

# Hortonworks Data Platform

Data Governance

(August 31, 2017)

## Hortonworks Data Platform: Data Governance

Copyright © 2012-2017 Hortonworks, Inc. Some rights reserved.

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, [training](#) and partner-enablement services. All of our technology is, and will remain, free and open source.

Please visit the [Hortonworks Data Platform](#) page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the [Support](#) or [Training](#) page. Feel free to [contact us](#) directly to discuss your specific needs.



Except where otherwise noted, this document is licensed under **Creative Commons Attribution ShareAlike 4.0 License**.  
<http://creativecommons.org/licenses/by-sa/4.0/legalcode>

## Table of Contents

1. HDP Data Governance .....	1
1.1. Apache Atlas Features .....	1
1.2. Atlas-Ranger Integration .....	2
2. Installing and Configuring Apache Atlas Using Ambari .....	5
2.1. Apache Atlas Prerequisites .....	5
2.2. Atlas Installation .....	5
2.2.1. Start the Installation .....	5
2.2.2. Customize Services .....	9
2.2.3. Dependent Configurations .....	14
2.2.4. Configure Identities .....	15
2.2.5. Complete the Atlas Installation .....	16
2.3. Enable the Ranger Plugin .....	19
2.4. Configure Atlas Tagsync in Ranger .....	19
2.5. Configure Atlas High Availability .....	19
2.6. Configuring Atlas Security .....	19
2.6.1. Additional Requirements for Atlas with Ranger and Kerberos .....	19
2.6.2. Enable Atlas HTTPS .....	22
2.6.3. Hive CLI Security .....	22
2.7. Installing Sample Atlas Metadata .....	22
2.8. Updating the Atlas Ambari Configuration .....	23
3. Searching and Viewing Entities .....	24
3.1. Using Basic and Advanced Search .....	24
3.1.1. Using Basic Search .....	24
3.1.2. Using Advanced Search .....	27
3.2. Viewing Entity Data Lineage & Impact .....	29
3.3. Viewing Entity Details .....	31
3.4. Manually Creating Entities .....	35
4. Working with Atlas Tags .....	38
4.1. Creating Atlas Tags .....	38
4.2. Associating Tags with Entities .....	39
4.3. Searching for Entities Associated with Tags .....	42
5. Apache Atlas REST API .....	44

# List of Figures

1.1. Atlas Overview ..... 2

## List of Tables

2.1. Apache Atlas LDAP Configuration Settings .....	11
2.2. Apache Atlas AD Configuration Settings .....	12
2.3. Apache Atlas Simple Authorization .....	13
2.4. Ranger Atlas Service Kerberos Properties .....	20

# 1. HDP Data Governance

Apache Atlas provides governance capabilities for Hadoop that use both prescriptive and forensic models enriched by business taxonomical metadata. Atlas is designed to exchange metadata with other tools and processes within and outside of the Hadoop stack, thereby enabling platform-agnostic governance controls that effectively address compliance requirements.

Apache Atlas enables enterprises to effectively and efficiently address their compliance requirements through a scalable set of core governance services. These services include:

- Search and Proscriptive Lineage – facilitates pre-defined and *ad hoc* exploration of data and metadata, while maintaining a history of data sources and how specific data was generated.
- Metadata-driven data access control.
- Flexible modeling of both business and operational data.
- Data Classification – helps you to understand the nature of the data within Hadoop and classify it based on external and internal sources.
- Metadata interchange with other metadata tools.



## Note

The Apache Atlas Business Taxonomy feature, which was a Technical Preview in previous releases, is currently being redesigned and will be reintroduced in a future release.

## 1.1. Apache Atlas Features

Apache Atlas is a low-level service in the Hadoop stack that provides core metadata services. Atlas currently provides metadata services for the following components:

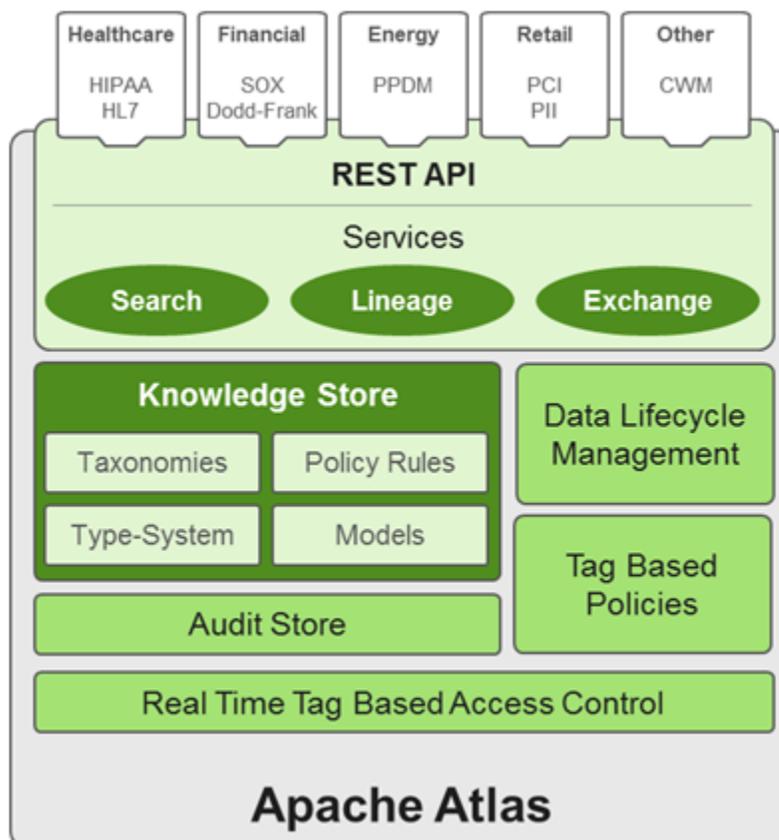
- Hive
- Ranger
- Sqoop
- Storm/Kafka (limited support)
- Falcon (limited support)

Apache Atlas provides the following features:

- **Knowledge store that leverages existing Hadoop metastores:** Categorized into a business-oriented taxonomy of data sets, objects, tables, and columns. Supports the exchange of metadata between HDP foundation components and third-party applications or governance tools.

- **Data lifecycle management:** Leverages existing investment in Apache Falcon with a focus on provenance, multi-cluster replication, data set retention and eviction, late data handling, and automation.
- **Audit store:** Historical repository for all governance events, including security events (access, grant, deny), operational events related to data provenance and metrics. The Atlas audit store is indexed and searchable for access to governance events.
- **Security:** Integration with HDP security that enables you to establish global security policies based on data classifications and that leverages Apache Ranger plug-in architecture for security policy enforcement.
- **Policy engine:** Fully extensible policy engine that supports metadata-based, geo-based, and time-based rules that rationalize at runtime.
- **RESTful interface:** Supports extensibility by way of REST APIs to third-party applications so you can use your existing tools to view and manipulate metadata in the HDP foundation components.

Figure 1.1. Atlas Overview



## 1.2. Atlas-Ranger Integration

Atlas provides data governance capabilities and serves as a common metadata store that is designed to exchange metadata both within and outside of the Hadoop stack. Ranger

provides a centralized user interface that can be used to define, administer and manage security policies consistently across all the components of the Hadoop stack. The Atlas-Ranger unites the data classification and metadata store capabilities of Atlas with security enforcement in Ranger.

You can use Atlas and Ranger to implement dynamic classification-based security policies, in addition to role-based security policies. Ranger's centralized platform empowers data administrators to define security policy based on Atlas metadata tags or attributes and apply this policy in real-time to the entire hierarchy of entities including databases, tables, and columns, thereby preventing security violations.

### Ranger-Atlas Access Policies

- **Classification-based access controls:** A data entity such as a table or column can be marked with the metadata tag related to compliance or business taxonomy (such as "PCI"). This tag is then used to assign permissions to a user or group. This represents an evolution from role-based entitlements, which require discrete and static one-to-one mapping between user/group and resources such as tables or files. As an example, a data steward can create a classification tag "PII" (Personally Identifiable Information) and assign certain Hive table or columns to the tag "PII". By doing this, the data steward is denoting that any data stored in the column or the table has to be treated as "PII". The data steward now has the ability to build a security policy in Ranger for this classification and allow certain groups or users to access the data associated with this classification, while denying access to other groups or users. Users accessing any data classified as "PII" by Atlas would be automatically enforced by the Ranger policy already defined.
- **Data Expiry-based access policy:** For certain business use cases, data can be toxic and have an expiration date for business usage. This use case can be achieved with Atlas and Ranger. Apache Atlas can assign expiration dates to a data tag. Ranger inherits the expiration date and automatically denies access to the tagged data after the expiration date.
- **Location-specific access policies:** Similar to time-based access policies, administrators can now customize entitlements based on geography. For example, a US-based user might be granted access to data while she is in a domestic office, but not while she is in Europe. Although the same user may be trying to access the same data, the different geographical context would apply, triggering a different set of privacy rules to be evaluated.
- **Prohibition against dataset combinations:** With Atlas-Ranger integration, it is now possible to define a security policy that restricts combining two data sets. For example, consider a scenario in which one column consists of customer account numbers, and another column contains customer names. These columns may be in compliance individually, but pose a violation if combined as part of a query. Administrators can now apply a metadata tag to both data sets to prevent them from being combined.

### Cross Component Lineage

Apache Atlas now provides the ability to visualize cross-component lineage, delivering a complete view of data movement across a number of analytic engines such as Apache Storm, Kafka, Falcon, and Hive.

This functionality offers important benefits to data stewards and auditors. For example, data that starts as event data through a Kafka bolt or Storm Topology is also analyzed

as an aggregated dataset through Hive, and then combined with reference data from a RDBMS via Sqoop, can be governed by Atlas at every stage of its lifecycle. Data stewards, Operations, and Compliance now have the ability to visualize a data set's lineage, and then drill down into operational, security, and provenance-related details. As this tracking is done at the platform level, any application that uses these engines will be natively tracked. This allows for extended visibility beyond a single application view.

## 2. Installing and Configuring Apache Atlas Using Ambari

### 2.1. Apache Atlas Prerequisites

Apache Atlas requires the following components:

- Ambari Infra (which includes an internal HDP Solr Cloud instance) or an externally managed Solr Cloud instance.
- HBase (used as the Atlas Metastore).
- Kafka (provides a durable messaging bus).

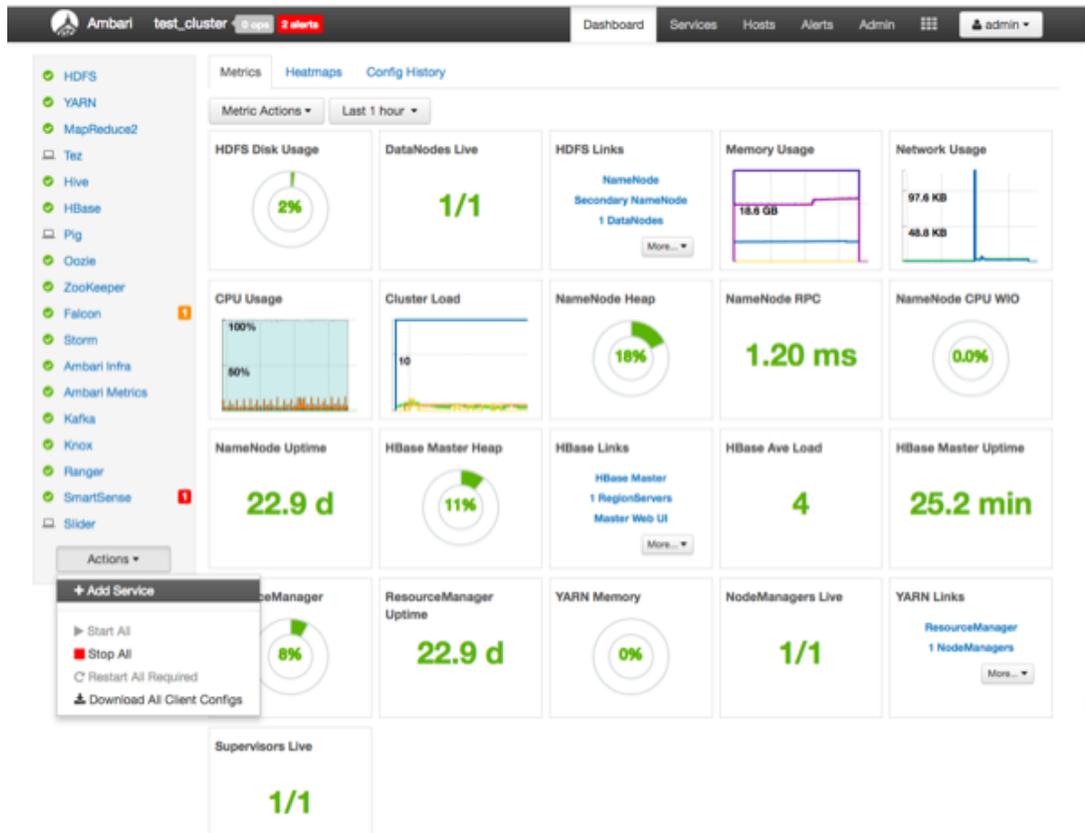
### 2.2. Atlas Installation

To install Atlas using Ambari:

1. [Start the Installation \[5\]](#)
2. [Customize Services \[9\]](#)
3. [Complete the Atlas Installation \[16\]](#)

#### 2.2.1. Start the Installation

1. On the Ambari Dashboard, click **Actions**, then select **Add Service**.



2. On the Choose Services page, select **Atlas**, then click **Next**.

### Add Service Wizard

ADD SERVICE WIZARD

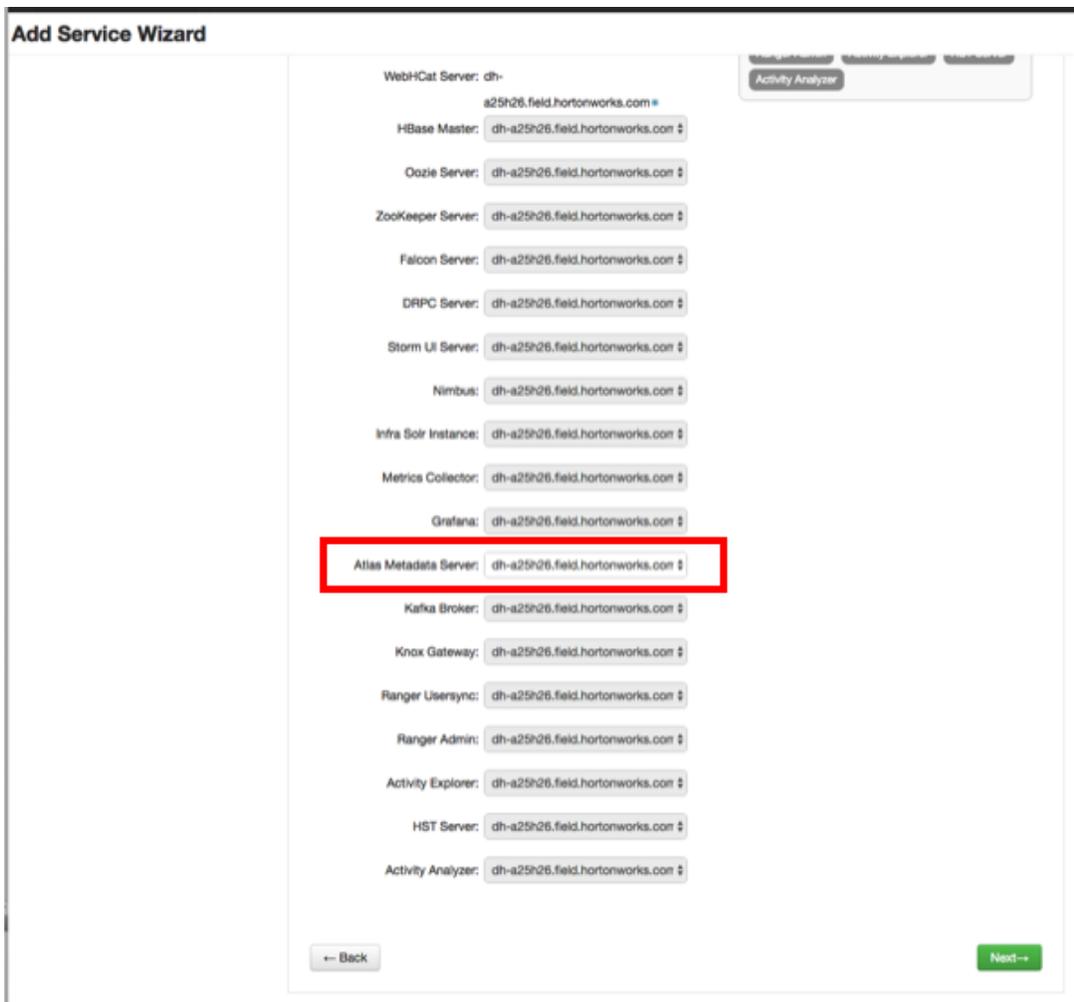
- Choose Services**
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Configure Identities
- Review
- Install, Start and Test
- Summary

### Choose Services

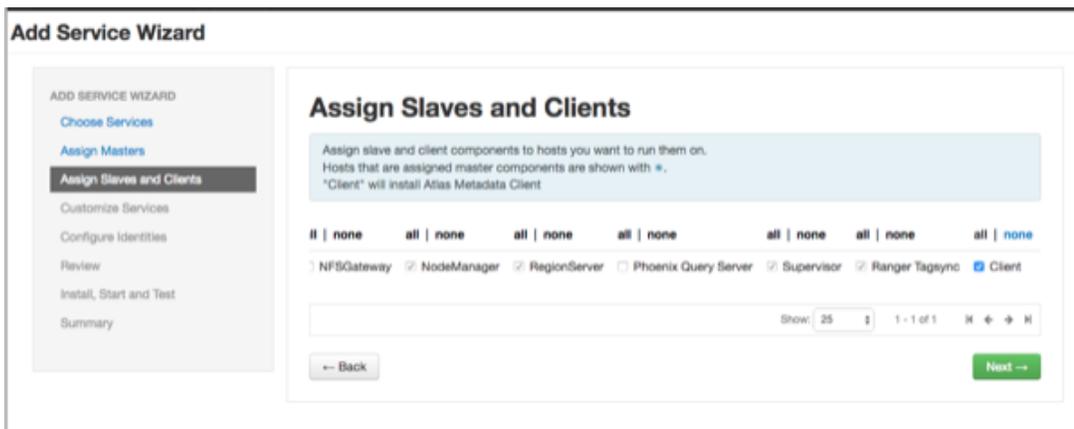
Choose which services you want to install on your cluster.

<input type="checkbox"/> Service	Version	Description
<input checked="" type="checkbox"/> HDFS	2.7.3	Apache Hadoop Distributed File System
<input checked="" type="checkbox"/> YARN + MapReduce2	2.7.3	Apache Hadoop NextGen MapReduce (YARN)
<input checked="" type="checkbox"/> Tez	0.7.0	Tez is the next generation Hadoop Query Processing framework written on top of YARN.
<input checked="" type="checkbox"/> Hive	1.2.1000	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service
<input checked="" type="checkbox"/> HBase	1.1.2	A Non-relational distributed database, plus Phoenix, a high performance SQL layer for low latency applications.
<input checked="" type="checkbox"/> Pig	0.16.0	Scripting platform for analyzing large datasets
<input type="checkbox"/> Sqoop	1.4.6	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases
<input checked="" type="checkbox"/> Oozie	4.2.0	System for workflow coordination and execution of Apache Hadoop jobs. This also includes the installation of the optional Oozie Web Console which relies on and will install the <a href="#">ExtJS Library</a> .
<input checked="" type="checkbox"/> ZooKeeper	3.4.6	Centralized service which provides highly reliable distributed coordination
<input checked="" type="checkbox"/> Falcon	0.10.0	Data management and processing platform
<input checked="" type="checkbox"/> Storm	1.1.0	Apache Hadoop Stream processing framework
<input type="checkbox"/> Flume	1.5.2	A distributed service for collecting, aggregating, and moving large amounts of streaming data into HDFS
<input type="checkbox"/> Accumulo	1.7.0	Robust, scalable, high performance distributed key/value store.
<input checked="" type="checkbox"/> Ambari Infra	0.1.0	Core shared service used by Ambari managed components.
<input checked="" type="checkbox"/> Ambari Metrics	0.1.0	A system for metrics collection that provides storage and retrieval capability for metrics collected from the cluster
<input checked="" type="checkbox"/> Atlas	0.8.0	Atlas Metadata and Governance platform
<input checked="" type="checkbox"/> Kafka	0.10.1	A high-throughput distributed messaging system
<input checked="" type="checkbox"/> Knox	0.12.0	Provides a single point of authentication and access for Apache Hadoop services in a cluster

3. The Assign Master page appears. Specify a host for the Atlas Metadata Server, then click **Next**.



4. The Assign Slaves and Clients page appears with Client (the Atlas Metadata Client) selected. Click **Next** to continue.



5. The Customize Services page appears. These settings are described in the next section.

## 2.2.2. Customize Services

The next step in the installation process is to specify Atlas settings on the Customize Services page.

### 2.2.2.1. Authentication Settings

You can set the Authentication Type to File, LDAP, or AD.

**Add Service Wizard**

ADD SERVICE WIZARD

- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services**
- Configure Identities
- Review
- Install, Start and Test
- Summary

### Customize Services

We have come up with recommended configurations for the services you selected. Customize them as you see fit.

HDFS YARN MapReduce2 Tez Hive HBase Pig Oozie ZooKeeper Falcon Storm Ambari Infra  
Ambari Metrics **Atlas** Kafka Knox Ranger SmartSense Slider Misc

There are 8 configuration changes in 4 services [Show Details](#)

Group: Default (1) Manage Config Groups Filter...

Authentication **Advanced**

#### Authentication Methods

- Enable File Authentication
- Enable LDAP Authentication
- Enable Atlas Knox SSO

#### File

atlas.authentication.method.file.filename

{{conf\_dir}}/users-credentials.properties

#### LDAP/AD

LDAP Authentication Type

AD

LDAP

AD ldap.ad.url

10.42.0.53

#### 2.2.2.1.1. File-based Authentication

When file-based authentication is selected, the `atlas.authentication.method.file.filename` property is automatically set to `{{conf_dir}}/users-credentials.properties`.

The `users-credentials.properties` file should have the following format:

```
username=group::sha256password
admin=ADMIN::e7cf3ef4f17c3999a94f2c6f612e8a888e5b1026878e4e19398b23bd38ec221a
```

The user group can be `ADMIN`, `DATA_STEWARD`, or `DATA_SCIENTIST`.

The password is encoded with the `sha256` encoding method and can be generated using the UNIX tool:

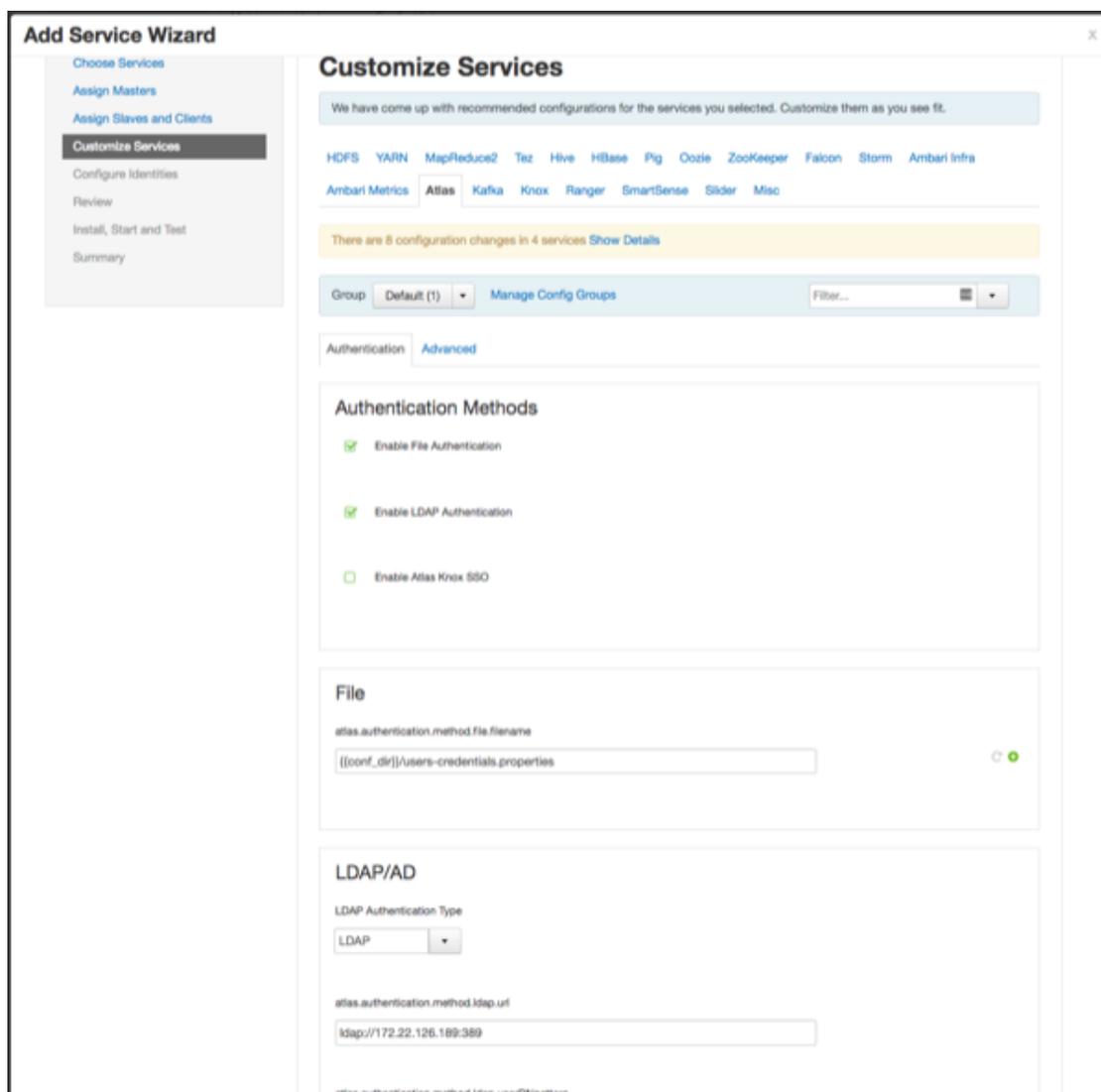
```
echo -n "Password" | sha256sum
e7cf3ef4f17c3999a94f2c6f612e8a888e5b1026878e4e19398b23bd38ec221a -
```

### 2.2.2.1.2. LDAP Authentication

To enable LDAP authentication, select **LDAP**, then set the following configuration properties.

**Table 2.1. Apache Atlas LDAP Configuration Settings**

Property	Sample Values
atlas.authentication.method.ldap.url	ldap://127.0.0.1:389
atlas.authentication.method.ldap.userDNpattern	uid={0},ou=users,dc=example,dc=com
atlas.authentication.method.ldap.groupSearchBase	dc=example,dc=com
atlas.authentication.method.ldap.groupSearchFilter	(member=cn={0},ou=users,dc=example,dc=com
atlas.authentication.method.ldap.groupRoleAttribute	cn
atlas.authentication.method.ldap.base.dn	dc=example,dc=com
atlas.authentication.method.ldap.bind.dn	cn=Manager,dc=example,dc=com
atlas.authentication.method.ldap.bind.password	PassW0rd
atlas.authentication.method.ldap.referral	ignore
atlas.authentication.method.ldap.user.searchfilter	(uid={0})
atlas.authentication.method.ldap.default.role	ROLE_USER

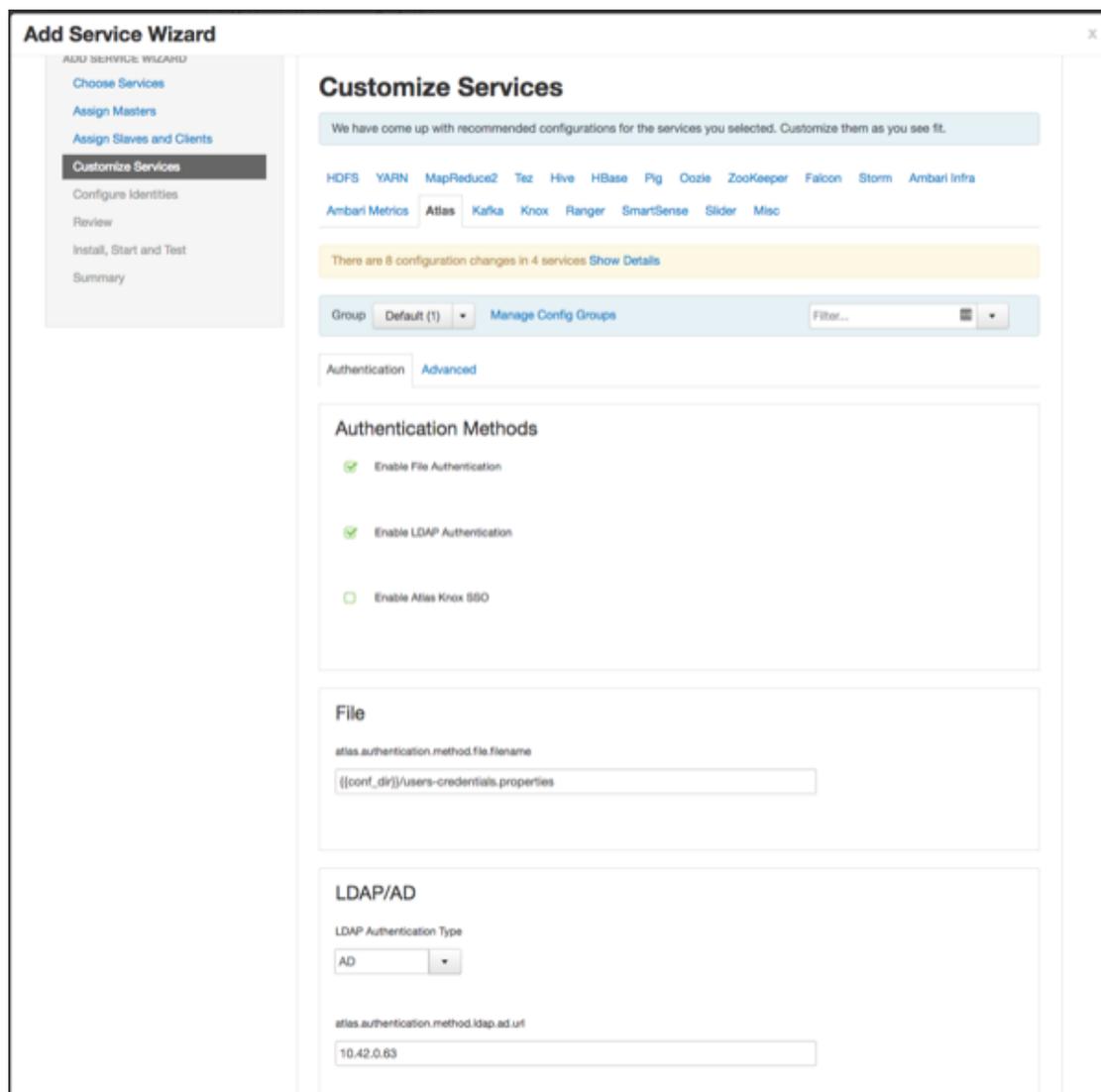


### 2.2.2.1.3. AD Authentication

To enable AD authentication, select **AD**, then set the following configuration properties.

**Table 2.2. Apache Atlas AD Configuration Settings**

Property	Sample Values
atlas.authentication.method.ldap.ad.url	ldap://127.0.0.1:389
Domain Name (Only for AD)	example.com
atlas.authentication.method.ldap.ad.base.dn	DC=example,DC=com
atlas.authentication.method.ldap.ad.bind.dn	CN=Administrator,CN=Users,DC=example,DC=com
atlas.authentication.method.ldap.ad.bind.password	PassW0rd
atlas.authentication.method.ldap.ad.referral	ignore
atlas.authentication.method.ldap.ad.user.searchfilter	(sAMAccountName={0})
atlas.authentication.method.ldap.ad.default.role	ROLE_USER



## 2.2.2.2. Authorization Settings

Two authorization methods are available for Atlas: Simple and Ranger.

### 2.2.2.2.1. Simple Authorization

The default setting is Simple, and the following properties are automatically set under **Advanced application-properties** on the Advanced tab.

**Table 2.3. Apache Atlas Simple Authorization**

Property	Value
atlas.authorizer.impl	simple
atlas.auth.policy.file	{{conf_dir}}/policy-store.txt

The screenshot shows the 'Add Service Wizard' window with the 'Advanced' tab selected. Under the 'Advanced application-properties' section, the following properties are visible:

- atlas.audit.hbase.tablename: ATLAS\_ENTITY\_AUDIT\_EVENTS
- atlas.audit.hbase.zookeeper.quorum: c6406.ambari.apache.org
- atlas.audit.zookeeper.session.timeout.ms: 1000
- atlas.auth.policy.file: {{conf\_dir}}/policy-store.txt** (highlighted with a red box)
- atlas.authentication.keytab: /etc/security/keytabs/atlas.service.keytab
- atlas.authentication.method: true
- atlas.authentication.method.file.filename: {{conf\_dir}}/users-credentials.properties
- atlas.authentication.method.kerberos: false
- atlas.authentication.method.idap: false
- atlas.authentication.principal: atlas
- atlas.authorizer.impl: simple** (highlighted with a red box)
- atlas.cluster.name: {{cluster\_name}}
- atlas.enableTLS: false
- atlas.graph.index.search.backend: solr5
- atlas.graph.index.search.solr.mode: cloud
- atlas.graph.index.search.solr.zookeeper-uri: c6406.ambari.apache.org:2181/infra-solr
- atlas.graph.storage: hbase

The `policy-store.txt` file has the following format:

```
Policy_Name ; User_Name : Operations_Allowed ; Group_Name : Operations_Allowed ; Resource_Type : Resource
```

For example:

```
adminPolicy;;admin:rwud;;ROLE_ADMIN:rwud;;type:*,entity:*,operation:*,
taxonomy:*,term:*
userReadPolicy;;readUser1:r,readUser2:r;;DATA_SCIENTIST:r;;type:*,entity:*,
operation:*,taxonomy:*,term:*
userWritePolicy;;writeUser1:rwu,writeUser2:rwu;;BUSINESS_GROUP:rwu,
DATA_STEWARD:rwud;;type:*,entity:*,operation:*,taxonomy:*,term:*
```

In this example `readUser1`, `readUser2`, `writeUser1` and `writeUser2` are the user IDs, each with its corresponding access rights. The `User_Name`, `Group_Name` and `Operations_Allowed` are comma-separated lists.

#### Authorizer Resource Types:

- Operation
- Type
- Entity
- Taxonomy
- Term
- Unknown

`Operations_Allowed` are `r` = read, `w` = write, `u` = update, `d` = delete

#### 2.2.2.2. Ranger Authorization

Ranger Authorization is activated by [enabling the Ranger Atlas plug-in](#) in Ambari.

### 2.2.3. Dependent Configurations

After you customize Atlas services and click **Next**, the Dependent Configurations page displays recommended settings for dependent configurations. Clear the checkbox next to a property to retain the current value. Click **OK** to set the selected recommended property values.

**Dependent Configurations**

**Recommended Changes**

Based on your configuration changes, Ambari is recommending the following dependent configuration changes. Ambari will update all checked configuration changes to the Recommended Value. Uncheck any configuration to retain the Current Value.

<input checked="" type="checkbox"/>	Property	Service	Config Group	File Name	Current Value	Recommended Value
<input checked="" type="checkbox"/>	hive.atlas.hook	Hive	Default	hive-env	false	true
<input checked="" type="checkbox"/>	hive.exec.post.hooks	Hive	Default	hive-site	org.apache.hadoop.hive.ql.hooks.ATSBok	org.apache.hadoop.hive.ql.hooks.ATSBok,org.apache.atlas.hive.hook.HiveHook
<input checked="" type="checkbox"/>	falcon.atlas.hook	Falcon	Default	falcon-env	false	true
<input checked="" type="checkbox"/>	storm.atlas.hook	Storm	Default	storm-env	false	true
<input checked="" type="checkbox"/>	ranger.tagsync.source.atlas	Ranger	Default	ranger-tagsync-site	false	true
<input checked="" type="checkbox"/>	ranger.tagsync.source.atlasrest.endpoint	Ranger	Default	ranger-tagsync-site		http://dh-a25h26.field.hortonworks.com:21000
<input checked="" type="checkbox"/>	atlas.rest.address	Hive	Default	hive-site	Property undefined	http://dh-a25h26.field.hortonworks.com:21000
<input checked="" type="checkbox"/>	storm.topology.submission.notifier.plugin.class	Storm	Default	storm-site	Property undefined	org.apache.atlas.storm.hook.StormAtlasHook

Cancel OK

If Ambari detects other configuration issues, they will be displayed on a Configurations pop-up. Click **Cancel** to go back and change these settings, or click **Proceed Anyway** to continue the installation without changing the configurations.

**Configurations**

Some service configurations are not configured properly. We recommend you review and change the highlighted configuration values. Are you sure you want to proceed without correcting configurations?

Type	Service	Property	Value	Description
Warning	Atlas	atlas.graph.storage.hostname	dh-a25h26rk.field.hortonworks.com	Atlas is configured to use the HBase installed in this cluster. If you would like Atlas to use another HBase instance, please configure this property and HBASE_CONF_DIR variable in atlas-env appropriately.

Cancel Proceed Anyway

## 2.2.4. Configure Identities

If Kerberos is enabled, the Configure Identities page appears. Click **Next** to continue with the installation.

**Add Service Wizard**

Choose Services  
Assign Masters  
Assign Slaves and Clients  
Customize Services  
**Configure Identities**  
Review  
Install, Start and Test  
Summary

### Configure Identities

Configure principal name and keytab location for service users and hadoop service components.

General **Advanced**

**Global**

Keytab Dir: /etc/security/keytabs  
Realm: EXAMPLE.COM  
Additional Realms:   
Principal Suffix: \${cluster\_name}toLower()  
Spnego Keytab: \${keytab\_dir}/spnego.service.keytab  
Spnego Principal: HTTP/\_HOST@\${realm}

**Ambari Principals**

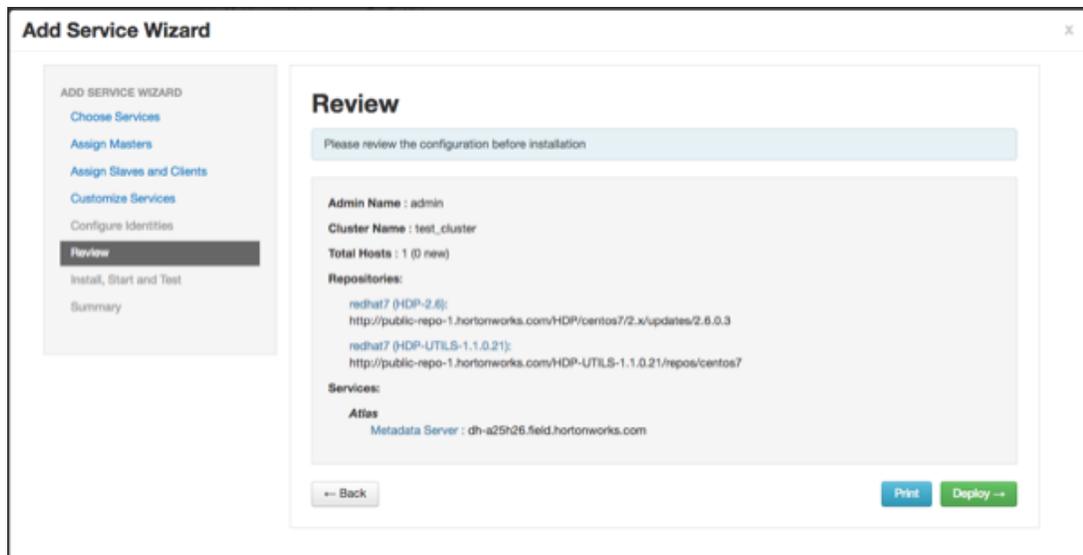
Smoke user keytab: \${keytab\_dir}/smokeuser.headless.keytab  
Smoke user principal: \${cluster-env}/smokeuser\${principal\_suffix}@\${realm}  
Ambari Keytab: \${keytab\_dir}/ambari.server.keytab  
Ambari Principal Name: ambari-server\${principal\_suffix}@\${realm}  
HBase user principal: \${hbase-env}/hbase\_user\${principal\_suffix}@\${realm}  
HBase user keytab: \${keytab\_dir}/hbase.headless.keytab  
HDFS user principal: \${hadoop-env}/hdfs\_user\${principal\_suffix}@\${realm}  
HDFS user keytab: \${keytab\_dir}/hdfs.headless.keytab  
Storm user keytab: \${keytab\_dir}/storm.headless.keytab  
Storm user principal: \${storm-env}/storm\_user\${principal\_suffix}@\${realm}

All configurations have been addressed.

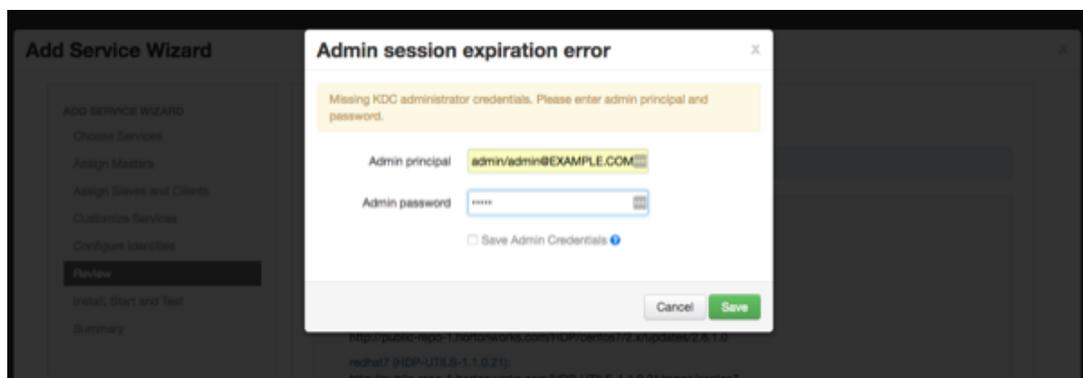
← Back Next →

## 2.2.5. Complete the Atlas Installation

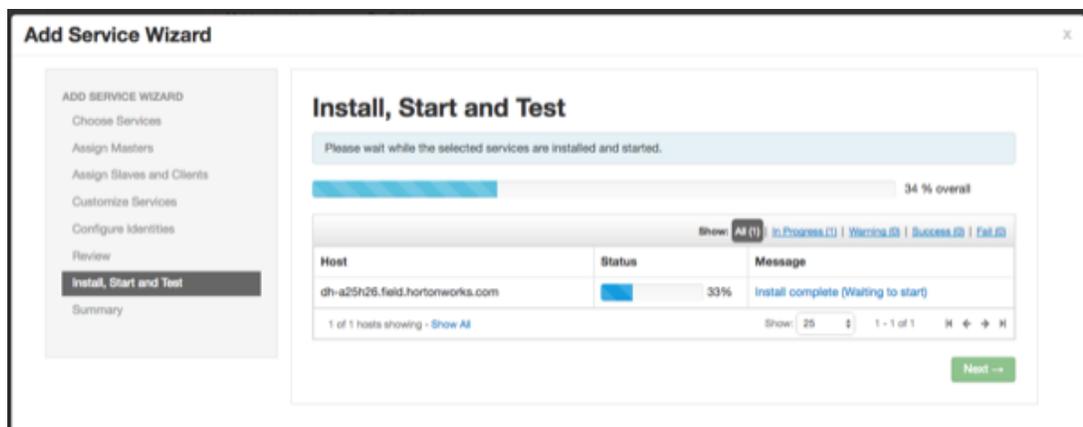
1. On the Review page, carefully review all of your settings and configurations. If everything looks good, click **Deploy** to install Atlas on the Ambari server.



If Kerberos is enabled, you are prompted to enter your KDC administrator credentials. Type in your KDC Admin principal and password, then click **Save**.



- When you click **Deploy**, Atlas is installed on the specified host on your Ambari server. A progress bar displays the installation progress.

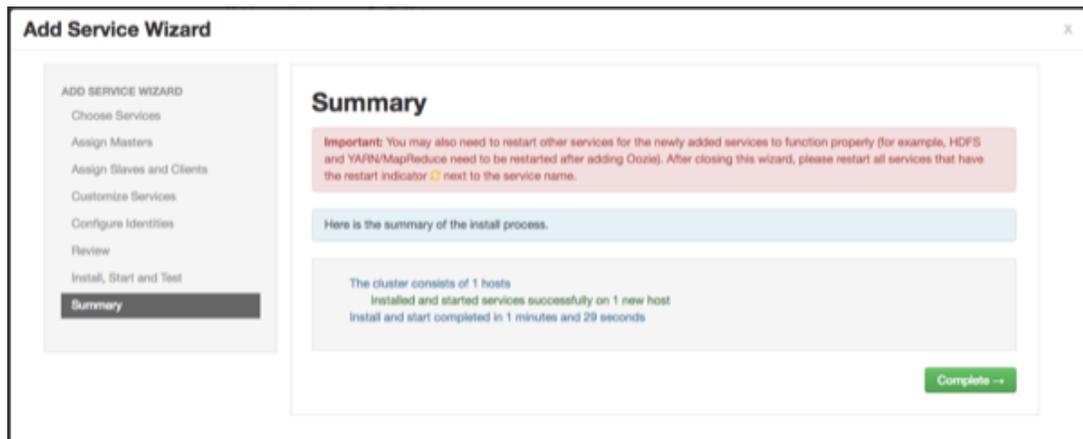


- When the installation is complete, a Summary page displays the installation details. Click **Complete** to finish the installation.

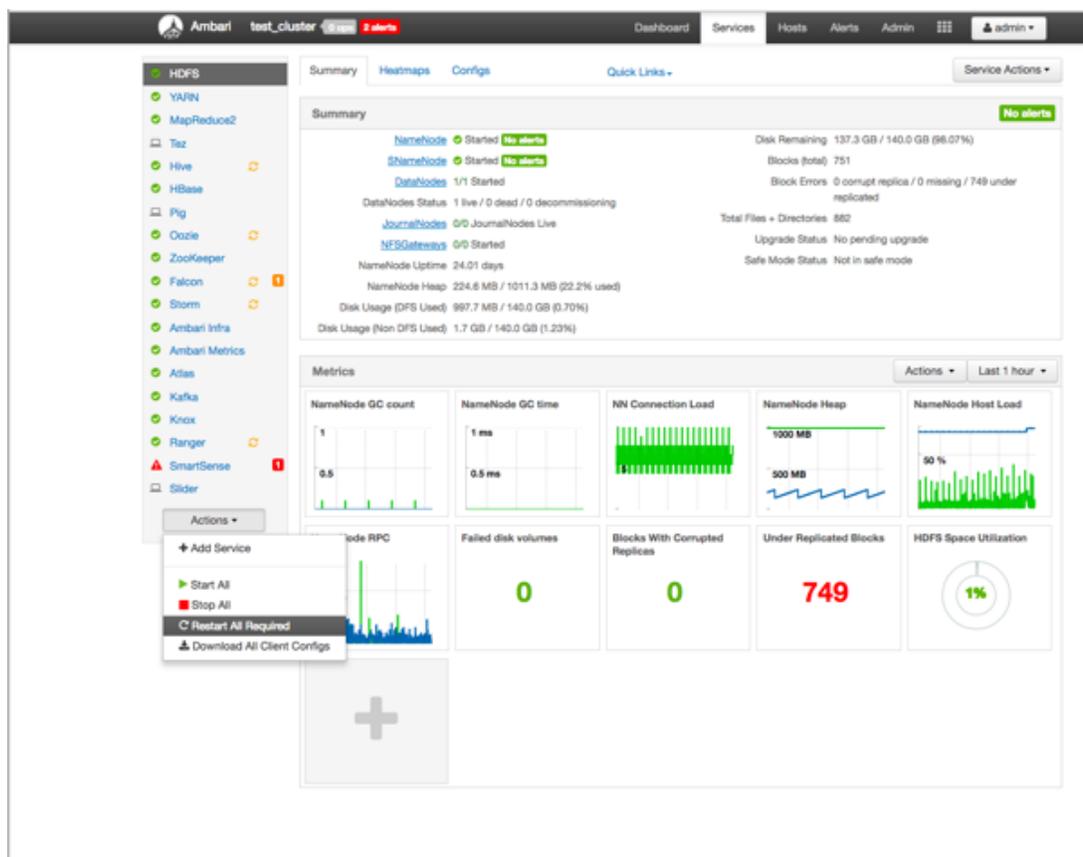


**Note**

The Atlas user name and password are set to `admin/admin` by default.



- Select **Actions > Restart All Required** to restart all cluster components that require a restart.



## 2.3. Enable the Ranger Plugin

The Ranger Atlas plugin enables you to establish and enforce global security policies based on data classifications. For more information, see [enabling the Ranger Atlas plugin](#) in Ambari.

## 2.4. Configure Atlas Tagsync in Ranger



### Note

Before configuring Atlas Tagsync in Ranger, you must enable Ranger Authorization in Atlas by [enabling the Ranger Atlas plug-in](#) in Ambari.

For information about configuring Atlas Tagsync in Ranger, see [Configure Ranger Tagsync](#).

## 2.5. Configure Atlas High Availability

For information about configuring High Availability (HA) for Apache Atlas, see [Apache Atlas High Availability](#).

## 2.6. Configuring Atlas Security

### 2.6.1. Additional Requirements for Atlas with Ranger and Kerberos

Currently additional configuration steps are required for Atlas with Ranger and in Kerberized environments.

#### 2.6.1.1. Additional Requirements for Atlas with Ranger

When Atlas is used with Ranger, perform the following additional configuration steps:



### Important

These steps are not required for Ambari-2.4.x and higher versions. For Ambari-2.4.x and higher, these steps will be performed automatically when Atlas is restarted.

- Create the following [HBase policy](#):
  - table: atlas\_titan, ATLAS\_ENTITY\_AUDIT\_EVENTS
  - user: atlas
  - permission: Read, Write, Create, Admin
- Create following [Kafka policies](#):

- topic=ATLAS\_HOOK  
permission=publish, create; group=public  
permission=consume, create; user=atlas (for non-kerberized environments, set group=public)
- topic=ATLAS\_ENTITIES  
permission=publish, create; user=atlas (for non-kerberized environments, set group=public)  
permission=consume, create; group=public

You should also ensure that an [Atlas service](#) is created in Ranger, and that the Atlas service includes the following configuration properties:

**Table 2.4. Ranger Atlas Service Kerberos Properties**

Property	Value
tag.download.auth.users	atlas
policy.download.auth.users	atlas
ambari.service.check.user	atlas

**Ranger** Access Manager Audit Settings

Service Manager > Edit Service

**Edit Service**

**Service Details :**

Service Name \*

Description

Active Status  Enabled  Disabled

Select Tag Service

**Config Properties :**

Username \*

Password \*

atlas.rest.address \*

Common Name for Certificate

Add New Configurations

Name	Value
tag.download.auth.users	atlas
policy.download.auth.users	atlas
ambari.service.check.user	atlas
<input type="text"/>	<input type="text"/>



## Note

If the Ranger Atlas service is not created after enabling the plugin and restarting Atlas, that indicates that either there is already a policy JSON on the Atlas host (in the `/etc/ranger/<service_name>/policycache/` directory), or Ambari was unable to connect to Ranger Admin during the Atlas restart. The solution for the first issue is to delete or move the `policycache` file, then restart Atlas.

- You can click the **Test Connection** button on the Ranger Atlas Service Details page to verify the configuration settings.
- You can also select **Audit > Plugins** in the Ranger Admin UI to check for the latest Atlas service entry.

Export Date ( IST ) *	Service Name	Plugin Id	Plugin IP	Cluster Name	HTTP Response Code	Status
06/23/2017 04:34:52 PM	cl_19_atlas	atlas@dk-rmp-8561-2-cl_19_atlas	172.22.104.72	cl_19	200	Policies synced to plugin

### 2.6.1.2. Additional Requirements for Atlas with Kerberos without Ranger

When Atlas is used in a Kerberized environment without Ranger, perform the following additional configuration steps:

- Start the HBase shell with the user identity of the HBase admin user ('hbase')
- Execute the following command in HBase shell, to enable Atlas to create necessary HBase tables:
  - `grant 'atlas', 'RWXCA'`
- Start (or restart) Atlas, so that Atlas would create above HBase tables
- Execute the following commands in HBase shell, to enable Atlas to access necessary HBase tables:
  - `grant 'atlas', 'RWXCA', 'atlas_titan'`
  - `grant 'atlas', 'RWXCA', 'ATLAS_ENTITY_AUDIT_EVENTS'`
- Kafka – To grant permissions to a Kafka topic, run the following commands as the Kafka user:

```
/usr/hdp/current/kafka-broker/bin/kafka-acls.sh --topic ATLAS_HOOK --allow-principals * --operations All --authorizer-properties "zookeeper.connect=hostname:2181"
/usr/hdp/current/kafka-broker/bin/kafka-acls.sh --topic ATLAS_ENTITIES --allow-principals * --operations All --authorizer-properties "zookeeper.connect=hostname:2181"
```

### 2.6.2. Enable Atlas HTTPS

For information about enabling HTTPS for Apache Atlas, see [Enable SSL for Apache Atlas](#).

### 2.6.3. Hive CLI Security

If you have Oozie, Storm, or Sqoop Atlas hooks enabled, the Hive CLI can be used with these components. You should be aware that the Hive CLI may not be secure without taking additional measures.

## 2.7. Installing Sample Atlas Metadata

You can use the `quick_start.py` Python script to install sample metadata to view in the Atlas web UI. Use the following steps to install the sample metadata:

1. Log in to the Atlas host server using a command prompt.
2. Run the following command as the Atlas user:

```
su atlas -c '/usr/hdp/current/atlas-server/bin/quick_start.py'
```



### Note

In an SSL-enabled environment, run this command as:

```
su atlas -c '/usr/hdp/current/atlas-server/bin/quick_start.py  
https://<fqdn_atlas_host>:21443'
```

When prompted, type in the Atlas user name and password. When the script finishes running, the following confirmation message appears:

```
Example data added to Apache Atlas Server!!!
```

If Kerberos is enabled, `kinit` is required to execute the `quick_start.py` script.

After you have installed the sample metadata, you can explore the Atlas web UI.



### Note

If you are using the HDP Sandbox, you do not need to run the Python script to populate Atlas with sample metadata.

## 2.8. Updating the Atlas Ambari Configuration

When you update the Atlas configuration settings in Ambari, Ambari marks the services that require restart, and you can select **Actions > Restart All Required** to restart all services that require a restart.



### Important

Apache Oozie requires a restart after an Atlas configuration update, but may not be included in the services marked as requiring restart in Ambari. Select **Oozie > Service Actions > Restart All** to restart Oozie along with the other services.

## 3. Searching and Viewing Entities

### 3.1. Using Basic and Advanced Search

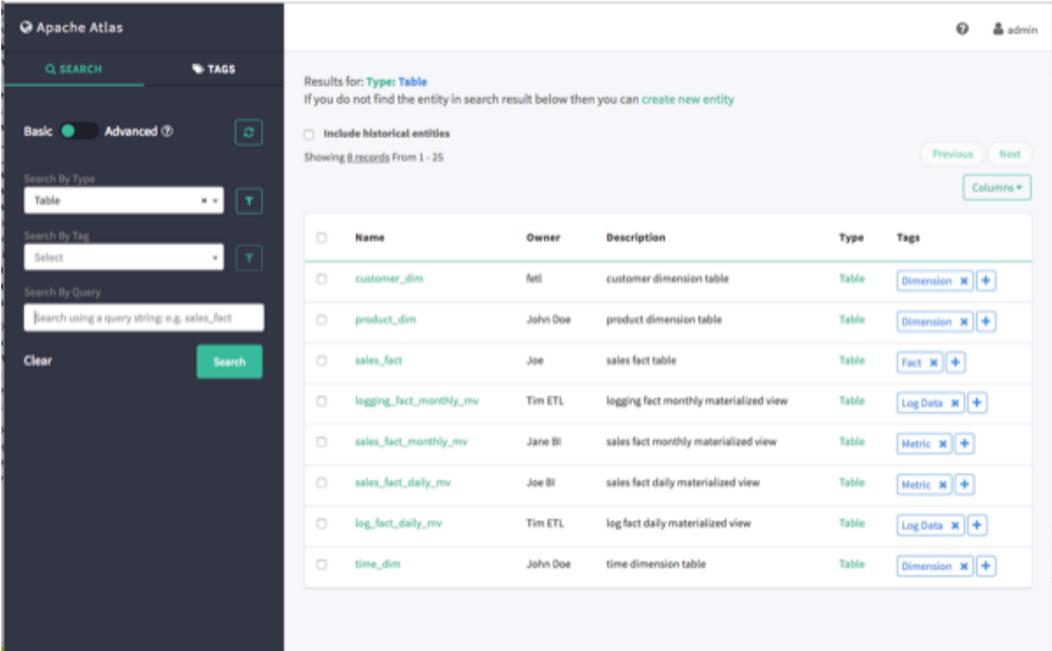
#### 3.1.1. Using Basic Search

You can search for entities using three basic search modes:

- Search by Type – search based on a selected Entity type.
- Search by Tag – search based on a selected Atlas tag.
- Search by Query – full-text search.

1. To search for entities, click **SEARCH** on the Atlas web UI. Select an entity type, an Atlas tag, or enter a text string, then click **Search** to display a list of the entities associated with the specified search criteria.

- In the example below, we searched for the Table entity type.



The screenshot shows the Apache Atlas search interface. On the left, the search sidebar is visible with the following options:

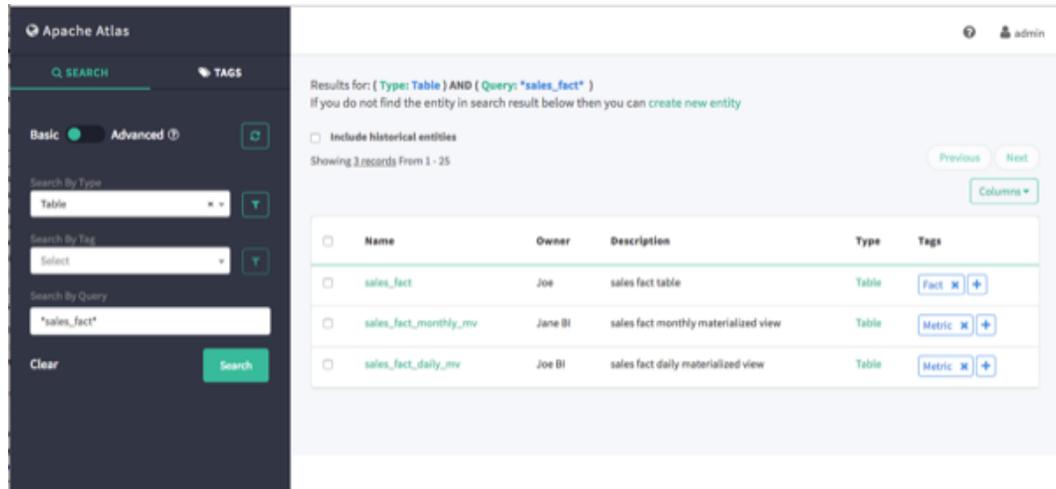
- Search By Type: Table
- Search By Tag: Select
- Search By Query: Search using a query string: e.g. sales\_fact
- Buttons: Clear, Search

The main search results area displays the following information:

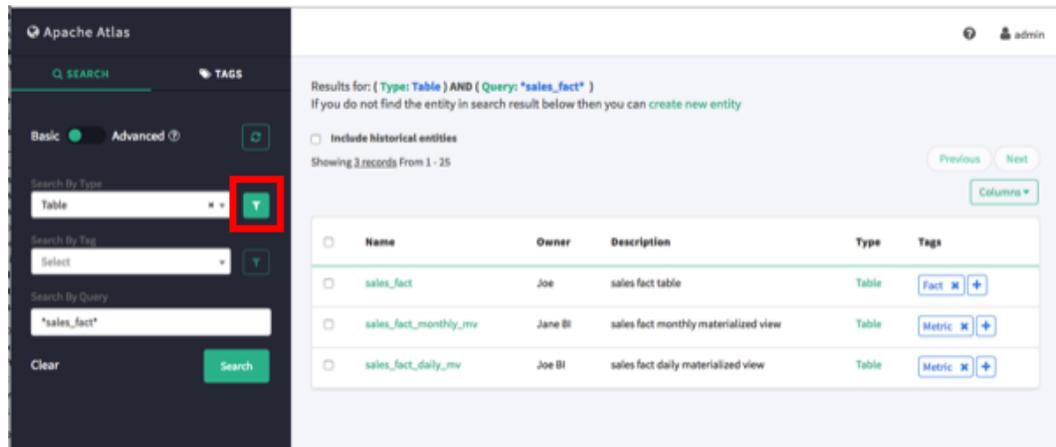
- Results for: Type: Table
- If you do not find the entity in search result below then you can create new entity
- Include historical entities
- Showing 8 records From 1 - 25
- Navigation: Previous, Next, Columns

Name	Owner	Description	Type	Tags
customer_dim	fedl	customer dimension table	Table	Dimension
product_dim	John Doe	product dimension table	Table	Dimension
sales_fact	Joe	sales fact table	Table	Fact
logging_fact_monthly_mv	Tim ETL	logging fact monthly materialized view	Table	Log Data
sales_fact_monthly_mv	Jane BI	sales fact monthly materialized view	Table	Metric
sales_fact_daily_mv	Joe BI	sales fact daily materialized view	Table	Metric
log_fact_daily_mv	Tim ETL	log fact daily materialized view	Table	Log Data
time_dim	John Doe	time dimension table	Table	Dimension

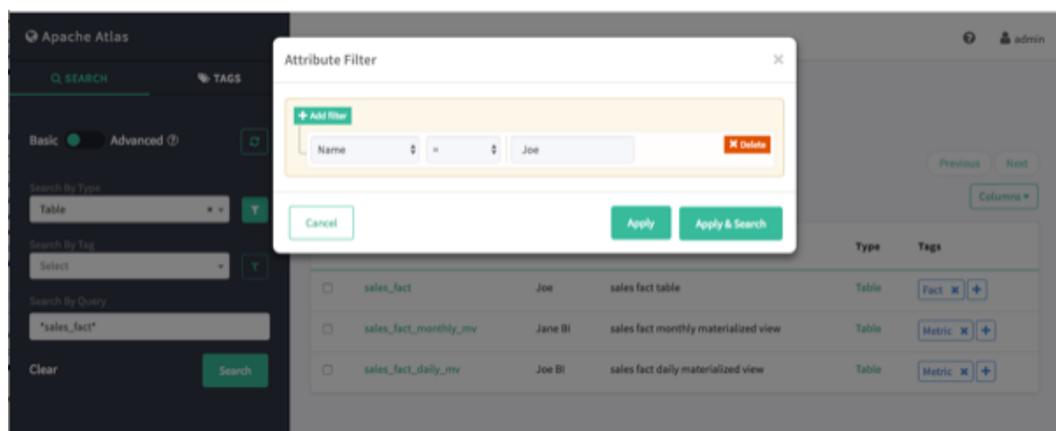
- You can also combine search criteria. In the example below, we combined Type and full-text search to find Table entities whose name contains the text string "sales\_fact".



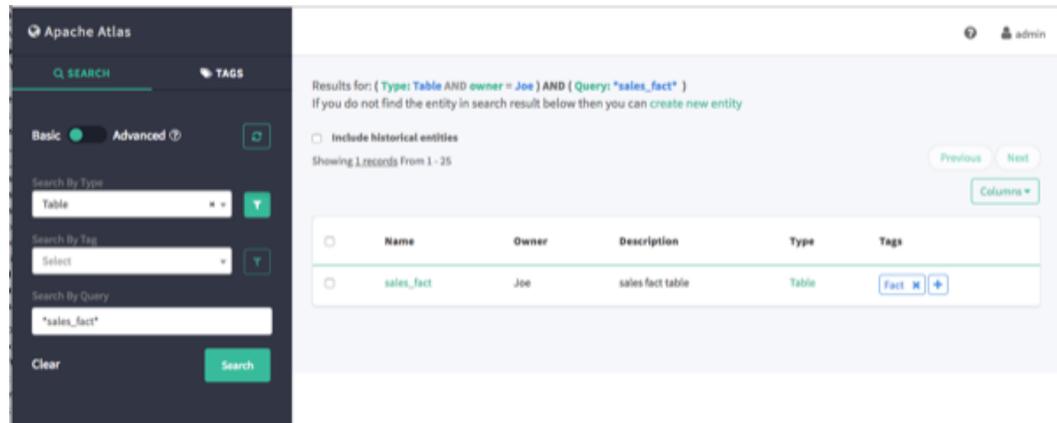
- You can use the attribute filters to further refine search criteria. Click an Attribute Filter symbol to display the Attribute Filter pop-up.



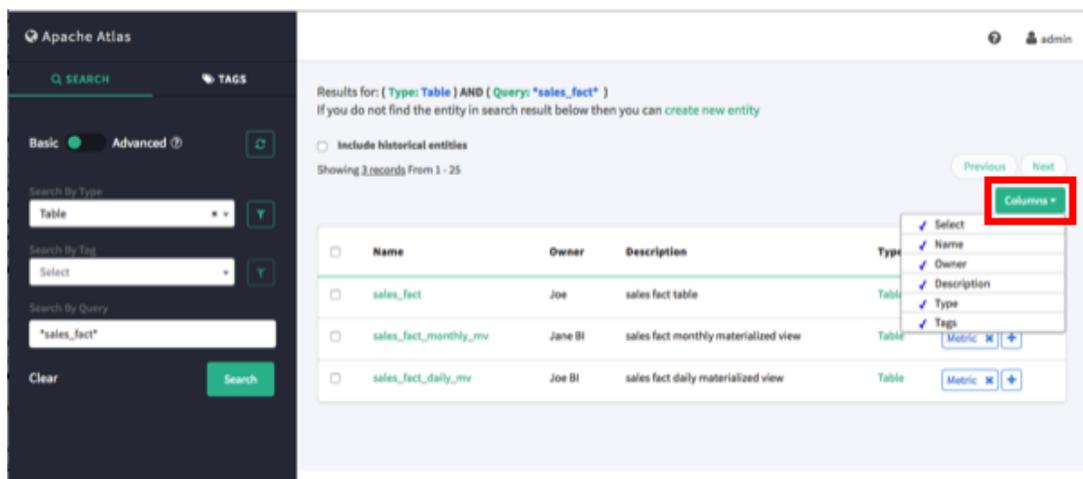
Use the selection boxes on the Attribute Filter pop-up to specify an attribute filter. The attributes listed reflect the entity type. In the following example, we set an attribute filter to return entities with an Owner attribute of "Joe".



- Click **Add filter** to add more attribute filters.
- Click **Delete** to remove an attribute filter.
- Click **Apply & Search** to immediately apply the attribute filter to the search results.
- Click **Apply** to temporarily save the attribute filter (to the current search) without applying it to the search results. Clicking the main **Search** button applies the attribute filter to the search results.



2. Click **Columns** to control which columns are displayed in the list of search results.



3. To view detailed information about an entity, click the entity in the search results list. In the example below, we selected the "sales\_fact" table from the list of search results.

The screenshot shows the Apache Atlas search results page for the entity 'sales\_fact (Table)'. On the left, there is a search sidebar with filters for 'Search By Type' (set to 'Table'), 'Search By Tag' (set to 'Select'), and 'Search By Query'. A 'Clear' button is visible at the bottom of the sidebar. The main content area displays the entity name 'sales\_fact (Table)' and a 'Tags' section with 'Fact' selected. Below this, a 'LINEAGE & IMPACT' section shows a flow diagram with five nodes: 'sales\_fact' (red), 'loadSalesOnly' (blue), 'sales\_fact\_daily...' (green), 'loadSalesMonthly' (blue), and 'sales\_fact\_monthly...' (green). Arrows indicate the flow between these nodes. A legend at the bottom of the diagram shows a green arrow for 'Lineage' and a red arrow for 'Impact'.

4. Click **Clear** to clear the search settings.

The screenshot shows the Apache Atlas search results page for 'Basic Search'. The search sidebar on the left has 'Search By Type' and 'Search By Tag' both set to 'Select'. The 'Clear' button at the bottom of the sidebar is highlighted with a red box. The main content area displays the title 'Basic Search' and the instruction 'Search Atlas for existing entities or create new entity'.

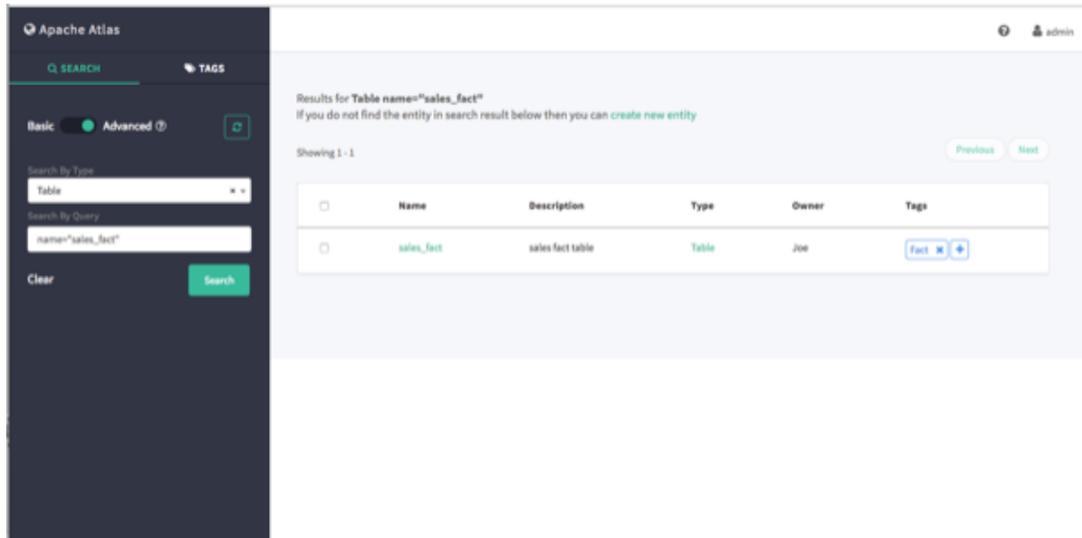
### 3.1.2. Using Advanced Search

To switch to Advanced search mode, slide the green toggle button from **Basic** to **Advanced**. You can search for entities using two advanced search modes:

- Search by Type – search based on a selected Entity type.
- Search by Query – search using an [Apache Atlas DSL](#) query. Atlas DSL (Domain-Specific Language) is a SQL-like query language that enables you to search metadata using complex queries.

1. To search for entities, select an entity type or enter an Atlas DSL search query, then click **Search** to display a list of the entities associated with the specified search criteria.

You can also combine search criteria. In the example below, we searched for Table entity types named "sales\_fact".



The screenshot shows the Apache Atlas search interface. On the left, the search sidebar is visible with the following details:

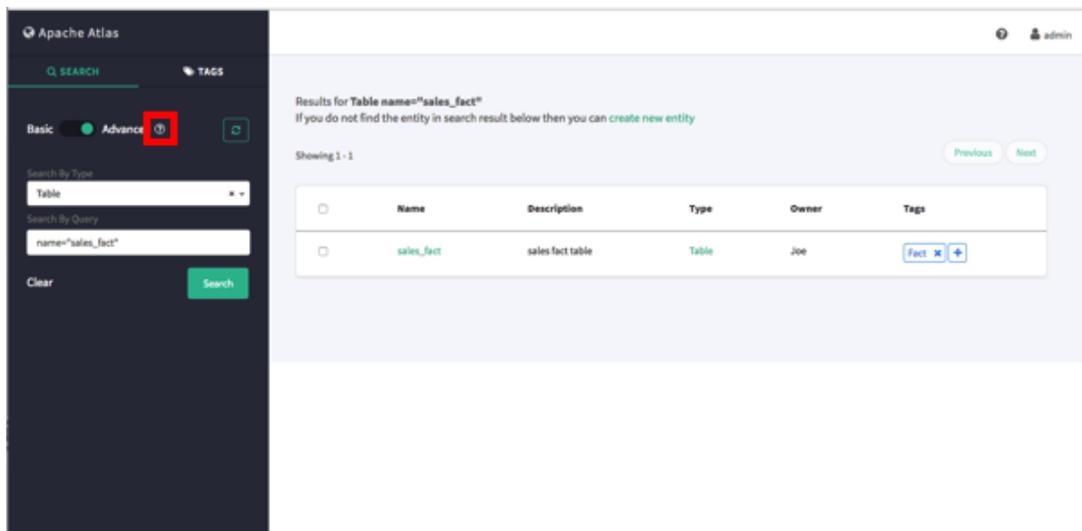
- Search By Type: Table
- Search By Query: name="sales\_fact"
- Search mode: Basic (selected), Advanced
- Buttons: Clear, Search

The main search results area displays the following information:

- Results for Table name="sales\_fact"
- Message: If you do not find the entity in search result below then you can [create new entity](#)
- Showing 1 - 1
- Navigation: Previous, Next
- Table with 6 columns: Name, Description, Type, Owner, Tags

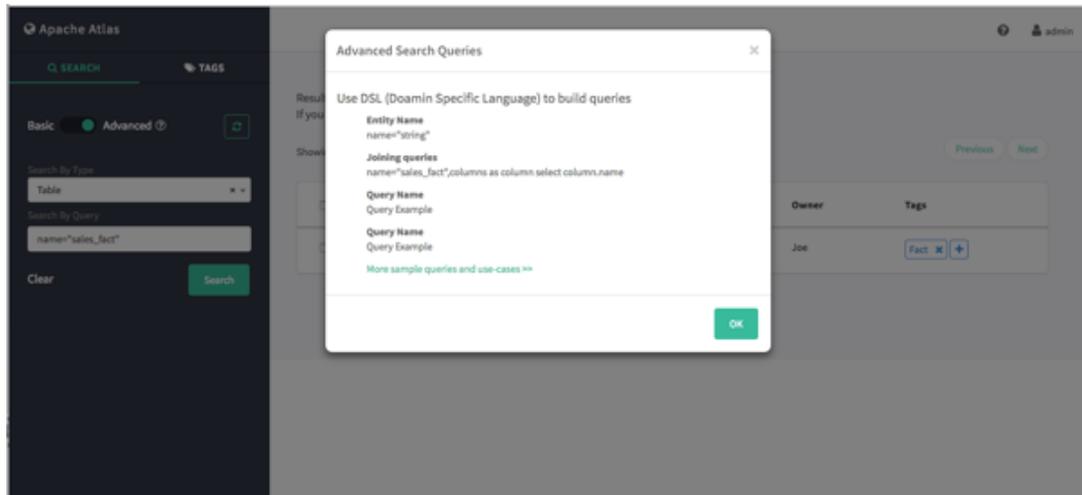
Name	Description	Type	Owner	Tags
sales_fact	sales fact table	Table	Joe	Fact

To display more information about Atlas DSL queries, click the question mark symbol next to the **Advanced** label above the search boxes.

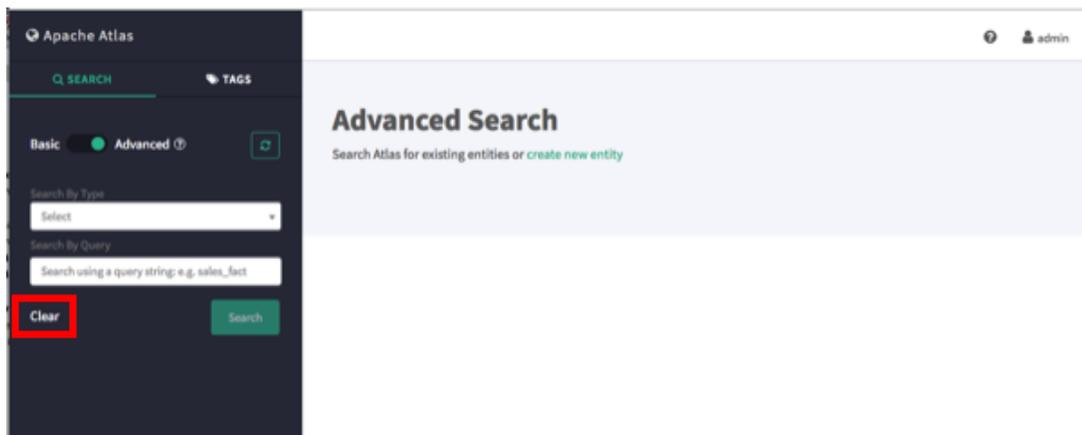


This screenshot is identical to the previous one, but with a red box highlighting the question mark icon next to the 'Advanced' search mode label in the sidebar.

The Advanced Search Queries lists example queries, along with a link to the Apache Atlas DSL query documentation:



2. Click **Clear** to clear the search settings.



## 3.2. Viewing Entity Data Lineage & Impact

1. Data lineage and impact is displayed when you select an entity. In the following example, we ran a Type search for `Table`, and then selected the "sales\_fact" entity. Data lineage and impact is displayed graphically, with each icon representing an action. You can use the + and - buttons to zoom in and out, and you can also click and drag to move the image.

The screenshot shows the Apache Atlas web interface for the 'sales\_fact' table. On the left is a search sidebar with filters for 'Table' and 'Fact'. The main content area is titled 'sales\_fact (Table)' and shows a lineage diagram with nodes for 'sales\_fact', 'loadSalesDaily', 'sales\_fact\_daily...', 'loadSalesMonthly', and 'sales\_fact\_month...'. Below the diagram is a 'DETAILS' section with tabs for 'PROPERTIES', 'TAGS', 'AUDITS', and 'SCHEMA'. The 'PROPERTIES' tab is active, displaying a table with the following data:

Key	Value
columns	time_id product_id customer_id sales
createTime	Mon Apr 24 2017 14:25:50 GMT-0400 (EDT)
db	Sales

2. Moving the cursor over an icon displays a pop-up with more information about the action that was performed. In the following example, we can see that a query was used to create the "loadSalesDaily" table from the "sales\_fact" table.

The screenshot displays the Apache Atlas web interface for the 'sales\_fact' table. The left sidebar contains search filters for 'Table' type and 'fact' tag. The main content area shows the 'LINEAGE & IMPACT' diagram, which illustrates the data flow from 'sales\_fact' through 'loadSalesDaily' and 'loadSalesMonthly' processes to 'sales\_fact\_daily...' and 'sales\_fact\_monthly...' tables. Below this, the 'DETAILS' section is active, showing the 'PROPERTIES' tab with a table of key-value pairs.

Key	Value
columns	time_id product_id customer_id sales
createTime	Mon Apr 24 2017 14:25:50 GMT-0400 (EDT)

### 3.3. Viewing Entity Details

When you select an entity, detailed information about the entity is displayed under DETAILS.

- The Properties tab displays all of the entity properties.

The screenshot shows the Apache Atlas web interface. On the left is a dark sidebar with search filters. The main area is titled 'DETAILS' and has tabs for 'PROPERTIES', 'TAGS', 'AUDITS', and 'SCHEMA'. The 'TAGS' tab is selected, showing a table of properties for a table entity.

Key	Value
columns	time_id product_id customer_id sales
createTime	Mon Apr 24 2017 14:25:50 GMT-0400 (EDT)
db	Sales
description	sales fact table
lastAccessTime	Mon Apr 24 2017 14:25:50 GMT-0400 (EDT)
name	sales_fact
owner	Joe
qualifiedName	sales_fact
retention	Mon Apr 24 2017 14:25:50 GMT-0400 (EDT)
sd	9686cb0e-663d-4273-9c00-025647135211
tableType	Managed
temporary	false
viewExpandedText	
viewOriginalText	

- Click the Tags tab to display the tags associated with the entity. In this case, the "fact" tag has been associated with the "sales\_fact" table.

The screenshot displays the Apache Atlas web interface for the 'sales\_fact' table. On the left is a dark sidebar with search filters. The main content area is titled 'sales\_fact (Table)' and includes a 'TAGS' section with a 'Fact' tag. Below this is the 'LINEAGE & IMPACT' section, which contains a flow diagram showing data lineage from 'sales\_fact' through various processes like 'loadSalesDaily' and 'loadSalesMonthly' to other tables. A legend indicates that green arrows represent lineage and red arrows represent impact. At the bottom is the 'DETAILS' section with tabs for 'PROPERTIES', 'TAGS', 'AUDITS', and 'SCHEMA'. The 'TAGS' tab is active, showing a table with one entry: 'Fact' with 'NA' attributes and a 'Tool' icon.

Tags	Attributes	Tool
Fact	NA	

- The Audits tab provides a complete audit trail of all events in the entity history. You can use the Detail button next to each action to view more details about the event.

The screenshot displays the Apache Atlas web interface for the 'sales\_fact' table. On the left is a dark sidebar with search filters. The main content area is titled 'sales\_fact (Table)' and includes a 'LINEAGE & IMPACT' diagram and a 'DETAILS' section with tabs for 'PROPERTIES', 'TAGS', 'AUDITS', and 'SCHEMA'. The 'AUDITS' tab is active, showing a table with one entry.

**LINEAGE & IMPACT**

Diagram showing data flow: sales\_fact (red) → loadSalesDaily (blue) → sales\_fact\_daily... (green) → loadSalesMonthly (blue) → sales\_fact\_monthly... (green). Legend: → Lineage, → Impact.

**DETAILS**

Showing 1 - 1

Users	Timestamp	Actions	Tools
admin	Mon Apr 24 2017 14:25:50 GMT-0400 (EDT)	Entity Created	<a href="#">Detail</a>

- The Schema tab shows schema information, in this case the columns for the table. We can also see that a PII tag has been associated with the "customer\_id" column.

The screenshot displays the Apache Atlas web interface. On the left is a dark sidebar with a search bar and filters. The main content area is divided into two sections: 'LINEAGE & IMPACT' and 'DETAILS'. The 'LINEAGE & IMPACT' section shows a flow diagram with nodes representing tables and arrows representing data lineage. The 'DETAILS' section shows a table with columns for Name, Comment, and Tags, displaying details for a specific entity.

**Apache Atlas**

Q SEARCH TAGS

Basic  Advanced

Search By Type  
Table

Search By Tag  
Select

Search By Query  
Search using a query string: e.g. sales\_fact

Clear Search

**LINEAGE & IMPACT**

sales\_fact loadSalesDaily sales\_fact\_daily... loadSalesMonthly sales\_fact\_monthly...

→ Lineage → Impact

**DETAILS**

PROPERTIES TAGS AUDITS SCHEMA

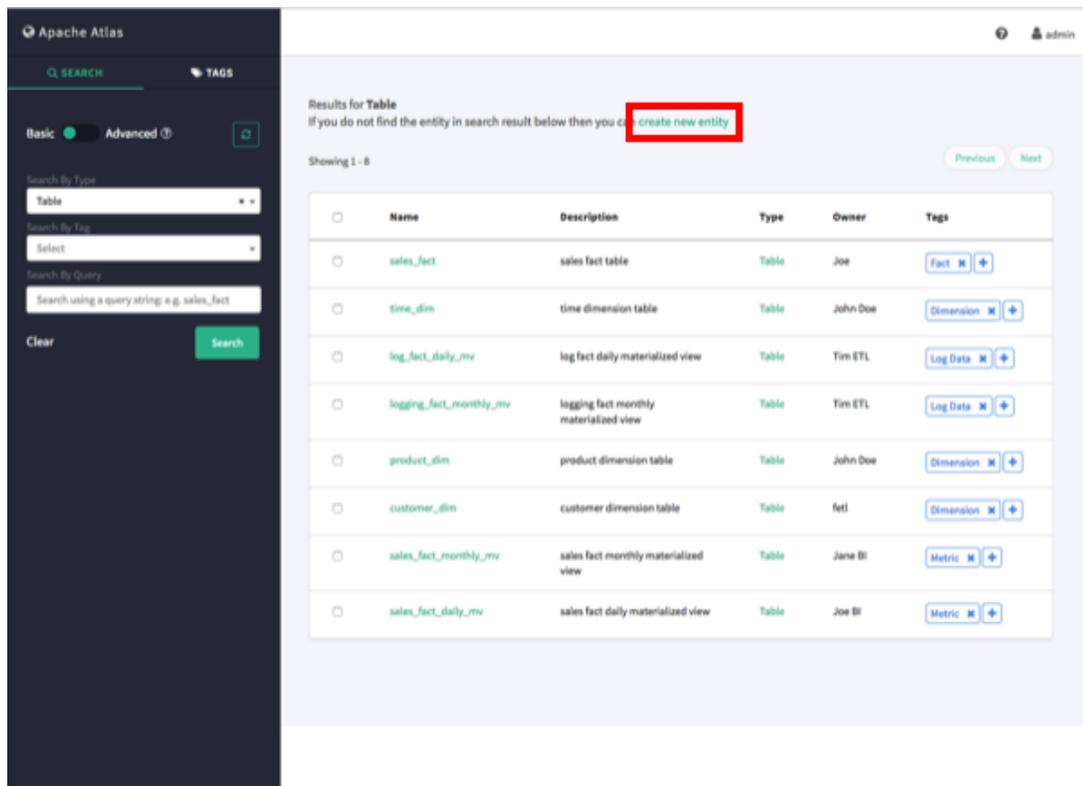
Showing 1 - 4

	Name	Comment	Tags
<input type="checkbox"/>	time_id	time id	<input type="button" value="+"/>
<input type="checkbox"/>	product_id	product id	<input type="button" value="+"/>
<input type="checkbox"/>	customer_id	customer id	<input type="button" value="PI"/> <input type="button" value="+"/>
<input type="checkbox"/>	sales	product id	<input type="button" value="Metric"/> <input type="button" value="+"/>

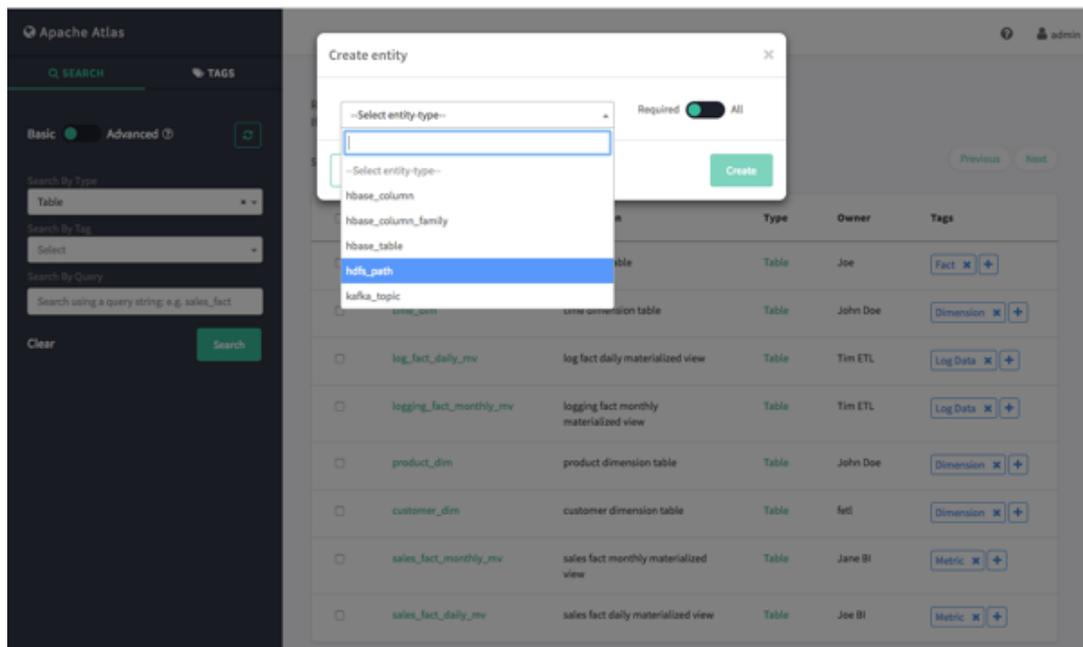
## 3.4. Manually Creating Entities

Currently there is no Atlas hook for HBase, HDFS, or Kafka. For these components, you must manually create entities in Atlas. You can then associate tags with these entities and control access using Ranger tag-based policies.

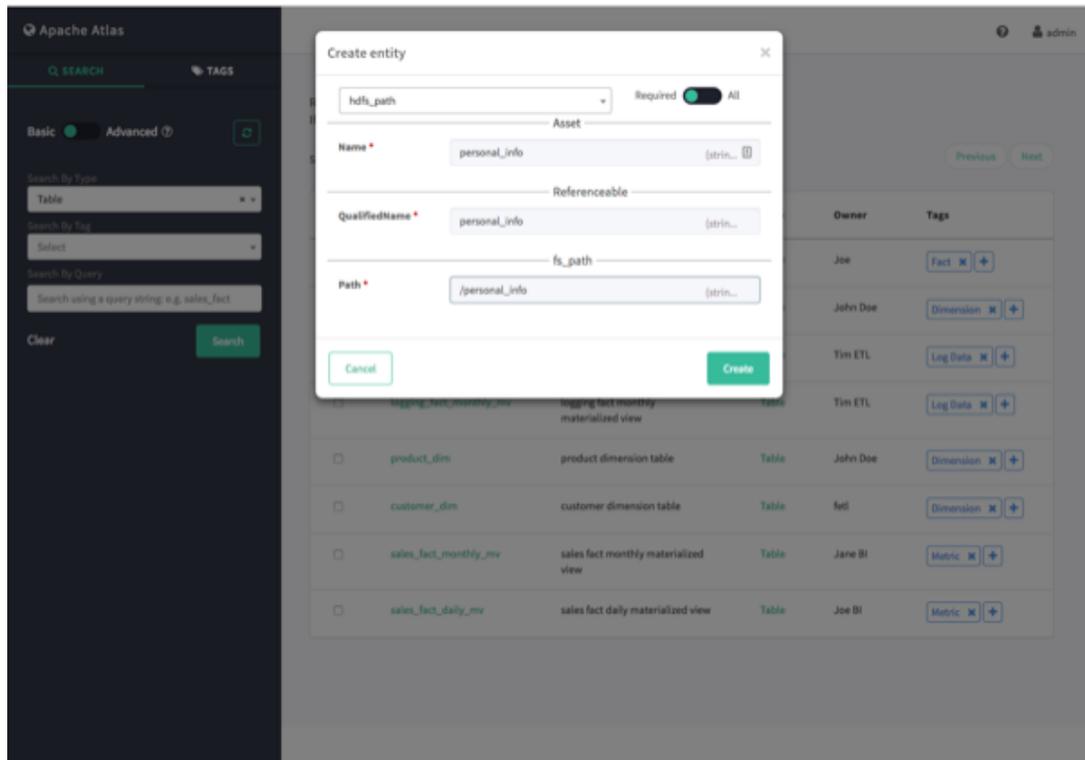
1. On the Atlas web UI Search page, click the **create new entity** link at the top of the page.



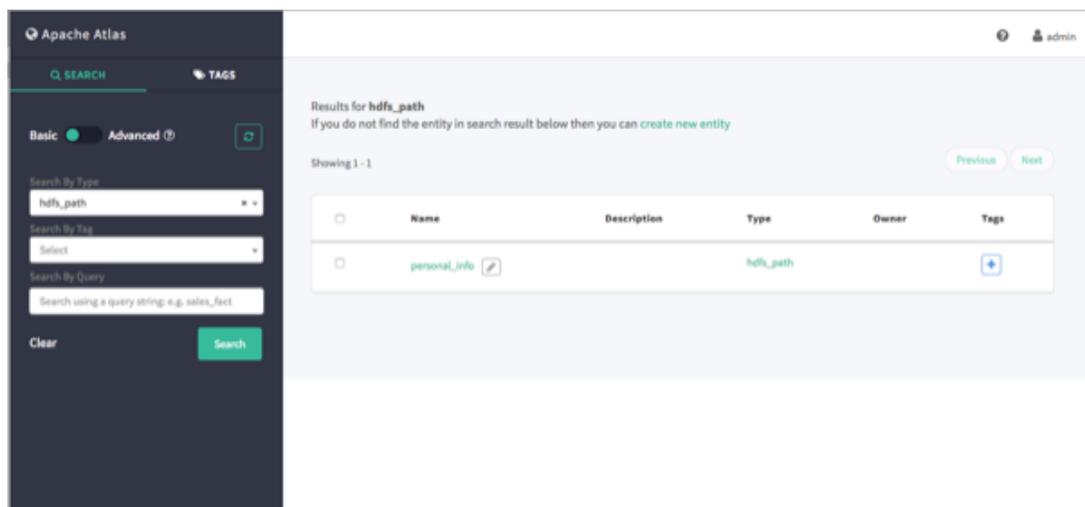
2. On the Create Entity pop-up, select an entity type.



3. Enter the required information for the new entity. Click **All** to display both required and non-required information. Click **Create** to create the new entity.



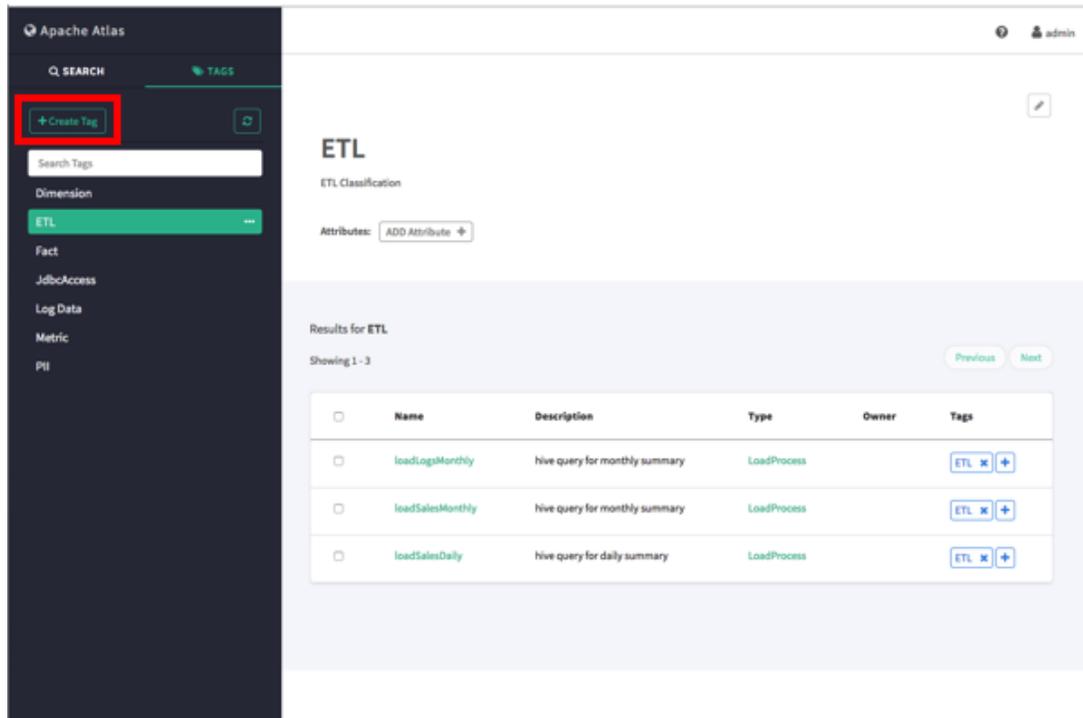
- The entity is created and returned in search results for the applicable entity type. You can now associate tags with the new entity and control access to the entity with Ranger tag-based policies.



## 4. Working with Atlas Tags

### 4.1. Creating Atlas Tags

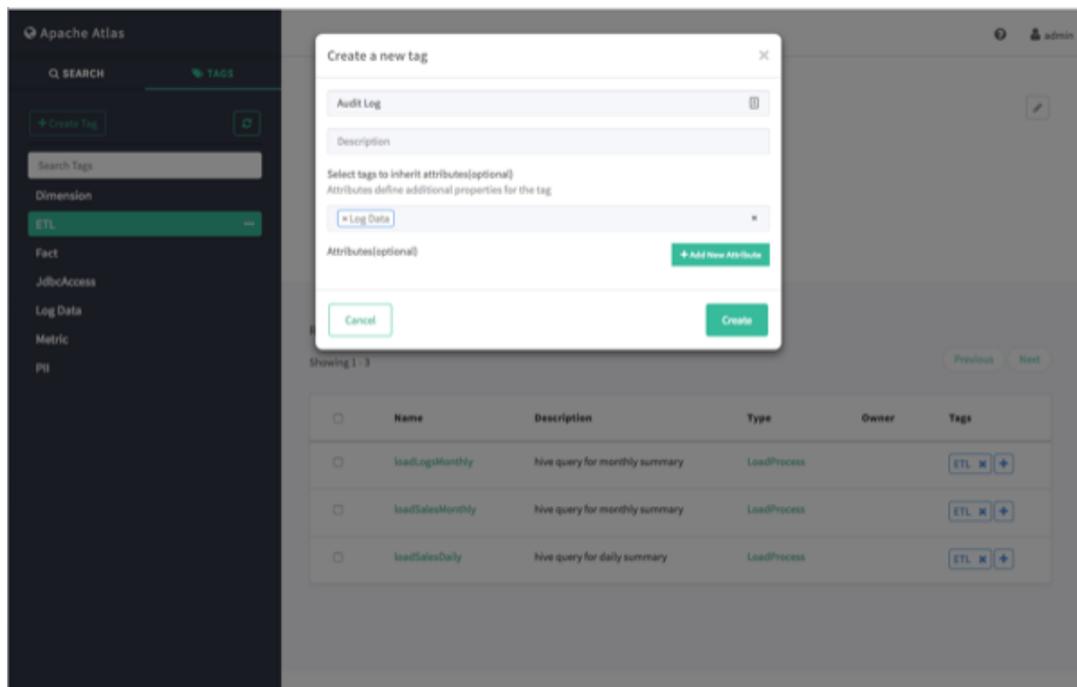
1. On the Atlas web UI, click **TAGS**, then click **Create Tag**.



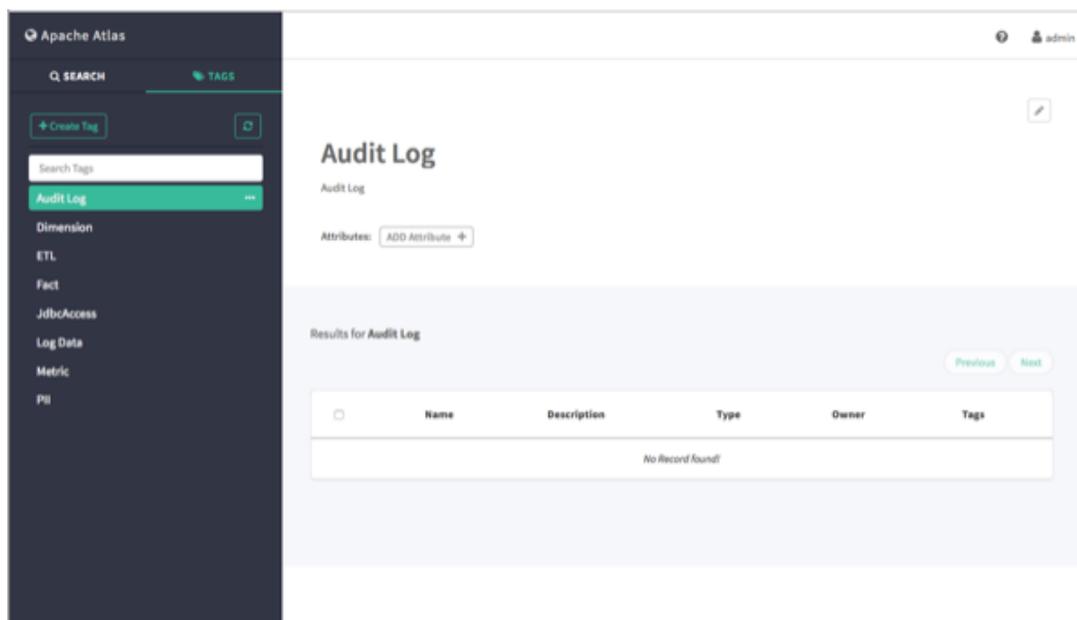
The screenshot shows the Apache Atlas web interface. On the left, a dark sidebar contains a 'TAGS' section with a '+ Create Tag' button highlighted in red. Below it is a search bar and a list of tag categories: Dimension, ETL (highlighted in green), Fact, JDBCAccess, Log Data, Metric, and PII. The main content area is titled 'ETL' and shows 'ETL Classification' with an 'ADD Attribute +' button. Below this, a section titled 'Results for ETL' shows 'Showing 1 - 3' items. A table lists three tags:

<input type="checkbox"/>	Name	Description	Type	Owner	Tags
<input type="checkbox"/>	loadLogsMonthly	hive query for monthly summary	LoadProcess		ETL ✕ +
<input type="checkbox"/>	loadSalesMonthly	hive query for monthly summary	LoadProcess		ETL ✕ +
<input type="checkbox"/>	loadSalesDaily	hive query for daily summary	LoadProcess		ETL ✕ +

2. On the Create a New Tag pop-up, type in a name and an optional description for the tag. You can use the **Select tags to inherit attributes** box to inherit attributes from other tags. Click **Add New Attribute** to add one or more new attributes to the tag. Click **Create** to create the new Tag.

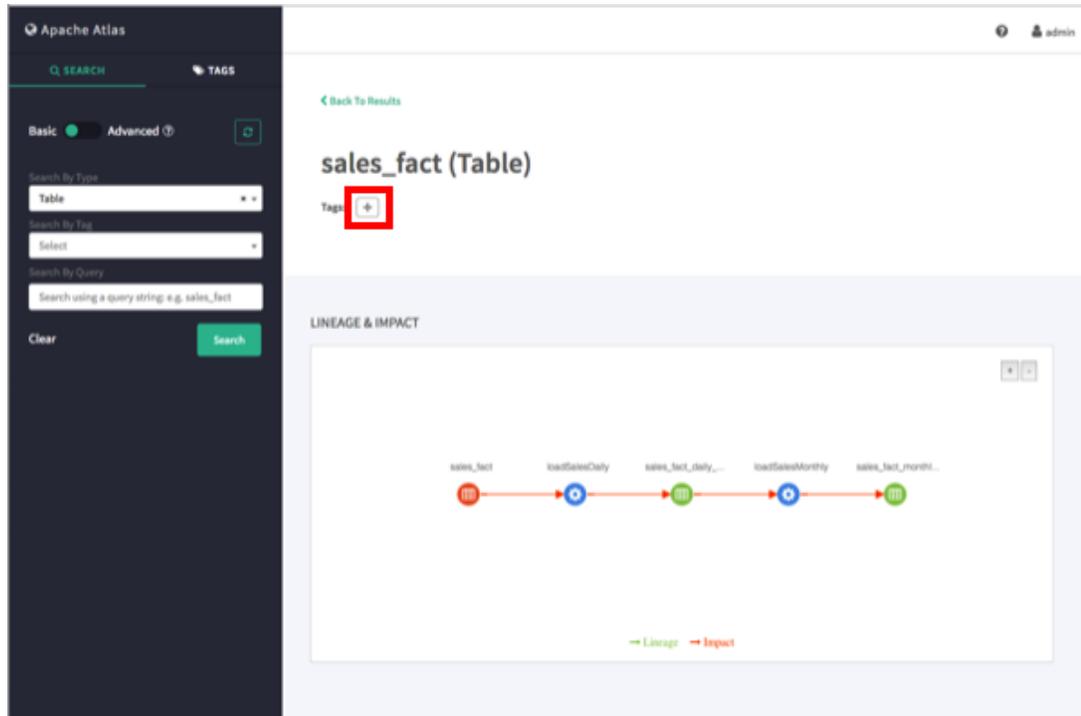


3. The new tag appears in the Tags list.



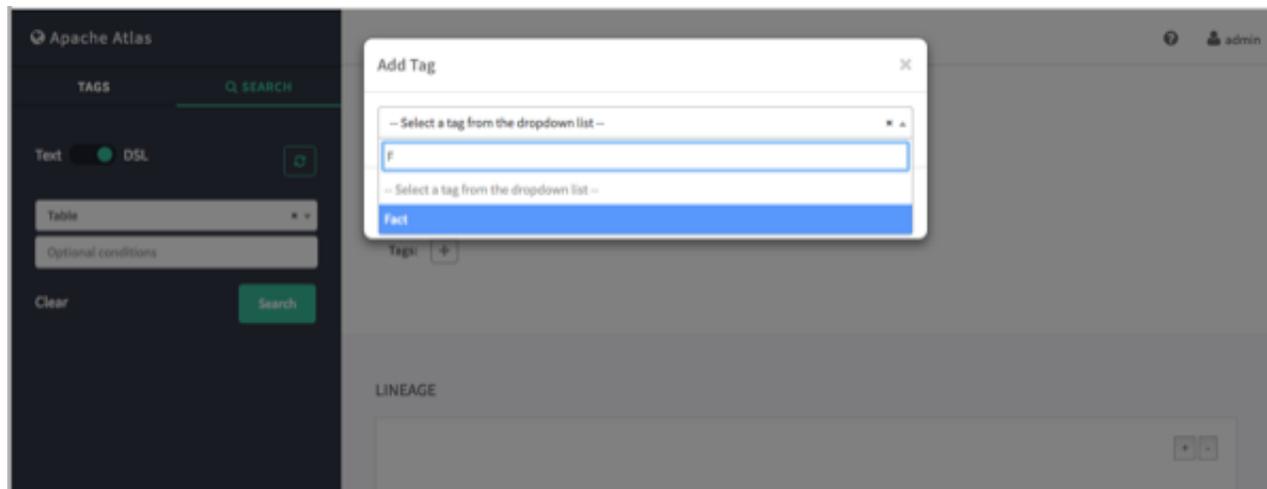
## 4.2. Associating Tags with Entities

1. Select an asset. In the example below, we searched for all `Table` entities, and then selected the "sales\_fact" table from the list of search results. To associate a tag with an asset, click the + icon next to the **Tags:** label.



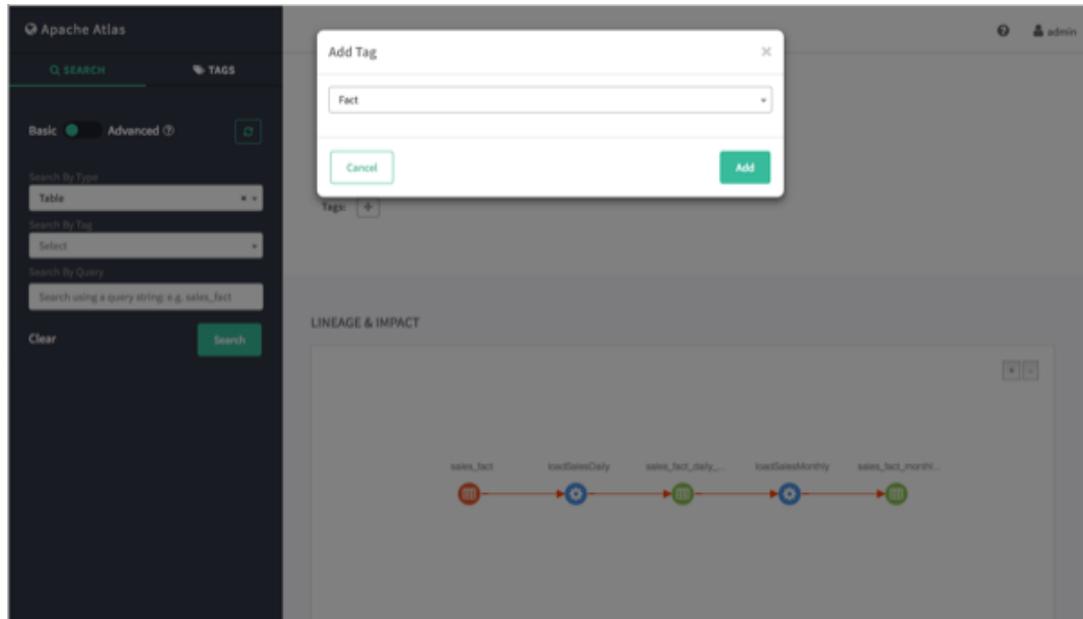
The screenshot shows the Apache Atlas interface. On the left is a search sidebar with options for 'Basic' and 'Advanced' search, and filters for 'Table' and 'Select'. The main content area displays the asset 'sales\_fact (Table)' with a 'Tags' section containing a red-bordered '+ Add Tag' button. Below this is a 'LINEAGE & IMPACT' diagram showing a flow of data from 'sales\_fact' through various intermediate tables like 'loadfactDaily' and 'sales\_fact\_daily...' to 'sales\_fact\_monthly...'.

2. On the Add Tag pop-up, click **Select Tag**, then select the tag you would like to associate with the asset. You can filter the list of tags by typing text in the Select Tag box.

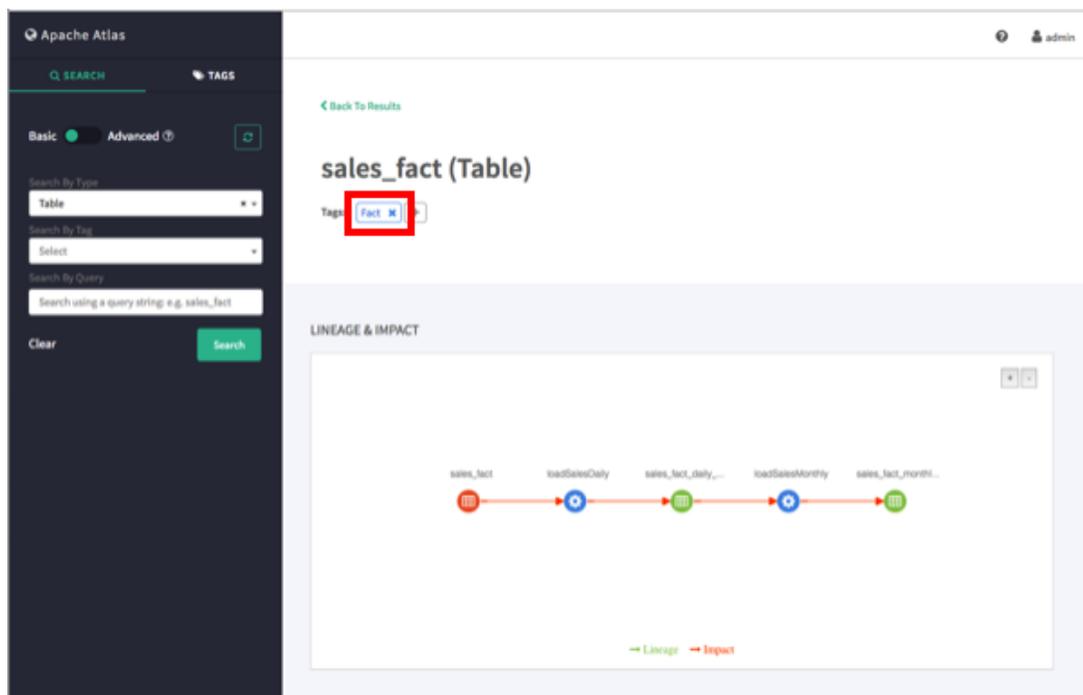


The screenshot shows the 'Add Tag' pop-up dialog box overlaid on the Apache Atlas interface. The dialog has a search input field with the letter 'F' entered. Below the input field, a dropdown list is open, showing 'Fact' as the selected option. The background shows the 'LINEAGE' section of the asset page.

3. After you select a tag, the Add Tag pop-up is redisplayed with the selected tag. Click **Add** to associate the tag with the asset.

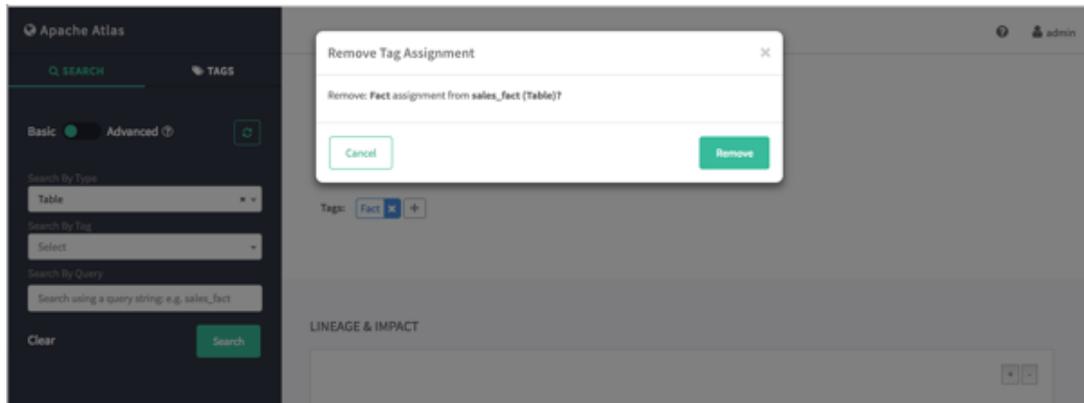


4. The new tag is displayed next to the **Tags**: label on the asset page.



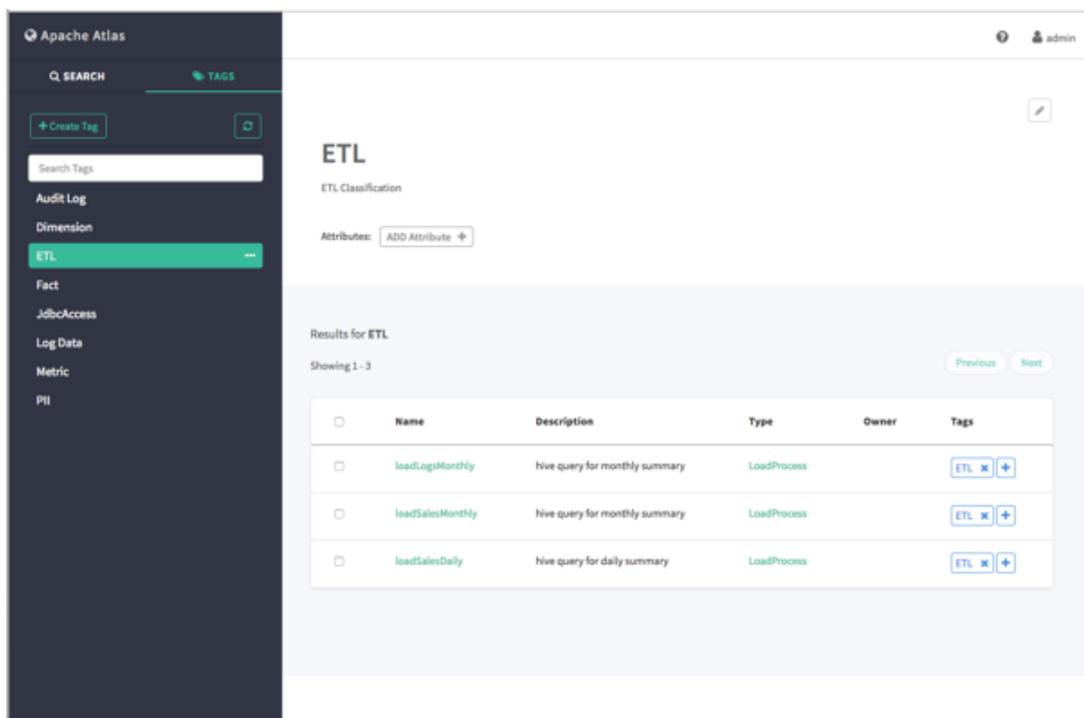
5. You can view details about a tag by clicking the tag name on the tag label.

To remove a tag from an asset, click the x symbol on the tag label, then click **Remove** on the confirmation pop-up. This removes the tag association with the asset, but does not delete the tag itself.

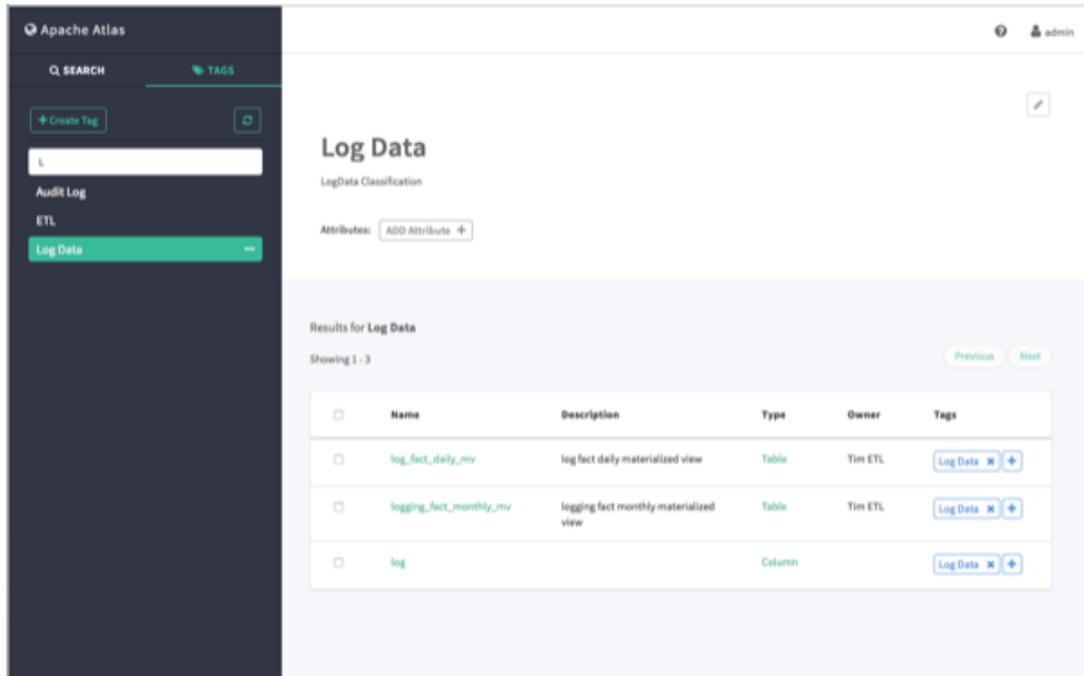


## 4.3. Searching for Entities Associated with Tags

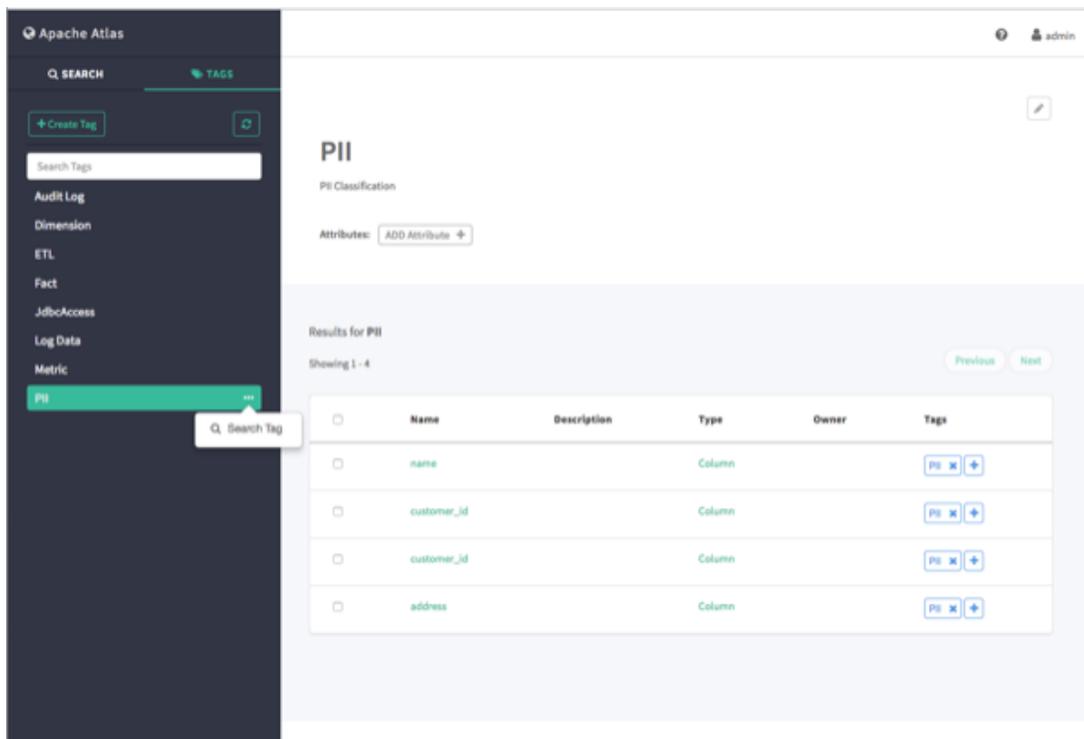
1. To display a list of all of the entities associated with a tag, click the tag name in the Atlas Tags list.



2. To filter the Tags list based on a text string, type the text in the Search Tags box. The list is filtered dynamically as you type to display the tags that contain that text string. You can then click a tag in the filtered list to display the entities associated with that tag.



- You can also search for entities associated with a tag by clicking the ellipsis symbol for the tag and selecting **Search Tag**. This launches a DSL search query that returns a list of all entities associated with the tag.



## 5. Apache Atlas REST API

Apache Atlas exposes a variety of REST endpoints that enable you to work with types, entities, lineage, and data discovery. The following resources provide detailed information about the Apache Atlas REST API:

- [Apache Atlas REST API](#)
- [Apache Atlas Swagger](#) interactive Atlas REST API interface