DLM Installation

Date of Publish: 2018-05-18

http://docs.hortonworks.com

Contents

DLM support requirements

Prior to installing Data Lifecycle Manager (DLM), you must consider various aspects of your HDP environment and prepare your clusters prior to DLM installation. The host on which you install DLM is the same host on which you install all DPS Platform.

Support Matrix information

You can find the most current information about interoperability for this release on the Support Matrix. The Support Matrix tool provides information about:

- Operating Systems
- Databases
- Browsers
- JDKs

To access the tool, go to: https://supportmatrix.hortonworks.com.

DLM Host requirements

The DLM application is installed on the same host as DPS Platform and has no requirements beyond what is required by DPS Platform. See the *DPS Platform Support Requirements* for details.

Requirements for clusters used with DLM Engine

The clusters on which you install the DLM Engine must meet the requirements identified in the following sections. After the DLM Engine is installed and properly configured on a cluster, the cluster can be registered with DPS and used for DLM replication.

> **Important:**
>
> Clusters used as source and destination in a DLM replication relationship must have exactly the same configurations for LDAP, Kerberos, Ranger, Knox, and HA.

See the Support Matrix for supported operating systems and databases.

Port and network requirements for clusters

Have the following ports available and open on each cluster:

Default Port Number	Purpose	Comments	Required to be open?
25968	Port for DLM Engine (Beacon) service on hosts	Accessibility is required from all clusters. "Beacon" is the internal name for the DLM Engine. You will see the name Beacon in some paths, commands, etc.	Yes
8020	NameNode host		Yes
50010	All DataNode hosts		Yes
8080	Ambari server host		Yes
10000	HiveServer2 host	Binary mode port (Thrift)	Yes
10001	HiveServer2 host	HTTP mode port	Yes
9083	Hive metastore		Yes
2181	ZooKeeper hosts		Yes
6080	Ranger port		Yes

Default Port Number	Purpose	Comments	Required to be open?
8050	YARN port		Yes

HDP component requirements for DLM

The following additional Apache components might be required on your clusters for DLM support, depending on the security configuration and type of replication being performed:

Component	Purpose	Comments
HDFS	For replicating HDFS data.	
Knox	Authentication federation from DPS	Knox must be enabled on clusters before you can register the clusters with DPS.
Ranger	Authorization on clusters during replication	Ranger is optional for HDFS replication, but required for Hive replication.
YARN		
Hive	For replicating Hive database content	Updates via Hive 1 (Based on Apache Hive 1.2.x) and HiveServerInteractive (Based on Apache Hive 2.1.x) are replicated. However, HiveServer2 from Hive 1 is always used for running the replication tasks.
HiveServer 2	Needed for Hive replication	
Hive Metastore	Needed for Hive replication	
Zookeeper	Needed for Hive	

Cross-version support of DLM and HDP

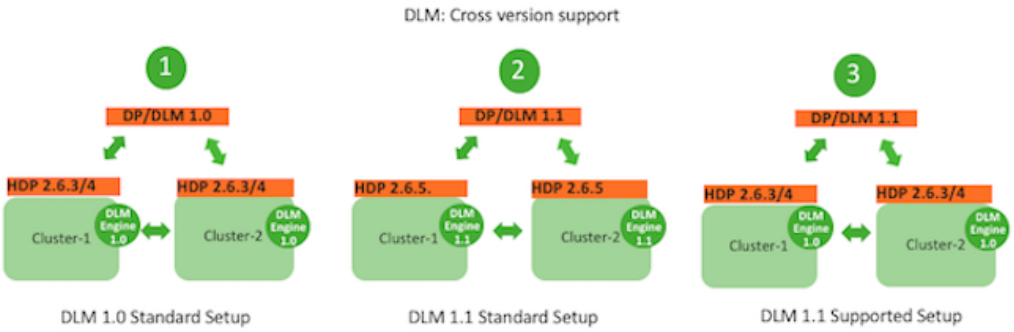
You need to ensure that you are using versions of DLM App, DLM Engine, HDP, and Ambari that are supported together.

DLM supports the following three basic configurations:

Table 1: Cross-version support scenarios

Scenario	DLM App Version	HDP/Ambari Version	DLM Engine Version
1. DLM 1.0 standard setup	DLM App 1.0	HDP 2.6.3/Ambari 2.6.0.0 and HDP 2.6.4/ Ambari 2.6.1.0	DLM Engine 1.0
2. DLM 1.1 standard setup	DLM App 1.1	HDP 2.6.5/Ambari 2.6.2.0	DLM Engine 1.1
3. DLM 1.1 additional supported setup	DLM App 1.1	HDP 2.6.3/Ambari 2.6.0.0 and HDP 2.6.4/ Ambari 2.6.1.0	DLM Engine 1.0

The *Cross-version support scenarios* table is illustrated in the following image:



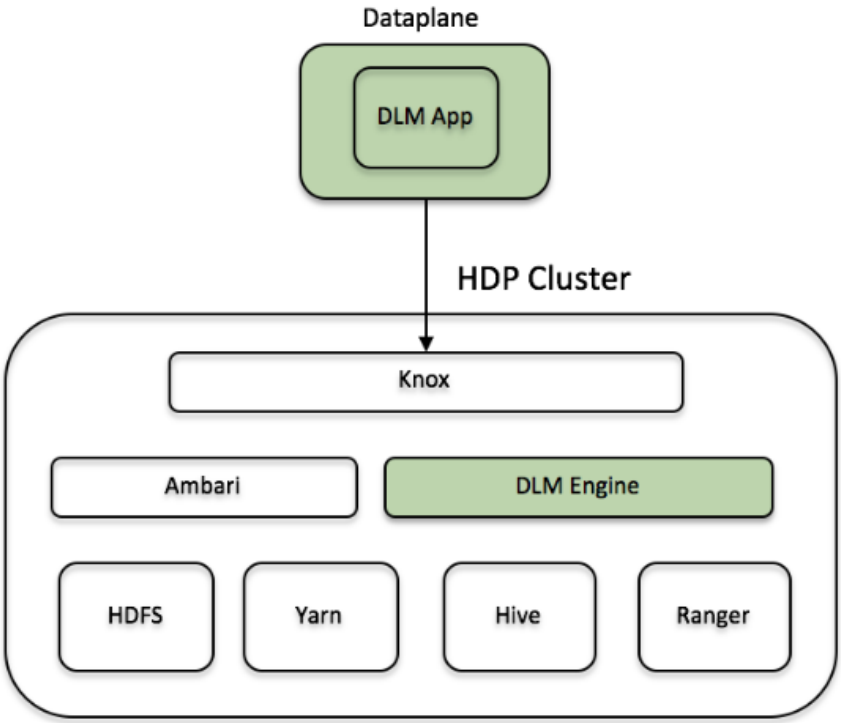
Important:

Pairing between clusters for DLM replication is only supported between source and destination clusters that use the same version of DLM Engine and the same HDP version.

Installation overview

The DLM Service is composed of an App and an Engine. The DLM App is installed on the same host as the DPS Platform App. The DLM Engine (also called "Beacon") is installed on each cluster that you plan to use with DLM.

DPS Platform and the DLM App communicate with the HDP cluster through Knox. Knox SSO is a required configuration for the DPS Platform host. DLM replication also requires HDFS, YARN, Hive, and Ranger on the cluster. Knox Gateway is recommended to protect data being transferred between clusters.



The installation process for DLM includes the following tasks:

1. Download the required tarballs from the customer portal, following the instructions provided as part of the product procurement process.
2. Set up a local repository for DLM on the DPS Platform host.
3. Install the DLM App on the DPS Platform host.

4. Install the DLM Engine on each cluster.
5. Configure the Beacon (DLM Engine) User.
6. Configure the DLM Engine on Ambari.

Setting Up the Local Repository for Your DLM Installation

To install Hortonworks Data Lifecycle Manager (DLM), you must have previously downloaded tarballs from the customer portal, following the instructions provided as part of the product procurement process. The DLM service is installed as an RPM in a repository on the DPS host. The tarballs containing the RPMs should have been downloaded to the DPS host.

You must also download tarballs containing MPacks. These tarballs should be downloaded to the Ambari hosts on each cluster to be used for DLM replication. You install the DLM Engine on the clusters by using the MPacks.

Set up a local repository for DLM

If you did not set up a repository for DLM when you created the DPS repository, you must do so now. Setting up a local repository involves moving the tarball to the selected mirror server and extracting the tarball to create the repository.

Before you begin

- You must have installed and configured DPS Platform.
- You must have downloaded the required tarball from the customer portal, following the instructions provided as part of the product procurement process.
- You must have properly prepared the web server during creation of the DPS repository.

Procedure

1. Copy the repository tarball for the DLM Instance to the web server directory and expand (uncompress) the archive file:
   a) Navigate to the web server directory you previously created.

   ```
   cd /var/www/html/
   ```

   All content in this directory is served by the web server.
   b) Move the tarball to the current directory and expand the repository tarball.

   Replace <filename> with the actual name of the RPM tarball that you are expanding.

   ```
   tar zxvf <file-name>.tar.gz
   ```

   During expansion of the tarball, subdirectories are created in /var/www/html/, such as DLM/centos7. These directories contain the repositories.

   Expanding the tarball might take several seconds.
2. Confirm that you can browse to the newly created local repository by using the *Base URL*:

   ```
   http://<your_webserver>:port/<repo_name>/<OS>/<version>
   ```

   - <your_webserver>:port

     This is the FQDN and port of the web server host.
   - <repo_name>

The repository name, usually the abbreviated name of the DPS component, for example DLM for *Data Lifecycle Manager*.

- <OS>

  The operating system.

- <version>

  The version number of the downloaded component.

Base URL example:

```
http://<your_webserver>:port/DLM/centos7/1.1.0.0
```

Remember this Base URL. You need it to set up the repository configuration file in subsequent steps.

3. If you have multiple repositories configured in your environment, deploy the following plugin on all the nodes in your cluster.

```
yum install yum-plugin-priorities
```

4. Edit the /etc/yum/pluginconf.d/priorities.conf file to add the following values:

```
[main]
enabled=1
gpgcheck=0
```

Results

The local repository is now set up and ready for use.

What to do next

Create the configuration file for the newly created local repository.

Create the repository configuration file

A repository configuration file (".repo file") must be created for the DLM Service on the DPS host. The file is required to identify the path to the repository data, establish whether a GPG signature check should be performed on the repository packages, etc. Only one repository configuration file is needed.

Procedure

1. Navigate to the repository directory.

```
cd /etc/yum.repos.d/
```

2. Create a repository file.

```
vi dlm.repo
```

Alternatively, you can copy an existing repository file to edit.

3. Add the following content to the repository file:

Important: Be sure to use the Base URL you created when setting up the local repository.

```
#VERSION_NUMBER=1.1.0.0
[DLM-1.1.0.0]
name=DLM  Version - DLM-1.1.0.0
baseurl=http://<your_webserver>:port/DLM/centos7/1.1.0.0
```

```
gpgcheck=1
gpgkey=http://<your_webserver>:port/DLM/centos7/1.1.0.0/RPM-GPG-KEY/RPM-
GPG-KEY-Jenkins
enabled=1
priority=1
```

What to do next

You are now ready to install the DLM Service software.

Install the DLM Service App

After installing the DPS Platform, install the DLM Service App. All service applications are installed as RPMs on the same host as DPS Platform. You can install one DPS service or a combination of DPS services with DPS Platform.

Before you begin

You must have root access to the host on which you are installing DLM.

You must have successfully installed DPS Platform.

Procedure

1. Log in as root to the host on which you set up the DPS repositories.

   ```
   sudo su
   ```

2. Start the Docker service, if not started.

   ```
   service docker start
   ```

3. Install the RPMs for the DLM service application.

   ```
   yum install dlm-app
   ```

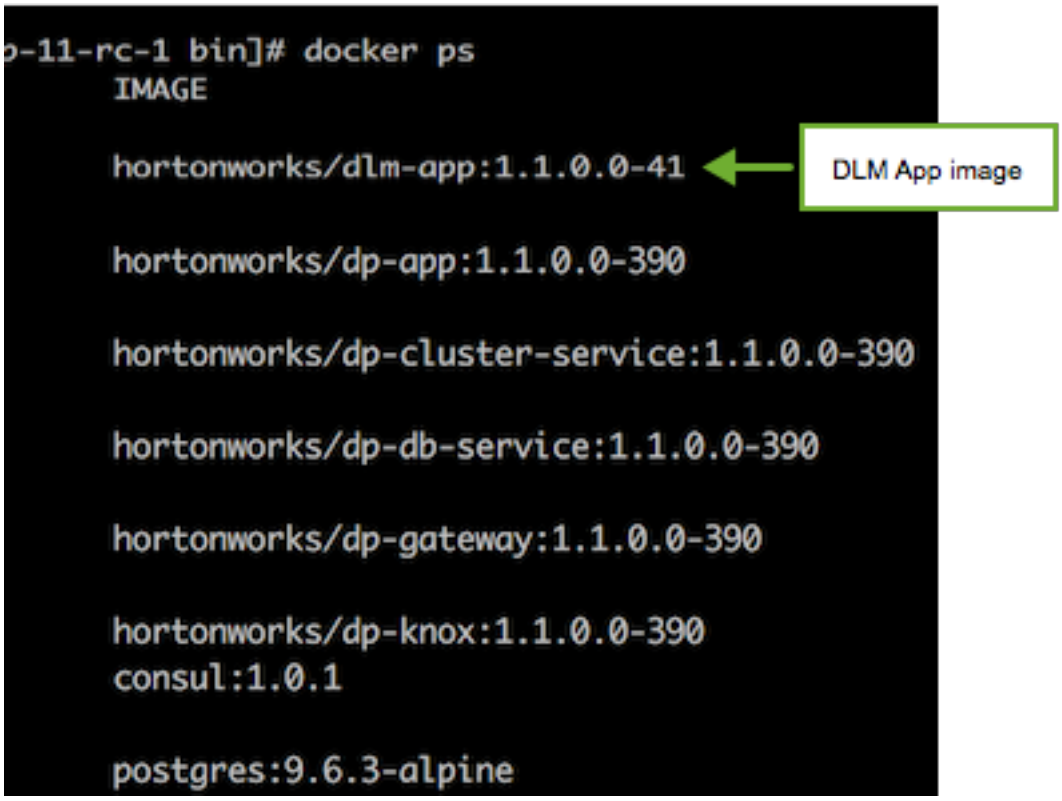
   A folder is created that contains the Docker image tarball files and a configuration script.

   If the yum command fails, then the local repository was not set up correctly. Check the repository file /etc/yum.repos.d/dlm.repo on the host.

4. Navigate to the directory containing the installation scripts for the DLM service, for example:
   cd /usr/dlm-app/current/apps/dlm/bin
5. Load the DLM Docker images and initialize the environment.
   ./dlmdeploy.sh init

   Loading the images might take a while.
6. Verify that the container you installed is running.
   docker ps

   The entry should be similar to the following:



If any containers are not running, you must destroy the containers and start over, as described in the troubleshooting section of the *DPS Installation* guide.

What to do next

You must install DLM Engine on each cluster.

Related Tasks

Install the DLM Engine on Ambari-managed clusters

Related reference

DLM support requirements

Related Information

DPS Installation

Hortonworks Support Matrix

Install the DLM Engine on Ambari-managed clusters

DLM requires that an engine be installed on each cluster that is to be used in replication jobs. The engine is installed on the Apache Ambari host, using an Ambari management pack (MPack). An MPack bundles service definitions, stack definitions, and stack add-on service definitions.

About this task

- This task must be completed on all clusters to be used with DLM.
- “Beacon” is the internal name for the DLM Engine. If you install DLM, you will see the name Beacon in some paths, commands, etc.

Before you begin

You must have root access to the Ambari Server host node to perform this task.

> **Important:** Prior to starting installation, you must have downloaded the required repository tarballs from the Hortonworks customer portal, following the instructions provided as part of the product procurement process.

Procedure

1. Log in as root to an Ambari host on a cluster.

```
ssh root@<ambari-ip-address>
```

2. Install the DLM Engine MPack by running the following command, replacing <mpack-file-name> with the name of the MPack.

```
ambari-server install-mpack --mpack <mpack-file-name> --verbose
```

3. Restart the Ambari server.

```
ambari-server restart
```

   The restart allows Ambari to create the beacon user for the service.

4. Repeat this task on each cluster being used with DLM.

What to do next

Configure the DLM Beacon user as HDFS superuser.

Related Tasks

Configure the DLM Beacon user

DLM Configuration

Configure the DLM Beacon user

The DLM Engine employs the default Beacon user to perform actions on the cluster nodes. The Beacon user is created during the DLM Engine installation, configured as a Hadoop Proxy superuser. However, the Beacon user must also be configured as an HDFS superuser, after completing the DLM Engine installation.

About this task

This task must be completed on every NameNode on clusters used with DLM.

“Beacon” is the internal name for the DLM Engine. You will see the name Beacon in some paths, commands, etc.

Procedure

1. Log in to a NameNode host as root.
2. Assign the Beacon user to the HDFS superuser group.

```
usermod -a -G hdfs beacon
```

3. Refresh configuration and mappings files.

```
hdfs dfsadmin -refreshSuperUserGroupsConfiguration
```

```
hdfs dfsadmin -refreshUserToGroupsMappings
```

4. Verify that Beacon was added as a user to the HDFS superuser group.

```
hdfs groups beacon
```

   The output should display HDFS as one of the groups.

5. Repeat this process on every NameNode used with DLM for replication.

Configure DLM Engine on existing Ambari-managed clusters

After the DLM Engine is installed on the Ambari host, you must properly configure it. You can configure the engine on existing clusters or when configuring newly created clusters. If configuring on new clusters, see the instructions for configuring DLM Engine on new clusters.

About this task

- You must have root access to the Ambari Server host node to perform this task.
- When you install DLM Engine in an existing cluster, a restart is required to get Ambari to create the service users.
- This task must be completed on all clusters to be used with DLM.
- “Beacon” is the internal name for the DLM Engine. If you install DLM, you will see the name Beacon in some paths, commands, etc.

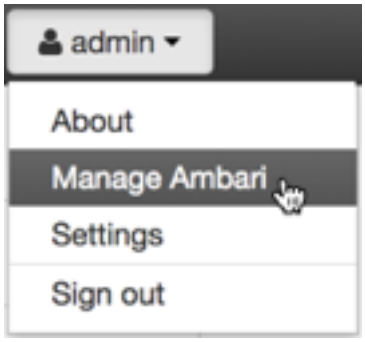
Procedure

1. Launch Ambari in a browser and log in.
   http://<ambari-server-host>:8080

   Default credentials are:

   Username: admin
   Password: admin

2. Click **Admin**>**Manage Ambari**.



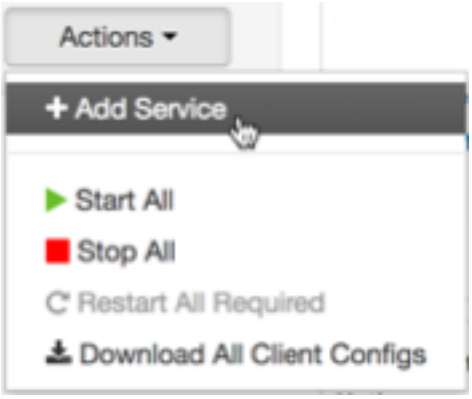
3. Click **Versions**, and then do the following on the Versions page:
   a) Click the HDP version in the **Name** column.
   b) Change the **Base URL** path for the DLM service to point to the local repository, for example:

```
http://webserver.com/DLM/centos7/1.1.0.0
```

   URLs shown are for example purposes only. Actual URLs might be different.

4. Click the Ambari logo to return to the main Ambari page.

5. In the Ambari Services navigation pane, click **Actions**>**Add Service**.



   The Add Service Wizard displays.
6. On the **Choose Services** page of the Wizard, select the DPS service to install in Ambari, and then follow the on-screen instructions.

   Other required services are automatically selected.
7. When prompted to confirm addition of dependent services, give a positive confirmation to all.

   This adds other required services.
8. On the **Assign Masters** page, you can choose the default settings.
9. On the **Customize Services** page, fill out all the required username and password fields that are highlighted.

   You can set credentials to whatever you want.
10. If doing Hive replication with DLM, navigate to **Customize Services** and enable Hive replication.
    a) Click **Hive** in the list of services.
    b) On the **Settings** tab, move the toggle to *off* for "Run as end user instead of Hive user".
    c) Click the **Advanced** tab and scroll to the **Custom hive-site** section.
    d) Verify that these properties have the following values, or set the properties as shown:

```
hive.metastore.dml.events=true
hive.metastore.transactional.event.listeners=org.apache.hive.hcatalog. \
listener.DbNotificationListener
hive.repl.cm.enabled=true
hive.repl.cmrootdir=/apps/hive/cmroot
hive.repl.rootdir=/apps/hive/repl
```

    e) Click **HDFS** in the list of services.
    f) Scroll to the **Custom core-site** section and modify the following parameter:
    hadoop.proxyuser.hive.hosts=*
11. Complete the remaining installation wizard steps and exit the wizard.
12. Ensure that all components required for the DLM Service have started successfully.

    See the DLM support requirements for information about required Apache components.
13. Repeat this procedure on the Ambari hosts on all remaining clusters on which you install the DLM agent.

Configure DLM Engine on new Ambari-managed clusters

If you installed DPS on a new cluster, you can install the service engine using the Install Wizard process. If using existing clusters, see the instructions for configuring the DLM Engine on existing clusters.

About this task

- You must have root access to the Ambari Server host node to perform this task.
- This task must be completed on all clusters to be used with DLM.

- See the *DPS Support Matrices* for information about Apache components required for the DPS services you are installing.

Procedure

1. Launch Ambari in a browser and log in.
   http://<ambari-server-host>:8080

   Default credentials are shown below.
   Username: admin
   Password: admin
2. From the Ambari Welcome page, choose Launch Install Wizard and begin the wizard.
3. On the **Select Versions** page, change the **Base URL** path for each DPS service to point to the local repository, for example:

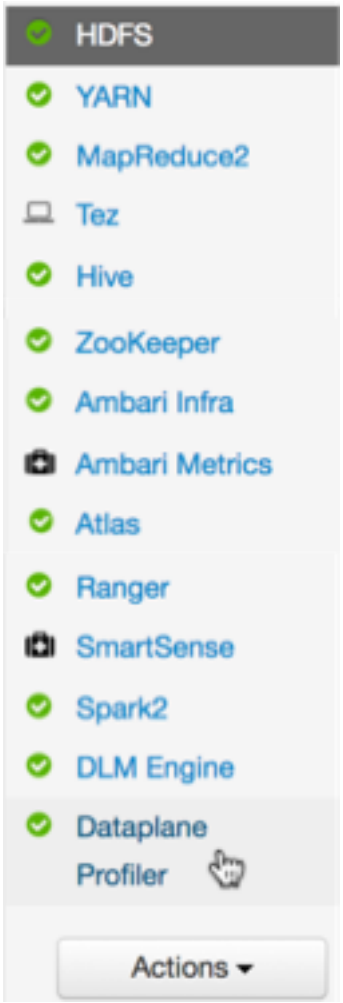
   ```
   http://webserver.com/DLM/centos7/1.1.0.0
   ```

   URLs shown above are for example purposes only. Actual URLs might be different.
4. On the **Select Services** page of the Install Wizard, select the engine or agent you want to configure, and then follow the on-screen instructions.

   Other required services are automatically selected.
5. When prompted to confirm addition of dependent services, give a positive confirmation to all.

   This adds other required services.
6. On the **Assign Masters** page, you can choose the default settings.
7. On the **Customize Services** page, fill out all the required username and password fields that are highlighted.

   You can set credentials to whatever you want, such as admin/admin.
8. If doing Hive replication with DLM, from the **Customize Services** page, enable Hive replication.
   a) Click **Hive** in the list of services.
   b) On the **Settings** tab, move the toggle to *off* for "Run as end user instead of Hive user".
   c) Click **HDFS** in the list of services.
   d) Scroll to the **Custom core-site** section and modify the following parameter:
   hadoop.proxyuser.hive.hosts=*
9. If using DSS, configure the DataPlane Profiler.
   a) In the Services pane, click **DataPlane Profiler**.



b) Click **Configs**>**Advanced**, and then scroll to **Custom core-site**.

c) Click **Add Property** and enter the following key-value pairs:

hadoop.proxyuser.livy.groups=*

hadoop.proxyuser.livy.hosts=*

10. Complete the remaining installation wizard steps and close the wizard.

11. Ensure that all components required for your DPS services have started successfully.

See the DPS support requirements for information about required Apache components.

12. Repeat this procedure on the Ambari hosts on all remaining clusters, for each DPS service engine and agent you installed.

Requirements for DLM authorization on Kerberos secured clusters

In addition to the security tasks you must complete for DPS, and to satisfy your environmental or corporate requirements, you must ensure the following are properly configured so that DLM replication jobs complete successfully on clusters with Kerberos enabled. No other special configuration is required for authorization and authentication with DLM on a cluster secured using Kerberos.

- HDFS, Hive, Knox, and Ranger are enabled in Ambari
- Ranger plugins are enabled for HDFS and Hive
- Clusters to be paired in DLM have identical configurations, including security
- Global LDAP is configured to share user-group mappings across clusters

- If using Kerberos with different KDCs, two-way trust is configured between the KDCs
- If using AD, there is no support for trust relationships across multiple domains or forests

Configure Knox SSO with DLM

If you have the DLM Engine on the cluster, you must take additional steps to set up your Knox SSO configuration.

About this task

You will perform this DLM Engine Knox SSO setup on your clusters after you perform the DPS Installation. Refer to DPS Installation for more information.

Procedure

1. Export the Knox certificate:
   a) From the Knox Gateway machine, run the following command: $JAVA_HOME/bin/keytool -export -alias gateway-identity -rfc -file <cert.pem> -keystore /usr/hdp/current/knox-server/data/security/keystores/gateway.jks
   b) When prompted, enter the Knox master password.
   c) Remember the location where you save the cert.pem file.
2. Enable the Knox SSO topology settings:
   a) From **Ambari** > **DLM Engine** > **Configs** > **Advanced** > **Advanced beacon-security-site**, click the checkbox beside **beacon.sso.knox.authentication.enabled**.
   b) Set **beacon.sso.knox.provideurl** to https://<knox-host>:8443/gateway/knoxsso/api/v1/websso.
   c) Copy the contents of the PEM file exported in Step 1 to **beacon.sso.knox.publicKey**

      Ensure the certificate headers are not copied.

   d) Click Save and click through the confirmation pop-ups.
   e) Restart DLM Engine.
   f) Select **Actions** > **Restart All Required** to restart all other services that require a restart.

Advanced configurations

In addition to the basic configuration required to set up and use DLM replication, there are some advanced configuration options you might choose to implement. For example, you might want to configure TLS wire encryption for protecting data in motion, or Transparent Data Encryption (TDE) for protecting data at rest.

Using TLS wire encryption with DLM

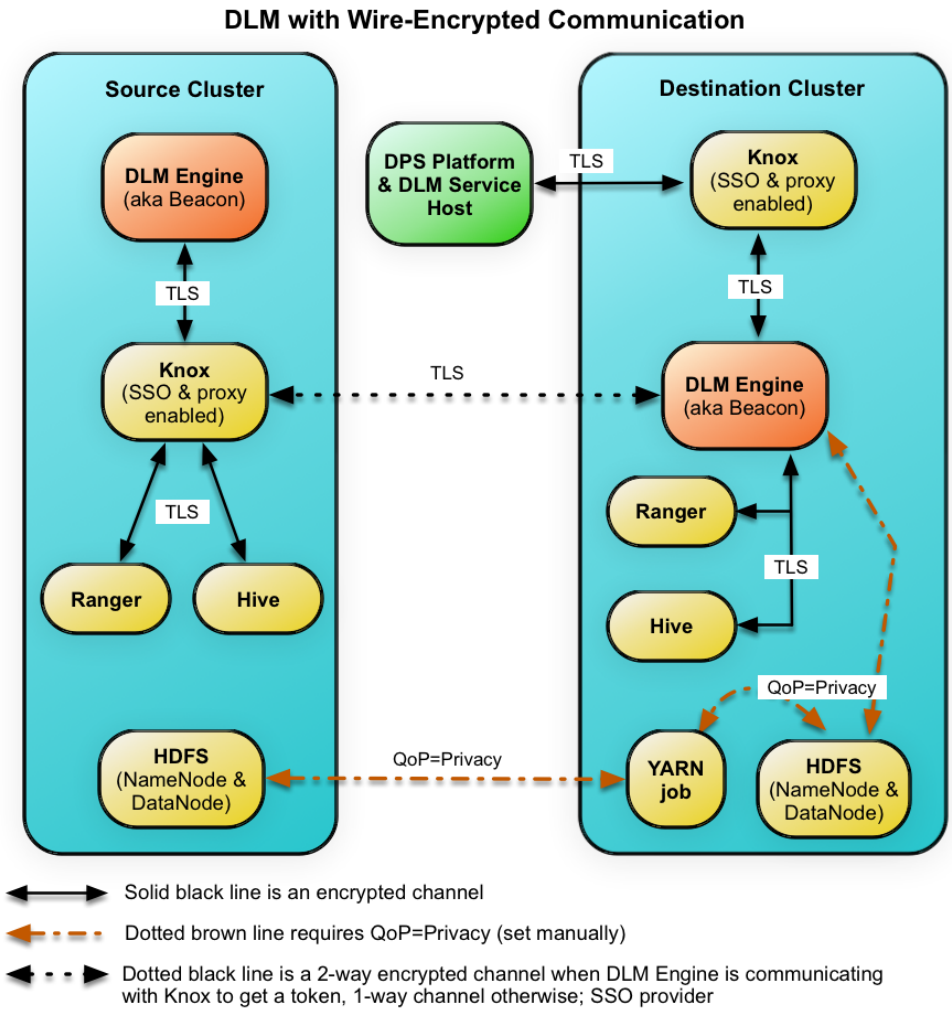
DLM includes an optional transport level security (TLS) wire encryption feature. TLS (formerly SSL) provides in-flight encryption so that data is securely transferred between source and destination clusters. TLS wire encryption uses Knox proxied endpoints for HDP services accessed by the DLM engine within each cluster.

Using wire encryption in DLM-enabled clusters is optional. However, wire encryption must be configured the same within clusters paired in a source/destination relationship in DLM. That is, both clusters in a pairing must have intracluster encryption enabled using TLS or both must have TLS disabled.

The diagram below illustrates the following configuration aspects of DLM wire encryption with TLS:

- DLM Engine communications on the source are proxied through Knox using HTTPS.
- DLM Engine on the destination cluster communicates with the source cluster services (source DLM Engine, Ranger, and Hive) by proxying through Knox using HTTPS.
- The DLM Engine uses two-way SSL to authenticate token requests.
- DistCp on the destination uses SASL+QoP=privacy to communicate with the HDFS services on the source and destination clusters.

Figure 1: TLS wire encryption in DLM-enabled clusters that use Kerberos



When wire encryption is enabled in the cluster, Knox proxying should be set up and enabled.

Configure DLM Engine for TLS with a trusted CA certificate

You can enable SSL for the DLM Engine using a certificate from a trusted Certificate Authority (CA). Certificates from a trusted CA are primarily used in production environments. For a test environment, you can use a self-signed certificate.

Before you begin

- You must have root user access to the clusters on which DLM Engine is installed.
- You must have obtained a certificate from your CA, following their instructions.

Procedure

1. Log in as root user on the cluster with DLM Engine installed.
2. Import the Certificate Chain Certificate and the certificate you obtained from your CA.

```
keytool -import -alias root -keystore <path_to_keystore_file> -
trustcacerts -file <certificate_chain_certificate>
```

```
keytool -import -alias jetty -keystore <path_to_keystore_file> -file
 <certificate_from_CA>
```

What to do next

Configure the keystore for DataPlane use.
Add link to security content

Configure DLM Engine for TLS with a self-signed certificate

You can enable SSL for the DLM Engine using a self-signed certificate. Self-signed certificates are primarily used in test environments. For a production environment, you should use a certificate from a trusted CA.

Before you begin

You must have root user access to the clusters on which DLM Engine is installed.

Procedure

1. Log in as root user on the cluster with DLM Engine installed.
2. Generate a key pair and keystore for use with DLM Engine.

```
keytool -genkey -alias jetty -keystore <certificate_file_path>
-storepass <keystore_password> -dname 'CN=beacon.host.com, OU=Eng, O=ABC
 Corp,
L=Santa Clara, ST=CA, C=US' -keypass <key_password>
```

Follow the prompts and enter the required information.

- CN must be the FQDN of the DLM Engine host
- Default value for the key password is *password*.

  If you change the password then you have to update the DLM configuration.

Following is sample command output:

```
keytool -genkey -alias jetty -keystore ~/tmp/ks -storepass password
What is your first and last name?
  [Unknown]:  beacon.host.com
What is the name of your organizational unit?
  [Unknown]:  Eng
What is the name of your organization?
  [Unknown]:  ABC Corp
What is the name of your City or Locality?
  [Unknown]:  Santa Clara
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=beacon.host.com, OU=Eng, O=ABC Corp, L=Santa Clara, ST=CA, C=US
 correct?
  [no]:  yes

Enter key password for <jetty>
```

```
 (RETURN if same as keystore password):
```

3. Export the certificate.

```
keytool -exportcert -alias jetty -keystore /my/file.keystore -file
 <certificate file path>
```

The keystore password is the same as the key password.

What to do next

Configure the keystore for DataPlane use.

Configure the keystore for use by DPS Platform

You must copy to the DPS Platform host the certificate file that was exported (for self-signed certificates) or the certificate you received from the Certificate Authority (CA).

Procedure

1. Log in as root on the DPS host.
2. Import the certificate.

```
keytool -import -alias jetty -file <certificate_file> -keystore $JRE_HOME/
lib/security/cacerts
```

- JRE_HOME is the JRE used by the host client.
- <certificate_file> is the server certificate file.
- The default password for the cacerts keystore is changeit.
- When prompted, enter yes to trust the certificate.

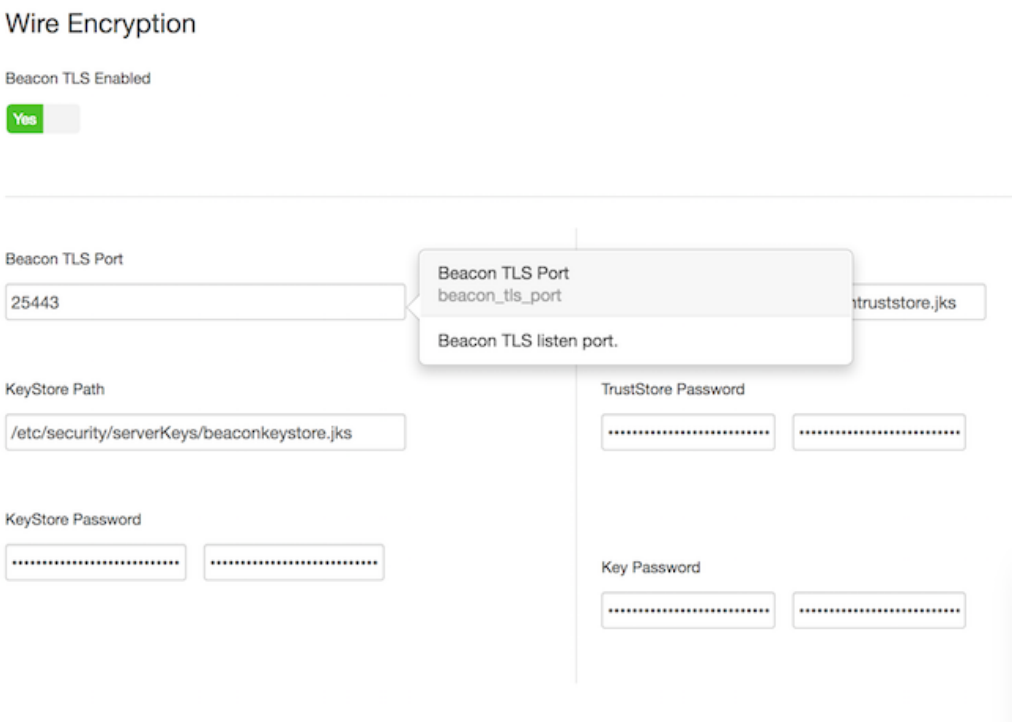
Configure DLM Engine for TLS in Ambari

In the Ambari UI, you enable TLS for DLM Engine and update the DLM Engine configuration if settings change.

Procedure

1. From the Ambari UI, stop the DLM Engine service.
   DLM Engine>Service Actions>Stop
2. Navigate to **DLM Engine>Configs>Settings** and scroll to the Wire Encryption settings.
3. Toggle the **Beacon TLS Enabled** switch and enter or modify the appropriate properties:

- Beacon TLS Port: The TLS listener port.
- KeyStore Path: Path to the DLM Engine keystore
- KeyStore Password: Password for the DLM Engine keystore
- TrustStore Path: Path to the DLM Engine truststore
- TrustStore Password: Password for the DLM Engine truststore
- Key Password: Password for the DLM Engine key



4. Restart the DLM Engine service.
   DLM Engine>Service Actions>Start

Configure DLM proxying for TLS wire-encrypted clusters

If you are using TLS (formerly SSL) wire encryption, you must configure DLM so that service requests are proxied through a Knox Gateway. This limits access to cluster services, providing a more secure environment. All cluster services such as Hive, Ambari, Ranger, etc. are accessed through a Knox proxy by DPS Platform and DLM Engine.

About this task

To use wire encryption with DLM, you must configure TLS on each cluster running DLM Engine so that the engine can authenticate and communicate with Knox across all paired clusters.

- You must perform this task on all cluster nodes that wire encryption enabled.
- If proxying is used, it must be enabled on both clusters in a DLM replication pair.

  By default, proxying with Knox is disabled in DLM.
- When proxying is enabled, you cannot pair a cluster running DLM Engine version 1.0 with a cluster running a higher version of the engine.

Before you begin

- TLS must be configured for Knox before proxying will work with DLM.
- To perform this task, you must have root user privileges on the DLM host and on all nodes that have Knox enabled.
- You must have created the dp-proxy.xml file during DPS configuration.

Before you begin

Procedure

1. In a terminal, navigate to the Knox topologies directory.

```
cd /etc/knox/conf/topologies
```

2. Log in as root and create a beacon-preauth.xml file.

```
vi beacon-preauth.xml
```

Example beacon-preauth.xml topology file:

You can copy and paste this sample content into your file and modify as needed.

```
<topology>
    <gateway>
         <provider>
            <role>federation</role>
            <name>HeaderPreAuth</name>
            <enabled>true</enabled>
            <param>
                <name>
                    preauth.custom.header
                </name>
                <value>
                    BEACON_USER
                </value>
            </param>
        </provider>
        <provider>
          <role>identity-assertion</role>
          <name>HadoopGroupProvider</name>
          <enabled>true</enabled>
       </provider>
        <!-- currently validating this acl for authorization -->
         <provider>
             <role>authorization</role>
             <name>AclsAuthz</name>
             <enabled>true</enabled>
             <param>
                   <name>knoxtoken.acl</name>
                   <value>beacon;*;*</value>
            </param>
            </provider>
    </gateway>
    <service>
        <role>KNOXTOKEN</role>
        <param>
            <name>knox.token.ttl</name>
            <value>120000</value>
        </param>
        <param>
           <name>knox.token.client.cert.required</name>
           <value>true</value>
        </param>
        <param>
           <name>knox.token.allowed.principals</name>
           <value><semicolon separated list of beacon dn names></value>
        </param>
        <param>
            <name>knox.token.client.data</name>
            <value>cookie.name=hadoop-jwt</value>
        </param>
```

```
    </service>
</topology>
```

3. Change ownership of the beacon-preauth.xml file to Knox.

```
chown knox:hadoop beacon-preauth.xml
```

4. Open the DPS proxy topology file.

   The dp-proxy.xml file was created during installation of the DPS Instance.

```
vi dp-proxy.xml
```

5. Ensure the following service definitions are in the file and configured with the correct FQDN host names.

   Important: All DLM Engine servers that are registered with DPS must be included in this file. As new wire-encrypted clusters are registered, they must be added to this file manually.

```
 <service>
  <role>BEACON</role>
  <url>https://<dlm_engine_host>:25443</url>
 </service>
```

```
 <service>
  <role>HIVE</role>
  <url>https://<hiveserver_host>:10001/cliservice</url>
 </service>
```

   Tip: You can get the HiveServer host from the default.xml file in the topology directory.

6. Repeat this task on all cluster nodes that have Knox Gateway enabled.

Using TDE with DLM

Encryption with Transparent Data Encryption (TDE) is supported in DLM for protecting data at rest. You can use TDE to prevent people from inappropriately gaining access to your data. All of the source data being replicated must be encrypted or unencrypted. DLM does not support replication in which some data is encrypted and some is not.

Replication scenarios for TDE-enabled data

DLM supports replication of HDFS and Hive data when:

- Both source and destination are encrypted with the same key (on-premise to on-premise replication only)
- Both source and destination are encrypted with different keys
- Source is unencrypted, but destination is encrypted

Note that DLM does *not* allow replication when the source is encrypted, but the destination is unencrypted.

TDE in HDFS

HDFS implements transparent, end-to-end encryption of data read from and written to HDFS.

- TDE should be configured in the HDFS service, and the directories have to be marked as encryption zones using the encryption keys.

  Refer to the Data Protection: HDFS Encryption in the HDP *Security* guide for more information.
- You can set TDE per directory or per cluster on HDFS.

TDE with Hive

- For Hive replication in DLM, any cluster that is using TDE and acts as a source for replication *must* have the entire data warehouse in a single encryption zone.
- You can set TDE only at cluster level for Hive replication.

Configure TDE for HDFS replication

You set up TDE for HDFS replication using the instructions in the HDP *Security* guide. You can set TDE per directory or per cluster on HDFS. During the replication process, the source data is decrypted using the source key and is encrypted using the destination key.

Procedure

1. (Optional) Encrypt the source directory and grant the DLM Engine user access to the KMS key in the source Ranger service.

   Refer to Encryption in HDFS and Ranger KMS Setup for instructions.
2. Encrypt the destination directory and grant the DLM Engine user access to the KMS key in the destination Ranger service.

   Refer to Encryption in HDFS and Ranger KMS Setup for instructions.

Results

After you configure TDE on the data to be replicated, DLM can identify which directories have TDE enabled. When configuring a replication policy in the DLM App, you can identify and select the TDE-enabled data. You also have the option of replicating data using the same TDE key on both the source and destination, to reduce the overhead of decryption and encryption.

Configure TDE for Hive replication

You set up TDE for HDFS replication using the instructions in the HDP *Security* guide. You can set TDE only at cluster level for Hive replication. During the replication process, the source data is decrypted using the source key and is encrypted using the destination key.

Procedure

1. (Optional) Encrypt the source Hive warehouse directory and any additional directories as required by the Hive service and grant the DLM Engine user access to the KMS key in the source Ranger service.

   Refer to Encryption in Hive and Ranger KMS Setup for instructions.
2. Encrypt the destination Hive warehouse directory and any additional directories as required by the Hive service and grant the DLM Engine user access to the KMS key in the destination Ranger service.

   Refer to Encryption in Hive and Ranger KMS Setup for instructions.

Results

After you configure TDE on the data to be replicated, DLM can identify which directories have TDE enabled. When configuring a replication policy in the DLM App, you can identify and select the TDE-enabled data. You also have the option of replicating data using the same TDE key on both the source and destination, to reduce the overhead of decryption and encryption.

On-premise replication using different keys and Ranger-KMS instances

For data at rest that is encrypted using transparent data encryption (TDE), DLM can replicate data across different Ranger Key Management Service (KMS) encryption zones and using different encryption keys. This capability applies to replication between on-premise clusters for both HDFS and Hive data.

- Permissions are replicated along with the data.
- Ranger key management and key authorization management must be done external to DLM by an administrator with access to Ranger.
- For Hive replication
  - The entire warehouse must be in one encryption zone.
  - The change management directory (cmroot directory) should also be setup in the same encryption zone as the warehouse directory.

Following is an example of how this replication scenario might apply:

- The source cluster with Ranger-KMS-1 instance uses key-1 to decrypt the data, then passes the data to the destination.
- The destination cluster with Ranger-KMS-2 instance uses key-2 to encrypt the data on the destination.