

Installing Apache Atlas

Date of Publish: 2019-12-17



Contents

Migrating Atlas metadata when upgrading to HDP-3.0+.....	3
Overview.....	3
Migrate Atlas metadata when upgrading to HDP-3.0+.....	3
Installing Atlas.....	4
Start the installation.....	4
Customize services.....	8
Authentication settings.....	8
Authorization settings.....	11
Dependent configurations.....	13
Configure identities.....	13
Complete the Atlas installation.....	14
Install sample Atlas metadata.....	16

Migrating Atlas metadata when upgrading to HDP-3.0+

When upgrading to HDP-3.0 and higher versions, you must perform additional steps to migrate the Atlas metadata from Titan to JanusGraph.

Overview

In HDP-3.0+, Apache Atlas uses the JanusGraph graph database to store metadata objects. In earlier versions, Atlas used the Titan graph database for metadata storage. When upgrading to HDP-3.0 and higher versions, you must use a command-line migration tool to migrate the Atlas metadata from Titan to JanusGraph.

1. Estimate the size of the existing Atlas repository and the time it will take to export the metadata from HDP 2.x Titan database to the HDP 3.x JanusGraph database.
2. Use the Atlas metadata migration tool to export the Atlas metadata from HDP 2.x.
3. Import the Atlas metadata into HDP 3.x.

Migrate Atlas metadata when upgrading to HDP-3.0+

Perform the following steps to migrate the Atlas metadata from Titan to JanusGraph when upgrading from HDP-2.x to HDP-3.0 and higher versions.

Procedure

1. Before upgrading HDP and Atlas, use one of the following methods to determine the size of the Atlas metadata on the HDP-2.x cluster.
 - Click SEARCH on the Atlas web UI, then slide the green toggle button from Basic to Advanced. Enter the following query in the Search by Query box, then click Search.

```
Asset select count()
```

- Run the following Atlas metrics REST API query:

```
curl -g -X GET -u admin:admin -H "Content-Type: application/json" /  
-H "Cache-Control: no-cache" "http://<atlas_server>:21000/api/atlas/  
admin/metrics"
```

Either of these methods returns the number of Atlas entities, which can be used to estimate the time required to export the Atlas metadata from HDP-2.x and import it into HDP-3.x. This time varies depending on the cluster configuration. The following estimates are for a node with a 4 GB RAM quad-core processor with both the Atlas and Solr servers on the same node:

- Estimated duration for export from HDP-2.x: 2 million entities per hour.
- Estimated duration for import into HDP-3.x: 0.75 million entities per hour.

The Atlas' migration exporter utility is used for migrating Atlas from HDP 2.x to HDP 3.x and beyond. Download from the location: <https://archive.cloudera.com/am2cm/hdp2/atlas-migration-exporter-0.8.0.2.6.6.0-332.tar.gz>

2. Before upgrading HDP and Atlas, perform the following steps on the HDP-2.x cluster.
 - a. Replace contents of `/usr/hdp/2.6.<current version>/atlas/tools/migration-exporter/`
 - b. Modify the permissions using `chown -R atlas:atlas <directory above>`
 - c. Execute the tool from location above. `atlas_migration.py -d <output directory>`
 - d. On the Ambari dashboard, click Atlas, then select Actions > Stop.

- e. Use the HDP-2.6. exporter tool to run the export. Typically the tool is located at `/usr/hdp/2.6.<current version>/atlas/tools/migration-exporter/`. Use the following command format to start the exporting the Atlas metadata:

```
python /usr/hdp/2.6.<current version>/atlas/tools/migration-exporter/
atlas_migration.py -d <output directory>
```

While running, the Atlas migration tool prevents Atlas use, and blocks all REST APIs and Atlas hook notification processing.

As described previously, the time it takes to export the Atlas metadata depends on the number of entities and your cluster configuration. You can use the following command to display the export status:

```
tail -f /var/log/atlas/atlas-migration-exporter.log
```

When the export is complete, the data is placed in the specified output directory.

- f. On the Ambari dashboard, Select Atlas > Configs > Advanced > Custom application-properties. Click Add Property, then add an `atlas.migration.data.filename` property and set its value to point to the full path to the `atlas-migration-data.json` file in the output folder you specified when you exported the HDP-2.x data.
3. Upgrade HDP and Atlas.
 4. The upgrade starts Atlas automatically, which initiates the migration of the uploaded HDP-2.x Atlas metadata into HDP-3.x. During the migration import process, Atlas blocks all REST API calls and Atlas hook notification processing.

You can use the following Atlas API URL to display the migration status:

```
http://<atlas_server>:21000/api/atlas/admin/status
```

The migration status is displayed in the browser window:

```
{"Status": "Migration", "currentIndex": 139, "percent": 67, "startTimeUTC": "2018-04-06T00:5
```

5. When the migration is complete, select Atlas > Configs > Advanced > Custom application-properties, then click the red Remove button to remove the `atlas.migration.data.filename` property.

Installing Atlas

You can use Apache Atlas to effectively and efficiently address your compliance requirements through a scalable set of core data governance services.

Before you begin

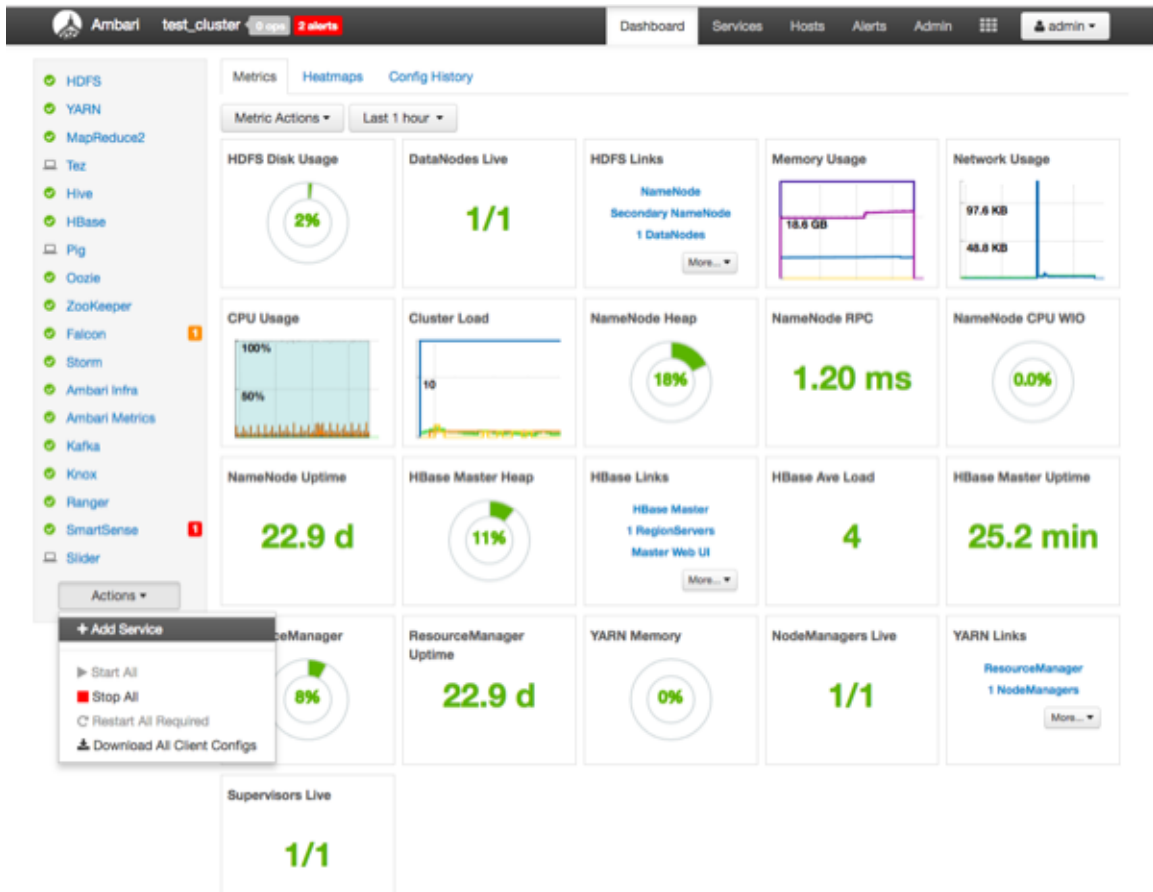
- Ambari Infra (which includes an internal HDP Solr Cloud instance) or an externally managed Solr Cloud instance
- Apache HBase (used as the Atlas metastore)
- Apache Kafka (provides a durable messaging bus)

Start the installation

Use the following steps to start the Apache Atlas installation.

Procedure

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

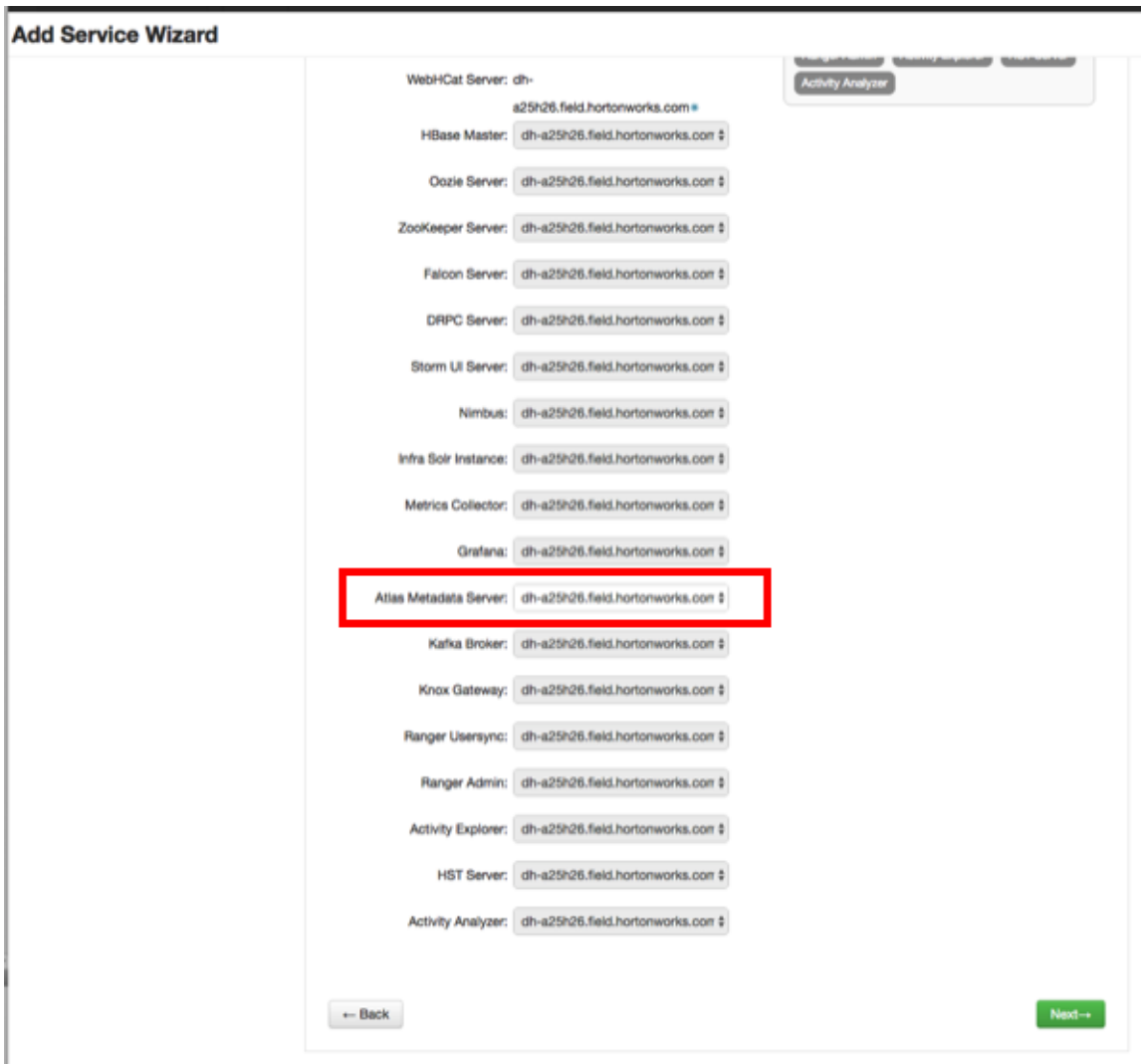
- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Configure Identities
- Review
- Install, Start and Test
- Summary

Choose Services

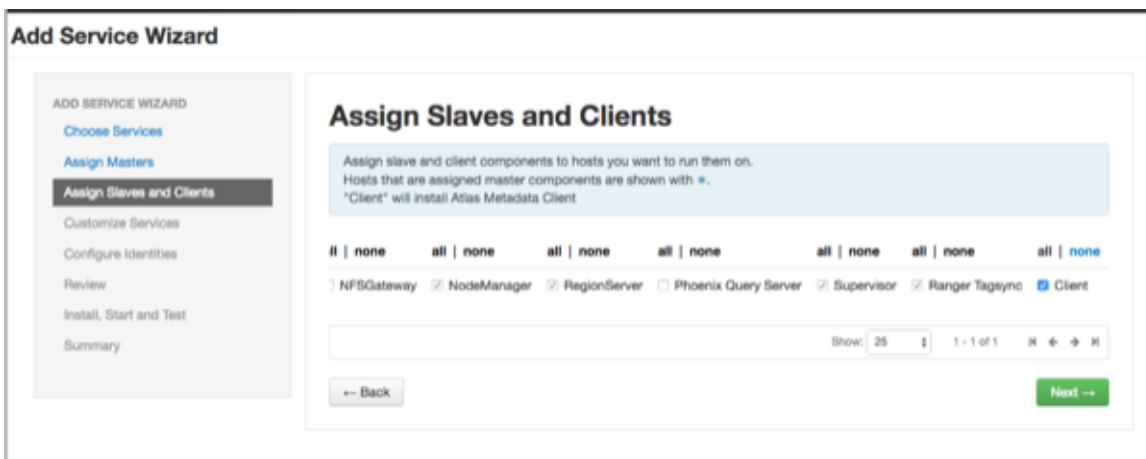
Choose which services you want to install on your cluster.

<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 ExtJS Library.
<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.

Customize services

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

Authentication settings

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

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 screenshot shows the 'Add Service Wizard' interface. On the left is a sidebar with steps: 'Choose Services', 'Assign Masters', 'Assign Slaves and Clients', 'Customize Services' (selected), 'Configure Identities', 'Review', 'Install, Start and Test', and 'Summary'. The main area is titled 'Customize Services' and lists various services like HDFS, YARN, MapReduce2, Tez, Hive, HBase, Pig, Oozie, ZooKeeper, Falcon, Storm, Ambari Infra, Ambari Metrics, Atlas, Kafka, Knox, Ranger, SmartSense, Slider, and Misc. A notification bar indicates 'There are 8 configuration changes in 4 services'. Below this, there's a 'Group' dropdown set to 'Default (1)' and a 'Manage Config Groups' button. The 'Authentication' section is active, showing 'Authentication Methods' with three options: 'Enable File Authentication' (checked), 'Enable LDAP Authentication' (checked), and 'Enable Atlas Knox SSO' (unchecked). The 'File' section contains a text input field for 'atlas.authentication.method.file.filename' with the value '{{conf_dir}}/users-credentials.properties' highlighted by a red box. The 'LDAP/AD' section shows 'LDAP Authentication Type' set to 'AD' and 'atlas.authentication.method ldap.ad.uri' set to '10.42.0.63'.

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 -
```



Note:

You can also set the Admin password using the Ambari UI: Select Advanced > Advanced atlas-env, then use the Admin password and Admin username boxes to set the Admin user name and password.

When updating these settings post-installation, click Save, then restart Atlas and all other components that require a restart.

LDAP Authentication

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

Table 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

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

LDAP/AD

LDAP Authentication Type

atlas.authentication.method.ldap.url

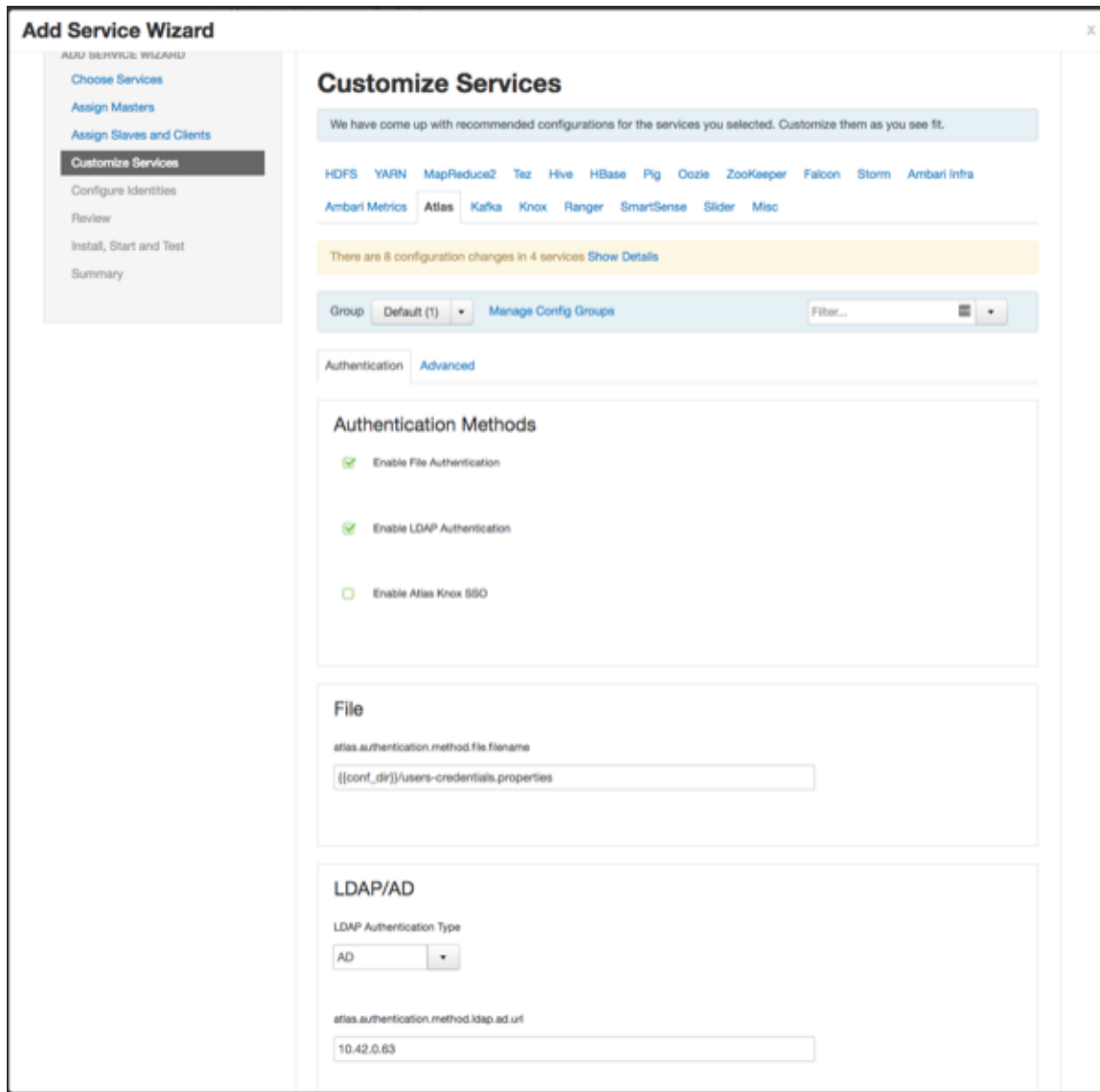
atlas.authentication.method.ldap.useObfuscated

AD Authentication

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

Table 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



Authorization settings

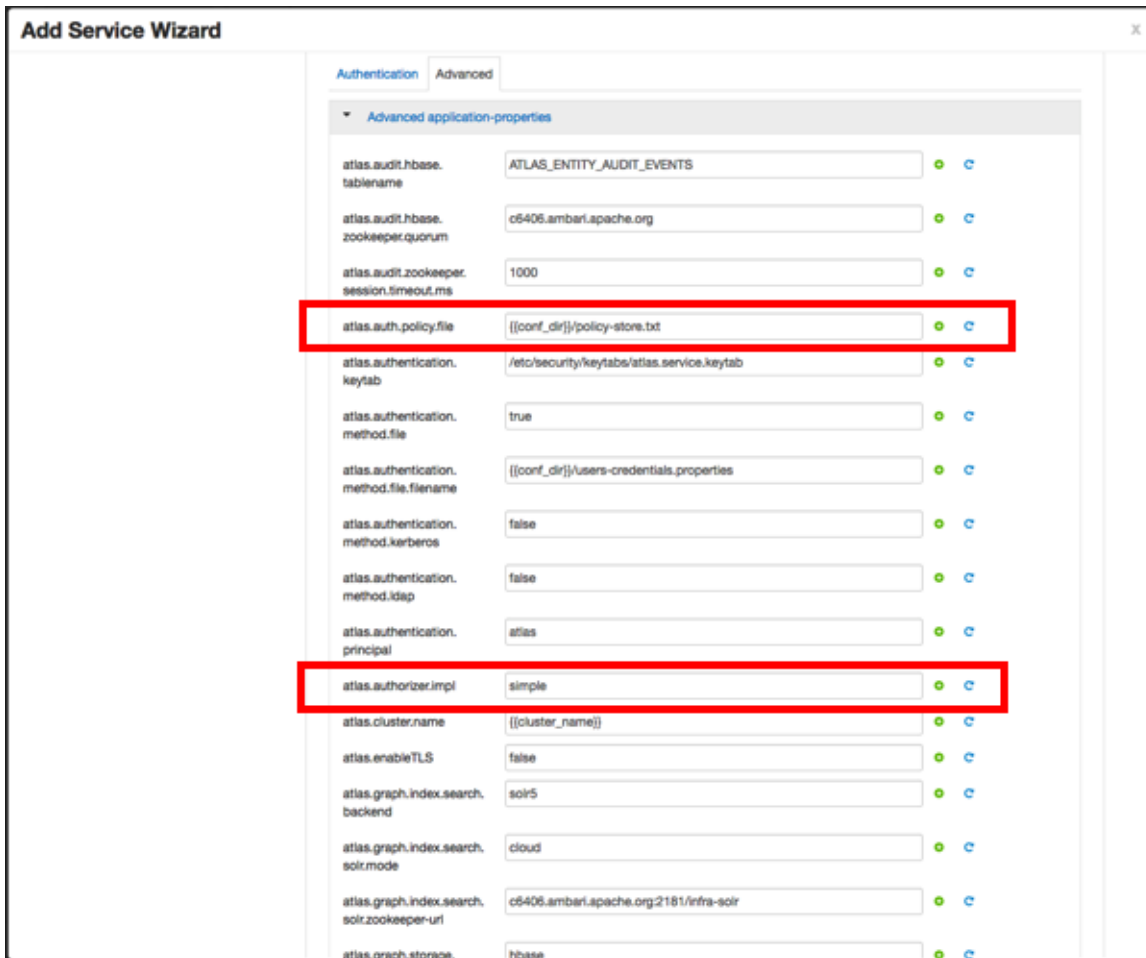
Two authorization methods are available for Atlas: Simple and Ranger.

Simple Authorization

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

Table 3: Apache Atlas Simple Authorization

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



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

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

For example:

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

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

Ranger Authorization

Ranger Authorization is activated by enabling the Ranger Atlas plug-in in Ambari.

Related Information

[Enable the Atlas Ranger Plugin](#)

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.ATSHook,org.apache.atlas.hive.hook.HiveHook	org.apache.hadoop.hive.ql.hooks.ATSHook,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.atlas.rest.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-a25h26k.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

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.

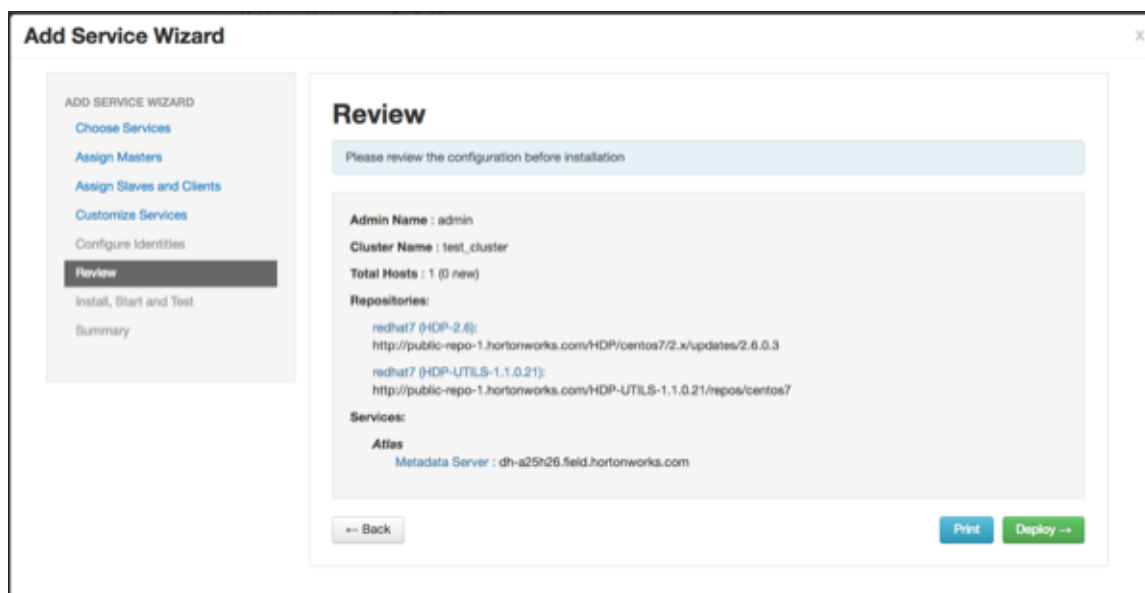
Back Next

Complete the Atlas installation

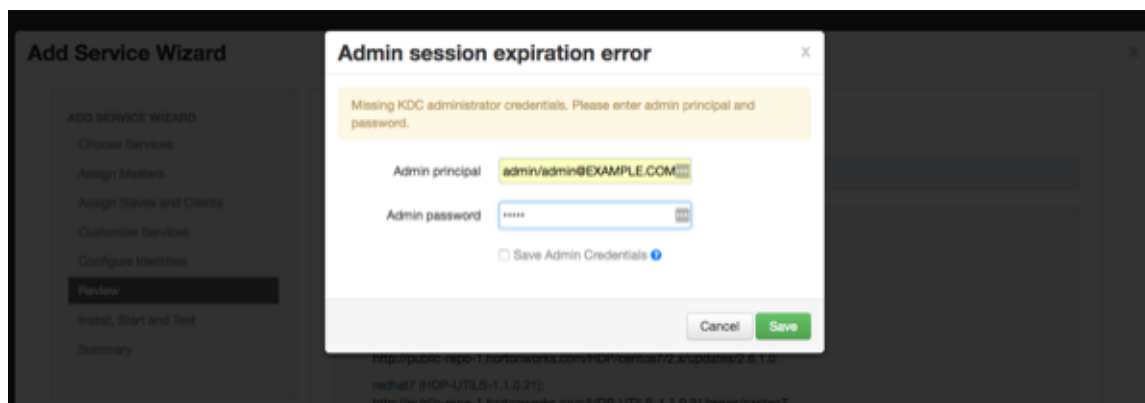
Review the Atlas configuration and complete the installation.

Procedure

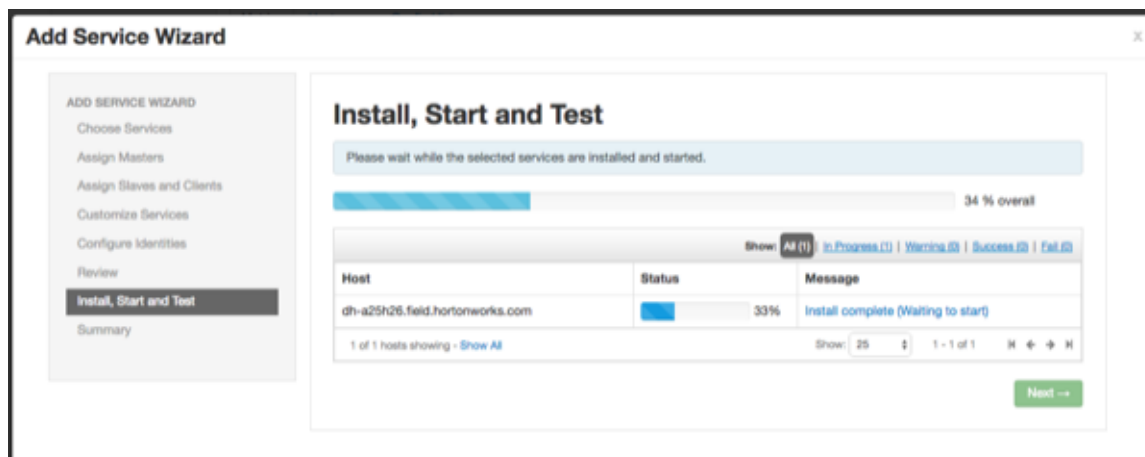
1. On the Review page, carefully review the configuration. If everything looks good, click Deploy to install Atlas on the Ambari server.



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.

Install sample Atlas metadata

You can use the quick_start.py Python script to install sample metadata to view in the Atlas web UI.

Procedure

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.