how many assets have been profiled by this profiler on that day; the color denotes whether they were all successful, or not.
3. Edit the profiler configuration.
4. Toggle each profiler on/off.

Edit Profiler Configuration

In addition to turning on and off the profiler configurations, the individual profilers can be run with their own execution parameters. These parameters are for submission of the profiler job onto Spark. You can edit the configuration of profilers and update these parameters to run profiler jobs.

Procedure

1. Click **Profilers** in the main navigation menu on the left.
2. Click **Configs** to view all of configured profilers.

3. Select the cluster for which you need to edit profiler configuration.

   The list of profilers for the selected clusters is displayed.
4. Click the name of the profiler whose configuration you wish to edit.

   The Profiler Configuration tab is displayed in the right panel.
5. Select the queue from the drop down list.
6. Select a schedule to run the profiler. This is implemented as a cron job.
7. Update the advanced options.
   - **Number of Executors** - Enter the number of executors to launch for running this profiler.
   - **Executor Cores** - Enter the number of cores to be used for each executor.
   - **Executor Memory** - Enter the amount of memory in GB to be used per executor process.
   - **Driver Cores** - Enter the number of cores to be used for the driver process.
   - **Driver Memory** - Enter the memory to be used for the driver processes.
8. Toggle the state of the profiler from **Active** to **Inactive** as needed.
9. Click **Save** to apply the configuration changes to the selected profiler. The changes should appear in the profiler description.

Related Information

Submitting Spark Applications Through Livy

Replicate Profiler Configuration

If you have configured your profilers on a cluster and want to use the same configurations on other clusters, you can replicate the profiler configuration to the second cluster using the magic wand feature.

About this task

To replicate profiler configurations:

Procedure

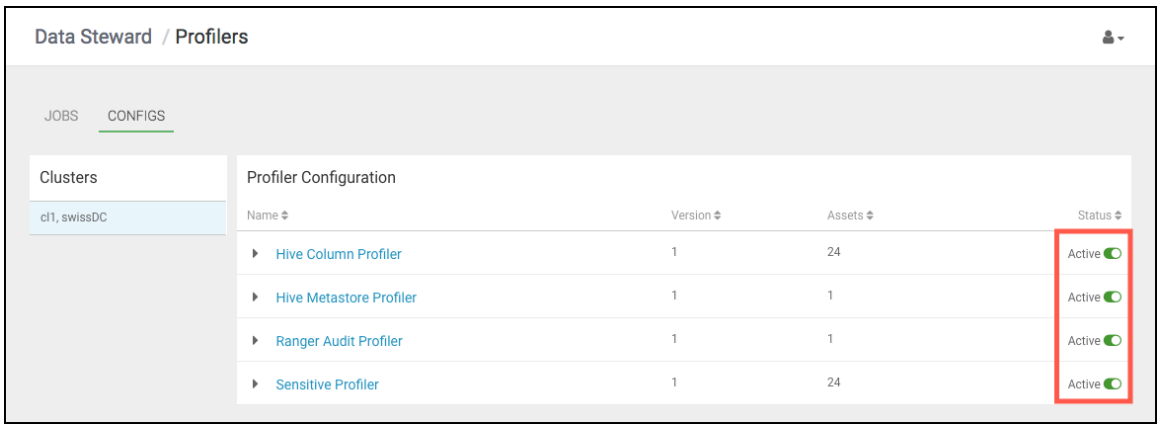
1. Select the cluster which has all the profiler configurations defined.
2. Click the magic wand icon on this cluster.
3. In the tab that appears, select the clusters to which you want to replicate the profiler configurations.
4. Select the parameters that you want to replicate on each cluster. You can choose to copy queue, schedule, or consider advanced Spark options.
5. Click **Apply**.

Enable or Disable Profilers

By default, profilers are enabled and run every 30 minutes. If you want to disable (or re-enable) a profiler, you can do this from the Configs tab.

Procedure

From **Profiler** > **Configs**, toggle the profiler **Active** or **Inactive**.



Custom Sensitivity Profilers

You can customize the Sensitive Data Profiler of Data Steward Studio to control how sensitive data is identified in a more flexible manner.

- You can customize the patterns that sensitivity profiler matches against.
- You can define a DSL to scan and identify patterns against a dictionary of values such as names, test credit cards from VISA and so forth.
- You can define a simple per-record algorithmic check.
- You can further combine the earlier granular capabilities into a composite capability.

Kraptr Framework

Using Kraptr framework, you can go through Hive tables, analyze data, and tag the columns for business, governance, or auditing purposes.

Data Steward Studio uses Kraptr framework for tag detection. To use this framework and customize the sensitive data profilers, perform the following steps:

1. Create Tag Schema files.
2. Create DSL files.

Step 1 Create Tag Schema files

Assign tags to columns when the rules match column names or the majority of values in the column. These labels also have associated metadata such as the percentage of values in a column that should match to tag the column with a label.

Define tags with the .kraptr_tag.json extension.

You must define the tags in one or more multiple tag schema files in the following path:

```
/apps/dpprofiler/profilers/sensitive_info_profiler/${profiler_version}/lib/
kraptr/tags/
```

Here is an example of a schema file. This schema defines two tags: luhns_demo and column_name_only_demo.

```
{
 "groupName": "demo",
 "tags": [
   {
     "tag": "luhns_demo",
     "nameWeight": 20,
     "valueWeight": 80,
     "isEnabled": true
```

```
    }
    {
      "tag": "column_name_only_demo",
      "nameWeight": 82,
      "valueWeight": 18,
      "isEnabled": true
    }
  ]
}
```

It is highly recommended to keep the groupName unique in the multiple schema files. This can be useful in modularizing different sets of tags.

1. tag - the label which will appear in DSS for this column. This label should be unique.
2. nameWeight - Percentage weight for column name. In case of values like age where identifying label based on value is extremely difficult, you can define a DSL for identifying column names close to age and its synonyms and keep the nameWeight value high to make sure that the column will be tagged when the name comes.
3. valueWeight - Percentage weight for column value. The match percentage among all rows in table is considered. Actual percentage attributed to the column for a tag will be computed as follows.
   a. actualValueWeight =valueWeight * number_of_rows_matched/ total_number_of_rows_in_sample
4. isEnabled - If this is set to false, this tag will not be identified.

Step 2 Create DSL files

To create the DSL files, you combine the different behaviors and use various operators available in Kraptr Grammar.

You can create a schema to map columns matching these DSLs to specific tags defined in the previous step.

Define DSLs with the .kraptr_dsl.json extension and place all the DSL schema files in the following path:

```
/apps/dpprofiler/profilers/sensitive_info_profiler/${profiler_version}/lib/
kraptr/dsl/sensitiveinfo/
```

Here is an example of sample schema with four DSL definitions.

```
{
 "groupName": "demo",
 "profilerInstanceName": "sensitivityinstance",
 "dsls": [
   {
     "matchType": "value",
     "dsl": "luhn_check",
     "tags": [
       "luhns_demo"
     ],
     "isEnabled": true
   },
   {
     "matchType": "value",
     "dsl": "regex(\"\\\\b(([a-zA-Z0-9_\\\\-\\\\.]+)@((\\\\[[0-9]{1,3}\\\\.
[0-9]{1,3}\\\\.[0-9]{1,3}\\\\.)|(([a-zA-Z0-9\\\\-]+\\\\.)+))([a-zA-Z]{2,4}|
[0-9]{1,3})(\\\\]?))\\\\b\")",
     "tags": [
       "combined_demo"
     ],
     "isEnabled": true
   },
   {
     "matchType": "value",
```

```
     "dsl": "luhn_check or whitelist(\"/apps/dpprofiler/profilers/
sensitive_info_profiler/1.0.0.1.2.1.0-16/meta/whitelist\")",
     "tags": [
       "combined_demo"
     ],
     "isEnabled": true
   },
   {
     "matchType": "name",
     "dsl": "is_in_ci(\"id\")",
     "tags": [
       "column_name_only_demo"
     ],
     "isEnabled": true
   }
 ],
 "isEnabled": true
}
```

The dsls section defines the different DSLs and the tag that they are mapped to.

1. matchType - states whether this DSL has to be matched on column name (ideal for cases like age where identification using value is difficult) or column value. To apply DSL on column name and value, you must set this to a value and name it accordingly.
2. dsl - defines the DSL expression which will get applied on column name or value based on matchType.
3. Tags - list of tags (defined in Tag Schema) to apply if DSL expression succeeds on this column name or value. Note that the DSL validation success on one column value is not enough to tag the column. Kraptr also considers the percentages in tag schema, column name matches if any and the percentage of rows matched in total rows to compute a final percentage. If this final percentage is more than a threshold (now fixed at 70 for system), the column is tagged.
4. isEnabled - If this is set to false, this DSL will not be applied.

Make use of Behaviors

You can use various behaviors to take single inputs of type text and evaluate them to a Boolean value.

The profiler can take column values of any type and pass the values to each behaviour as text. Behaviors include the following:

1. Regular expressions
2. Inline whitelist and blacklist checks
3. File based whitelist and blacklist checks
4. Luhn algorithm

Regular expressions

You can include one or more regular expressions and evaluate to True if one of these matches the provided value.

Keyword: regex

A regex that matches everything can be defined as follows:

```
regex(\"[\\\\s\\\\S]+\")
```

A regex that includes multiple expressions can be defined as follows:

```
regex(\"[\\\\s\\\\S]+\",\"^[0-9]*$\")
```

Creating Regular Expressions in Kraptr Framework

Make sure to modify the regular expressions for the Kraptr framework.

An example regular expression appears as follows in its pure form:

```
regex("(\s|,|;|\A)([A-Za-z]{2}\d{6})(\s|,|;|\Z)")
```

For the Kraptr framework to understand this regular expression, perform the following steps:

1. Escape the double quotes enclosing the attribute.
2. Escape the attributes twice.

The final behavior appears as follows:

```
regex(\"(\\\\s|,|;|\\\\A)([A-Za-z]{2}\\\\d{6})(\\\\s|,|;|\\\\Z)\")
```

In the current version, Kraptr does not support custom behaviors. You can use these in-built behaviors for your business purposes.

Inline whitelist and blacklist checks

Use the inline whitelist and blacklist checks to validate if the value is present in the list of strings you passed as attributes.

Keywords: is_in, is_in_ci, is_not_in, is_not_in_ci

Make sure the number of entries for whitelist and blacklist should not be more than 20 to save readability of DSL.

- is_in and is_in_ci checks if the value is present in the given attributes (whitelist).
- is_not_in and is_not_in_ci checks if the value is absent in the given attributes (blacklist).

is_in and is_not_in are case sensitive whereas is_in_ci and is_not_in_ci are case insensitive.

For example:

```
is_in(\"ssn\",\"ssnnumber\",\"ssnno\")
```

```
is_in_ci(\"ssn\",\"ssnNumber\",\"ssnNo\")
```

```
is_not_in(\"ssn\",\"ssnnumber\",\"ssnno\")
```

```
is_not_in_ci(\"ssn\",\"ssnNumber\",\"ssnNo\")
```

File based blacklist and whitelist checks

When the number of whitelist and blacklist entries are many and cannot be defined inline, you can create whitelist or blacklist files with one value in each line.

Keywords: whitelist, blacklist

Make sure to place the file in an HDFS location that is accessible to the Profiler Agent user.

You can provide the location of the file as an attribute in the DSL definition.

For example:

```
whitelist(\"/apps/dpprofiler/profilers/sensitive_info_profiler/1.0/lib/
kraptr/meta/whitelist\")
```

```
blacklist(\"/apps/dpprofiler/profilers/sensitive_info_profiler/1.0/lib/
kraptr/meta/blacklist\")
```

Luhn algorithm

You can do a Luhn check on identification numbers in columns.

Keyword: luhn_check

Use the Luhn algorithm or Luhn formula to validate a variety of identification numbers such as credit card number, IMEI numbers, National Provider Identifier numbers in the United States, Canadian Social Insurance numbers, Israel ID Numbers, and Greek Social Security Numbers.

IBAN validation

You can also do IBAN validation.

Keyword: iban_check

The International Bank Account Number (IBAN) is an internationally agreed system of identifying bank accounts across national borders to facilitate the communication and processing of cross border transactions with a reduced risk of transcription errors.

An IBAN is validated by converting it into an integer and performing a basic mod-97 operation (as described in ISO 7064) on it. If the IBAN is valid, the remainder equals 1. This algorithm does that validation which adds an extra layer of security.

Make use of DSL Grammar

Using DSL grammar, you can combine different behaviours in intuitive ways to bring out a functionality.

The two dummy behaviours available in this framework are as follows:

1. falseIdentity - Always evaluates to false, regardless of the input.
2. trueIdentity - Always evaluates to true, regardless of the input.

These two behaviors are used in the following examples and descriptions.

Binary AND operator

Keyword: and

And works the same way it does it other languages. Hence following observations.

```
falseIdentity and trueIdentity == falseIdentity
```

```
falseIdentity and falseIdentity == falseIdentity
```

```
trueIdentity and trueIdentity == trueIdentity
```

```
trueIdentity and falseIdentity == falseIdentity
```

Here we are using == to show their equality.

Value declaration operator

Keywords: val

Symbols: =

Value declaration lets you modularize things to make your DSL. A DSL only using the and operator and val operator appears as follows:

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
rule1 and rule2
```

The above expression evaluates to false.

Binary OR operator

The or operator works the same way it does it other languages.

```
falseIdentity or trueIdentity == trueIdentity
```

```
falseIdentity or falseIdentity == falseIdentity
```

```
trueIdentity or trueIdentity == trueIdentity
```

```
trueIdentity or falseIdentity == trueIdentity
```

Let's expand our DSL to use or as follows.

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
val rule3=rule1 and rule2
```

```
rule3 or trueIdentity
```

The above expression evaluates to true.

Unary Not operator

not negates value of a behaviour (from true to false and vice versa). Hence following observations.

```
not(falseIdentity) == trueIdentity
```

```
not(trueIdentity) == falseIdentity
```

Use the not operator as follows.

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
val rule3=rule1 and rule2
```

```
val rule4=rule3 or trueIdentity
```

```
rule4 and not(falseIdentity)
```

The above expression evaluates to true.

All operator

all takes a list of behaviours and evaluates to true only if all of the behaviours evaluate to true.

```
all(falseIdentity) == falseIdentity
```

```
all(falseIdentity,falseIdentity) == falseIdentity
```

```
all(trueIdentity) == trueIdentity
```

```
all(trueIdentity,trueIdentity) == trueIdentity
```

```
all(trueIdentity,falseIdentity) == falseIdentity
```

```
all(falseIdentity,........) == falseIdentity
```

Use the all operator as follows.

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
val rule3=rule1 and rule2
```

```
val rule4=rule3 or trueIdentity
```

```
val rule5=rule4 and not(falseIdentity)
```

```
all(rule5,trueIdentity)
```

The above expression evaluates to true.

Brackets

Use brackets to either remove ambiguity or group expressions to improve readability. For example, consider following expression:

```
trueIdentity or trueIdentity and falseIdentity
```

What does above expression evaluates to? If or executes first, this will result in false. Otherwise in true. To remove this ambiguity you can be explicit by grouping sub-expressions using brackets as follows.Hence following observations.

```
trueIdentity or (trueIdentity and falseIdentity) == trueIdentity
```

```
(trueIdentity or trueIdentity) and falseIdentity == falseIdentity
```

Use Brackets as follows.

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
val rule3=rule1 and rule2
```

```
val rule4=rule3 or trueIdentity
```

```
val rule5=rule4 and not(falseIdentity)
```

```
(all(rule5,trueIdentity) and falseIdentity) or trueIdentity
```

The above expression evaluates to true.

Any Operator

Keywords: any

The any operator takes a list of behaviours and evaluates to true if any one of the behaviours evaluate to true.

```
any(falseIdentity) == falseIdentity
```

```
any(falseIdentity,falseIdentity) == falseIdentity
```

```
any(trueIdentity) == trueIdentity
```

```
any(trueIdentity,trueIdentity) == trueIdentity
```

```
any(trueIdentity,falseIdentity) == trueIdentity
```

```
any(trueIdentity,........) == trueIdentity
```

Use the any operator as follows:

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
val rule3=rule1 and rule2
```

```
val rule4=rule3 or trueIdentity
```

```
val rule5=rule4 and not(falseIdentity)
```

```
val rule6=(all(rule5,trueIdentity) and falseIdentity) or trueIdentity
```

```
any(rule6,falseIdentity,falseIdentity)
```

The above expression evaluates to true.

Given-Choose-Otherwise Operation

Keywords: given,choose,otherwise

Given-Choose-Otherwise works like if conditions in languages like Java and Scala. If given behaviour evaluates to true, behaviour specified for choose is evaluated as a result else behaviour specified for otherwise is evaluated as a result(or falseIdentity if otherwise clause is missing)

```
given(falseIdentity) choose falseIdentity == falseIdentity
```

```
given(falseIdentity) choose trueIdentity == falseIdentity
```

```
given(trueIdentity) choose falseIdentity == falseIdentity
```

```
given(trueIdentity) choose trueIdentity == trueIdentity
```

```
given(falseIdentity) choose falseIdentity otherwise trueIdentity ==
 trueIdentity
```

```
given(falseIdentity) choose falseIdentity otherwise falseIdentity ==
 falseIdentity
```

```
given(falseIdentity) choose trueIdentity otherwise trueIdentity ==
 trueIdentity
```

```
given(falseIdentity) choose trueIdentity otherwise falseIdentity ==
 falseIdentity
```

```
given(trueIdentity) choose falseIdentity otherwise falseIdentity ==
 falseIdentity
```

```
given(trueIdentity) choose falseIdentity otherwise trueIdentity ==
 falseIdentity
```

```
given(trueIdentity) choose trueIdentity otherwise falseIdentity ==
 trueIdentity
```

```
given(trueIdentity) choose trueIdentity otherwise trueIdentity ==
 trueIdentity
```

Use given, choose, otherwise operator as follows.

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
val rule3=rule1 and rule2
```

```
val rule4=rule3 or trueIdentity
```

```
val rule5=rule4 and not(falseIdentity)
```

```
val rule6=(all(rule5,trueIdentity) and falseIdentity) or trueIdentity
```

```
val rule7=any(rule6,falseIdentity,falseIdentity)
```

```
given(rule7) choose falseIdentity otherwise trueIdentity
```

The above expression evaluates to false.

Accept or Reject Auto-Suggested Tags

The Sensitive Data Profiler detects data types and automatically suggests tags. If accepted, the tags are pushed back to Apache Atlas. Automatically suggested tags display as purple on the Schema tab.

Procedure

- To accept tags: **Schema tab** > **Edit Tags** > **Click purple tags** > **Save**.

  Once accepted by the user, the tag is saved back to Atlas in the cluster. In Atlas, the tag will be prefixed with “dp_” to denote that it comes from Dataplane Service.
- To reject tags: **Schema tab** > **Edit Tags** > **hover over purple tags** > **click (x) icon**.

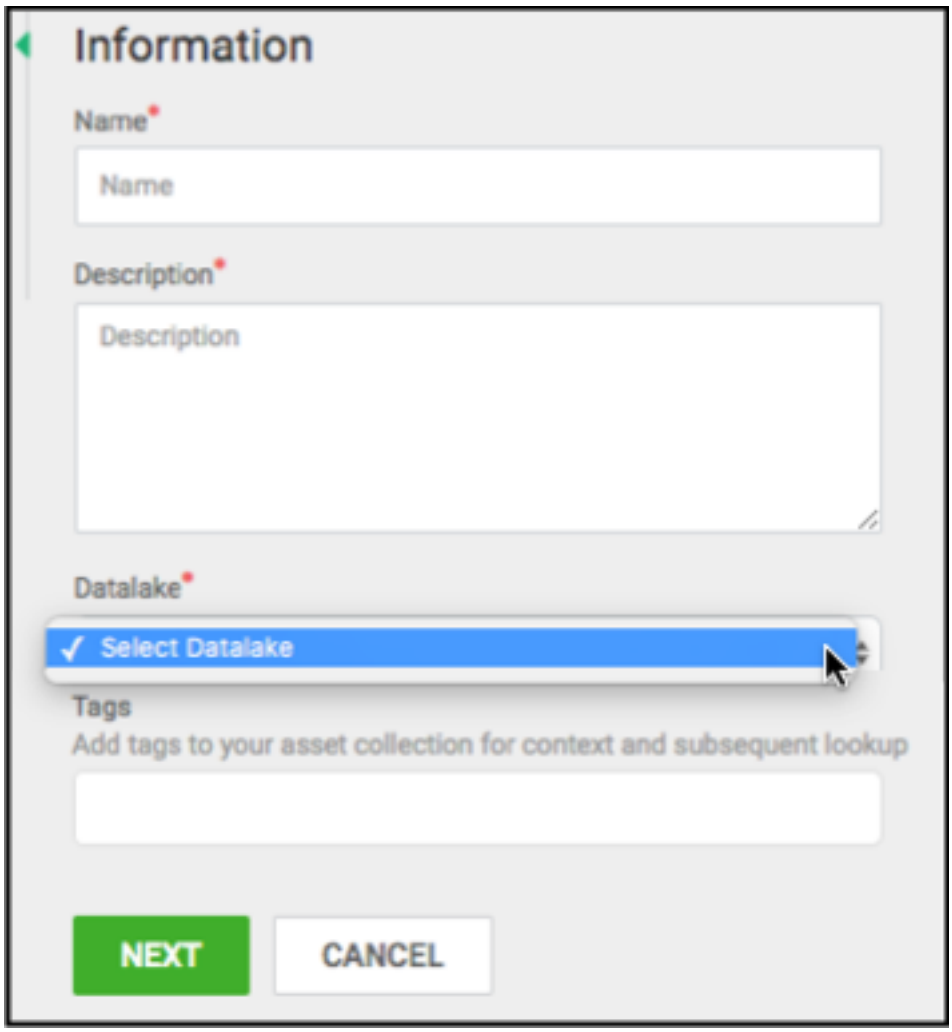
DSS Troubleshooting

This chapter contains common issues (with workarounds) and error message help for Data Steward Studio (DSS).

No data lake available when creating an Asset Collection

When creating an Asset Collection, no datalake displays in the drop-down menu.

A datalake is a cluster that has Apache Atlas and Apache Ranger installed. If registered clusters do not have Apache Atlas installed or there are no clusters registered to Hortonworks DataPlane Service, then no datalakes are available.



Procedure

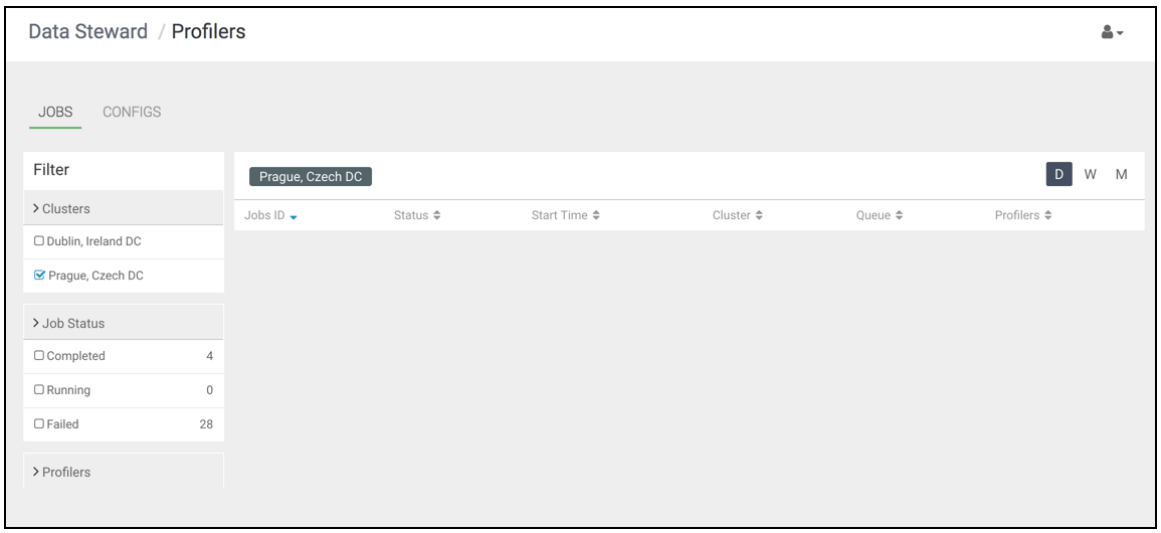
Register the cluster or install Apache Atlas and Apache Ranger on the cluster:

- Register a cluster in DataPlane
- Install Apache Atlas
- Install Apache Ranger

Profiler data does not load

Condition

When loading the **Profilers** tabs, profiler data does not load.



Cause

In Ambari, the Dataplane Profiler service is down.

Remedy

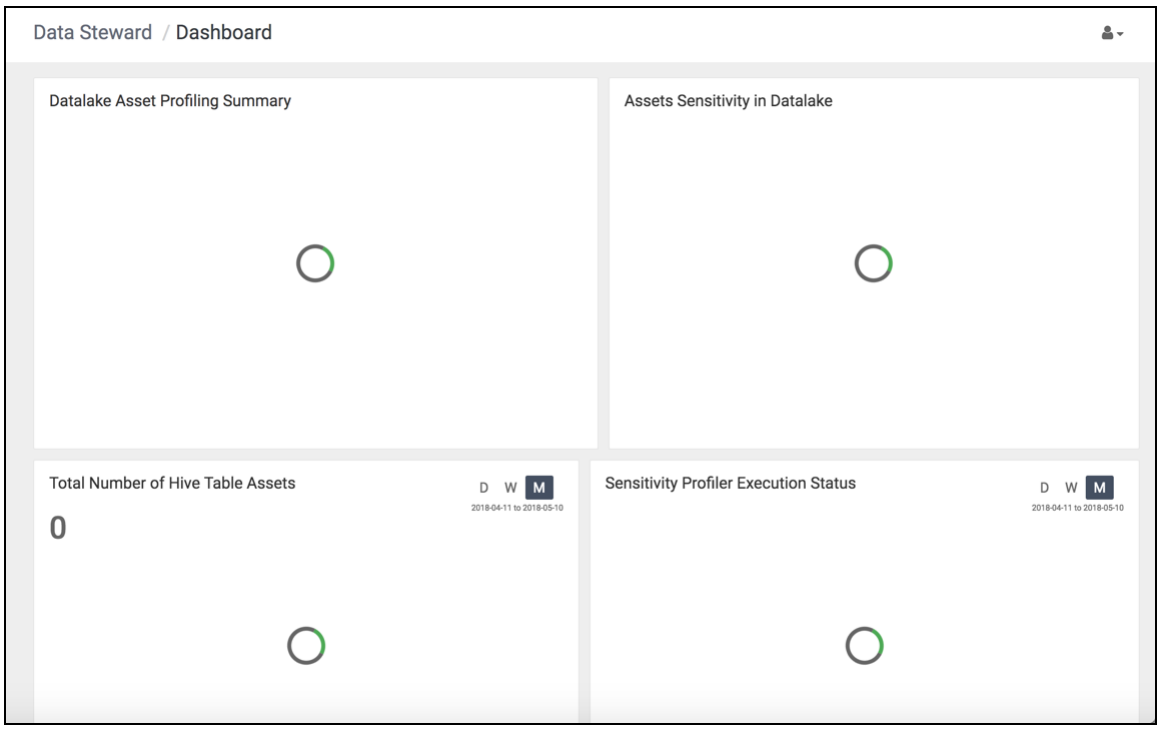
Procedure

Go to **Ambari** > **Dataplane Profiler** and turn the service on.

Widgets do not load on dashboard

Condition

When loading the **Dashboard**, the widgets do not load correctly (no data loaded).



Cause

There are errors occurring on the Profiler jobs.

Procedure

- Check for failed profiler jobs:
  a) From **Profiler** > **Jobs**, filter to the cluster whose dashboard is failing.
  b) Filter the job status to **Failed**.
  c) Use these failed profiler jobs to help troubleshoot the root cause.
- Verify that the Dataplane Profiler service is running in Ambari.
  a) Go to **Ambari** > **Dataplane Profiler** and check the status of the service.
  b) If the service is down, turn it on.