Hortonworks Data Platform

Ranger KMS Administration Guide

(December 21, 2015)

docs.cloudera.com

Hortonworks Data Platform: Ranger KMS Administration Guide

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.

Table of Contents

1. Ranger Key Management Service 1
2. Installing the Ranger Key Management Service 2
    2.1. Install Ranger KMS using Ambari (Kerberized Cluster) 2
        2.1.1. Setting up Database Users Without Sharing DBA Credentials 7
        2.1.2. Configure HDFS Encryption to use Ranger KMS Access 8
        2.1.3. Use a Kerberos Principal for the Ranger KMS Repository 9
3. Enable Ranger KMS Audit 11
    3.1. **Save Audit to Solr** 12
    3.2. **Save Audit to HDFS** 13
4. Enabling SSL for Ranger KMS 15
5. Install Multiple Ranger KMS 20
6. Using the Ranger Key Management Service 22
    6.1. Accessing the Ranger KMS Web UI 22
    6.2. Listing and Creating Keys 23
    6.3. Rolling Over an Existing Key 25
    6.4. Deleting a Key 25
7. Ranger KMS Properties 26
8. Troubleshooting Ranger KMS 31

List of Tables

7.1. Properties in Advanced dbks-site Menu (dbks-site.xml) 26
7.2. Properties in Advanced kms-env 26
7.3. Properties in Advanced kms-properties (install.properties) 26
7.4. Properties in Advanced kms-site (kms-site.xml) 27
7.5. Properties in Advanced ranger-kms-audit (ranger-kms-audit.xml) 28
7.6. Properties in Advanced ranger-kms-policymgr-ssl 29
7.7. Properties in Advanced ranger-kms-security 29
8.1. Troubleshooting Suggestions 31

1. Ranger Key Management Service

Ranger Key Management Service (Ranger KMS) is a open source, scalable cryptographic key management service supporting HDFS "data at rest" encryption*.

Ranger KMS is based on the Hadoop KMS originally developed by the Apache community. The Hadoop KMS stores keys in a file-based Java keystore by default. Ranger extends the native Hadoop KMS functionality by allowing you to store keys in a secure database.

Ranger provides centralized administration of the key management server through the Ranger admin portal.

There are three main functions within the Ranger KMS:

1. **Key management**. Ranger admin provides the ability to create, update or delete keys using the Web UI or REST APIs. All Hadoop KMS APIs work with Ranger KMS using the keyadmin username and password.
2. **Access control policies**. Ranger admin also provides the ability to manage access control policies within Ranger KMS. The access policies control permissions to generate or manage keys, adding another layer of security for data encrypted in Hadoop.
3. **Audit**. Ranger provides full audit trace of all actions performed by Ranger KMS.

Ranger KMS along with HDFS encryption are recommended for use in all environments. In addition to secure key storage using a database, Ranger KMS is also scalable, and multiple versions of Ranger KMS can be run behind a load balancer.

This guide is intended as an introductory quick start to the Ranger Key Management Service. Content will be updated regularly.

For more information about HDFS encryption, see HDFS "Data at Rest" Encryption in the *HDFS Administration Guide*.

2. Installing the Ranger Key Management Service

This section describes how to install the Ranger Key Management Service (KMS) using Ambari on a Kerberized cluster.

Prerequisites

Ranger KMS requires HDFS and Ranger to be installed and running on the cluster.

To install Ranger using Ambari, refer to the Ranger Installation Guide. (For more information about the Ambari Add Service Wizard, see Adding a Service in the *Ambari User's Guide*.)

To use 256-bit keys, install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all hosts in the cluster. For installation information, see the Ambari Security Guide. Make sure that the Java location is specified in the $PATH environment variable.

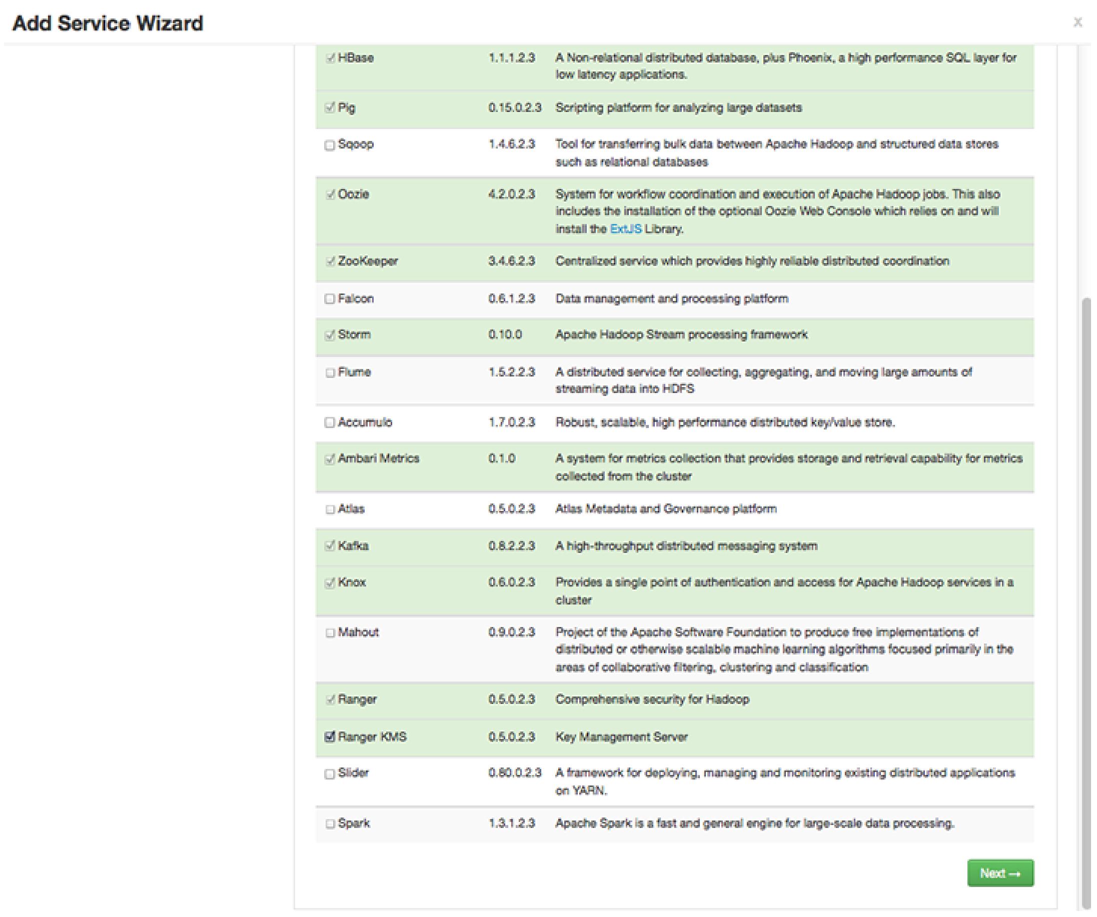
Note

If you use the OpenJDK package, the JCE file is already built into the package.

2.1. Install Ranger KMS using Ambari (Kerberized Cluster)

To install Ranger KMS on a Kerberized cluster, complete the following steps.

1. Go to the Ambari Web UI, `http://<gateway-URL>:8080`.
2. From the Ambari dashboard, go to the **Actions** menu. Choose **Add Service**.
3. On the next screen, check the box next to **Ranger KMS**:

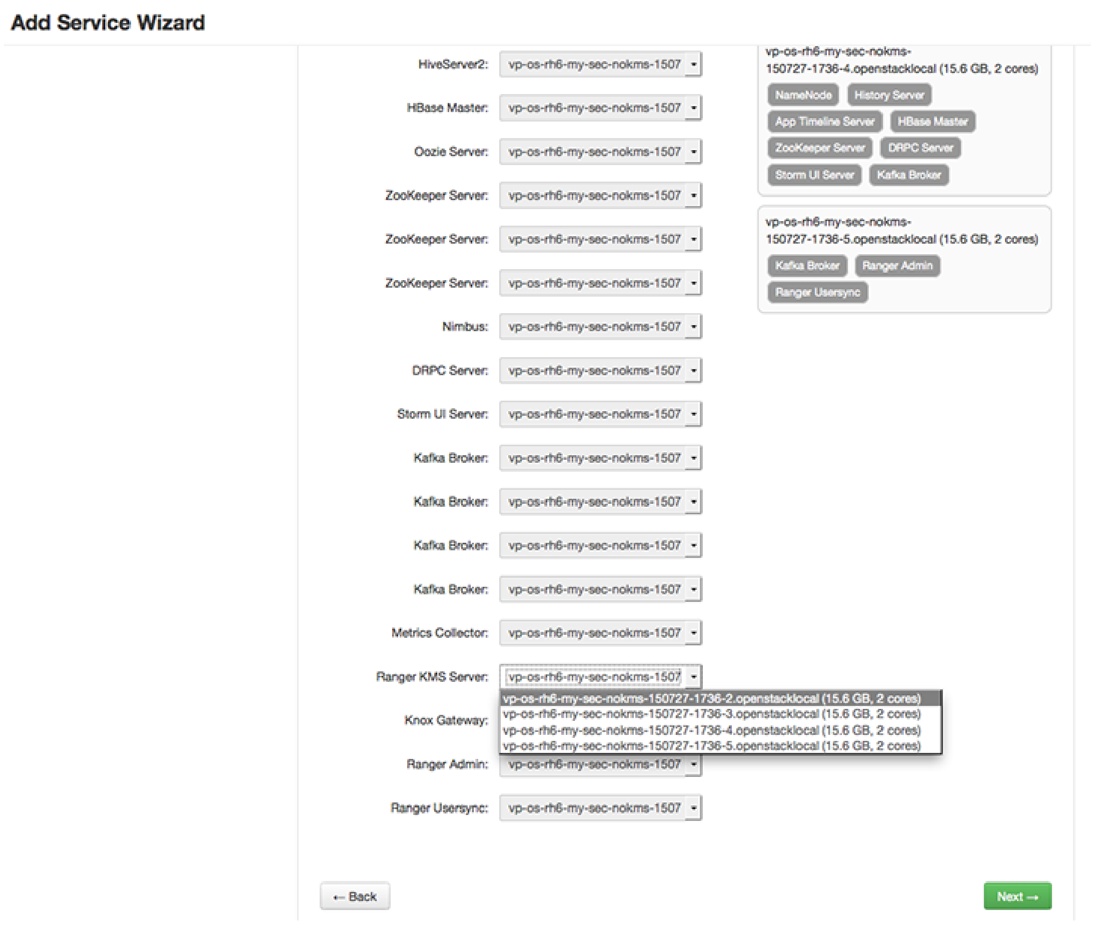


Add Service Wizard

✓ HBase	1.1.1.2.3	A Non-relational distributed database, plus Phoenix, a high performance SQL layer for low latency applications.
✓ Pig	0.15.0.2.3	Scripting platform for analyzing large datasets
☐ Sqoop	1.4.6.2.3	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases
✓ Oozie	4.2.0.2.3	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.
✓ ZooKeeper	3.4.6.2.3	Centralized service which provides highly reliable distributed coordination
☐ Falcon	0.6.1.2.3	Data management and processing platform
✓ Storm	0.10.0	Apache Hadoop Stream processing framework
☐ Flume	1.5.2.2.3	A distributed service for collecting, aggregating, and moving large amounts of streaming data into HDFS
☐ Accumulo	1.7.0.2.3	Robust, scalable, high performance distributed key/value store.
✓ Ambari Metrics	0.1.0	A system for metrics collection that provides storage and retrieval capability for metrics collected from the cluster
☐ Atlas	0.5.0.2.3	Atlas Metadata and Governance platform
✓ Kafka	0.8.2.2.3	A high-throughput distributed messaging system
✓ Knox	0.6.0.2.3	Provides a single point of authentication and access for Apache Hadoop services in a cluster
☐ Mahout	0.9.0.2.3	Project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification
✓ Ranger	0.5.0.2.3	Comprehensive security for Hadoop
☑ Ranger KMS	0.5.0.2.3	Key Management Server
☐ Slider	0.80.0.2.3	A framework for deploying, managing and monitoring existing distributed applications on YARN.
☐ Spark	1.3.1.2.3	Apache Spark is a fast and general engine for large-scale data processing.

4. Then, choose **Next**.

5. (Optional) In **Assign Masters**, if you wish to override the default host setting, specify the Ranger KMS host address. For example:



Add Service Wizard

HiveServer2:	vp-os-rh6-my-sec-nokms-1507
HBase Master:	vp-os-rh6-my-sec-nokms-1507
Oozie Server:	vp-os-rh6-my-sec-nokms-1507
ZooKeeper Server:	vp-os-rh6-my-sec-nokms-1507
ZooKeeper Server:	vp-os-rh6-my-sec-nokms-1507
ZooKeeper Server:	vp-os-rh6-my-sec-nokms-1507
Nimbus:	vp-os-rh6-my-sec-nokms-1507
DRPC Server:	vp-os-rh6-my-sec-nokms-1507
Storm UI Server:	vp-os-rh6-my-sec-nokms-1507
Kafka Broker:	vp-os-rh6-my-sec-nokms-1507
Kafka Broker:	vp-os-rh6-my-sec-nokms-1507
Kafka Broker:	vp-os-rh6-my-sec-nokms-1507
Kafka Broker:	vp-os-rh6-my-sec-nokms-1507
Metrics Collector:	vp-os-rh6-my-sec-nokms-1507
Ranger KMS Server:	vp-os-rh6-my-sec-nokms-1507
Knox Gateway:	
Ranger Admin:	vp-os-rh6-my-sec-nokms-1507
Ranger Usersync:	vp-os-rh6-my-sec-nokms-1507

vp-os-rh6-my-sec-nokms-150727-1736-2.openstacklocal (15.6 GB, 2 cores)
vp-os-rh6-my-sec-nokms-150727-1736-3.openstacklocal (15.6 GB, 2 cores)
vp-os-rh6-my-sec-nokms-150727-1736-4.openstacklocal (15.6 GB, 2 cores)
vp-os-rh6-my-sec-nokms-150727-1736-5.openstacklocal (15.6 GB, 2 cores)

vp-os-rh6-my-sec-nokms-150727-1736-4.openstacklocal (15.6 GB, 2 cores)
NameNode, History Server, App Timeline Server, HBase Master, ZooKeeper Server, DRPC Server, Storm UI Server, Kafka Broker

vp-os-rh6-my-sec-nokms-150727-1736-5.openstacklocal (15.6 GB, 2 cores)
Kafka Broker, Ranger Admin, Ranger Usersync

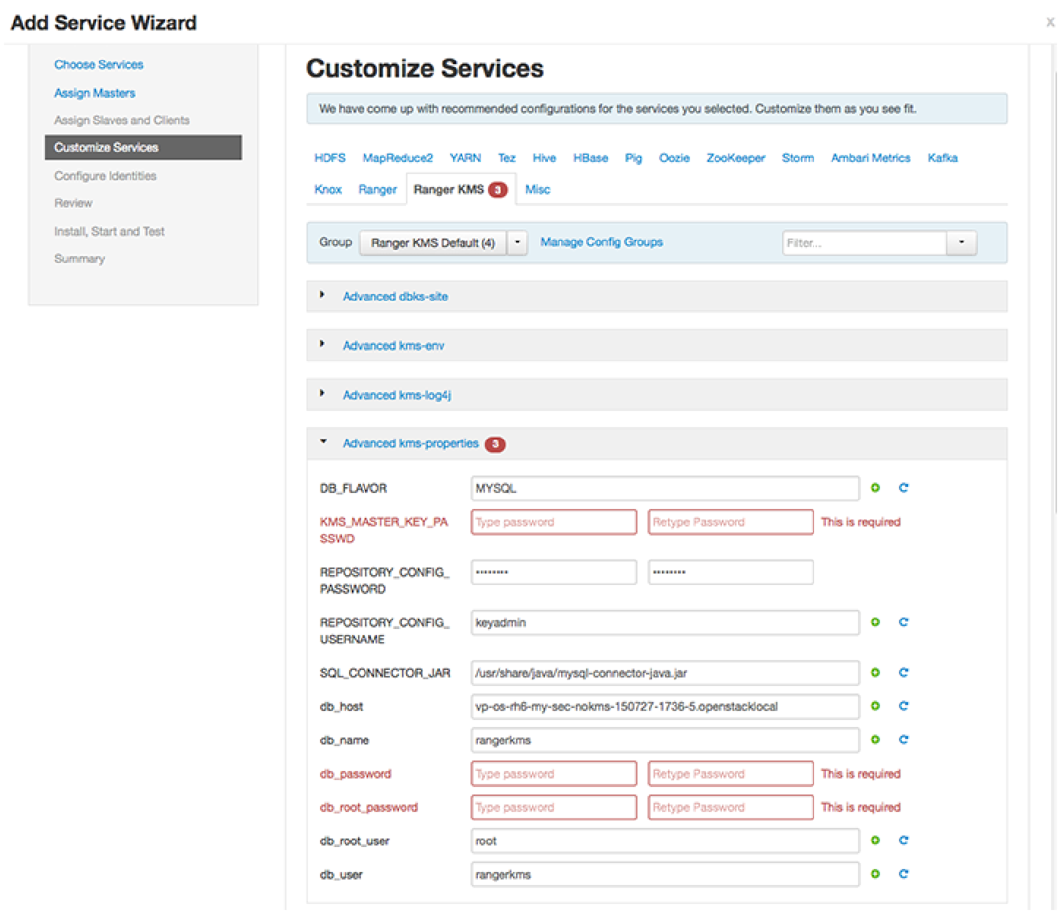
6. In **Customize Services**, set required values (marked in red). Review other configuration settings, and determine whether you'd like to change any of the default values. (For more information about these properties, see Ranger KMS Properties.)

   a. Set the following required settings, marked in red in the "Advanced kms-properties" section:

      - `KMS_MASTER_KEY_PASSWD`
      - `db_password`
      - `db_root_password`

Note

If do not wish to provide system Database Administrator (DBA) account details to the Ambari Ranger installer, you can use the `dba_script.py` Python script to create Ranger DB database users without exposing DBA account information to the Ambari Ranger installer. For more information, see Setting up Database Users Without Sharing DBA Credentials.



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 MapReduce2 YARN Tez Hive HBase Pig Oozie ZooKeeper Storm Ambari Metrics Kafka Knox Ranger Ranger KMS 3 Misc

Group Ranger KMS Default (4) Manage Config Groups

Advanced dbks-site

Advanced kms-env

Advanced kms-log4j

Advanced kms-properties 3

DB_FLAVOR	MYSQL		
KMS_MASTER_KEY_PASSWD	Type password	Retype Password	This is required
REPOSITORY_CONFIG_PASSWORD	••••••••	••••••••	
REPOSITORY_CONFIG_USERNAME	keyadmin		
SQL_CONNECTOR_JAR	/usr/share/java/mysql-connector-java.jar		
db_host	vp-os-rh6-my-sec-nokms-150727-1736-5.openstacklocal		
db_name	rangerkms		
db_password	Type password	Retype Password	This is required
db_root_password	Type password	Retype Password	This is required
db_root_user	root		
db_user	rangerkms		

Also specify the username for REPOSITORY_CONFIG_USERNAME, so that Ranger will be able to connect to the Ranger KMS Server and look up keys for creating access policies. This user will need to be set to proxy into Ranger KMS in a Kerberos mode (steps included below).

b. Add values for the following properties in the "Custom kms-site" section. These properties allow the specified system users (`hive`, `oozie`, and others) to proxy on behalf of other users when communicating with Ranger KMS. This helps individual services (such as Hive) use their own keytabs, but retain the ability to access Ranger KMS as the end user (use access policies associated with the end user).

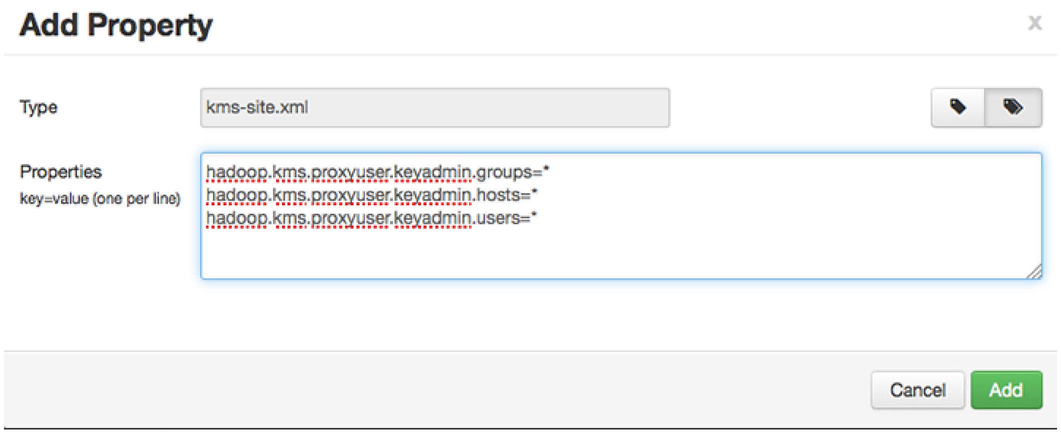
- `hadoop.kms.proxyuser.hive.users`
- `hadoop.kms.proxyuser.oozie.users`
- `hadoop.kms.proxyuser.HTTP.users`
- `hadoop.kms.proxyuser.ambari.users`
- `hadoop.kms.proxyuser.yarn.users`
- `hadoop.kms.proxyuser.hive.hosts`

- `hadoop.kms.proxyuser.oozie.hosts`
- `hadoop.kms.proxyuser.HTTP.hosts`
- `hadoop.kms.proxyuser.ambari.hosts`
- `hadoop.kms.proxyuser.yarn.hosts`

c. Add the following properties to the Custom KMS-site section of the configuration. These properties use the REPOSITORY_CONFIG_USERNAME specified in the first step in this section.

If you are using an account other than `keyadmin` to access Ranger KMS, replace “keyadmin” with the configured user for the Ranger KMS repository in Ranger admin:

- `hadoop.kms.proxyuser.keyadmin.groups=*`
- `hadoop.kms.proxyuser.keyadmin.hosts=*`
- `hadoop.kms.proxyuser.keyadmin.users=*`



Type: kms-site.xml

```
hadoop.kms.proxyuser.keyadmin.groups=*
hadoop.kms.proxyuser.keyadmin.hosts=*
hadoop.kms.proxyuser.keyadmin.users=*
```

d. Confirm settings of the following values in the "advanced kms-site" group:

- `hadoop.kms.authentication.type=kerberos`
- `hadoop.kms.authentication.kerberos.keytab=/etc/security/keytabs/spnego.service.keytab`
- `hadoop.kms.authentication.kerberos.principal=*`

7. Then, choose **Next**.

8. Review the default values on the **Configure Identities** screen. Determine whether you'd like to change any of the default values. Then, choose **Next**.

9. In **Review**, make sure the configuration values are correct. Ranger KMS will be listed under **Services**.

10. Then, choose **Deploy**.

11. Monitor the progress of installing, starting, and testing the service. When the service installs and starts successfully, choose **Next**.

12. The Summary screen displays the results. Choose **Complete**.

13. Restart the Ranger and Ranger KMS services.

2.1.1. Setting up Database Users Without Sharing DBA Credentials

If do not wish to provide system Database Administrator (DBA) account details to the Ambari Ranger installer, you can use the `dba_script.py` Python script to create Ranger DB database users without exposing DBA account information to the Ambari Ranger installer. You can then run the normal Ambari Ranger installation without specify a DBA user name and password.

To create Ranger DB users using the `dba_script.py` script:

1. Download the Ranger rpm using the yum install command.

   ```
   yum install ranger-kms
   ```

2. You should see one file named `dba_script.py` in the `/usr/hdp/current/ranger-admin` directory.

3. Get the script reviewed internally and verify that your DBA is authorized to run the script.

4. Execute the script by running the following command:

   ```
   python dba_script.py
   ```

5. Pass all values required in the argument. These should include `db flavor`, `JDBC jar`, `db host`, `db name`, `db user`, and other parameters.

   - If you would prefer not to pass runtime arguments via the command prompt, you can update the `/usr/hdp/current/ranger-admin/install.properties` file and then run:

   -
     ```
     python dba_script.py -q
     ```

     When you specify the `-q` option, the script will read all required information from the `install.properties` file

   - You can use the `-d` option to run the script in "dry" mode. Running the script in dry mode causes the script to generate a database script.

     ```
     python dba_script.py -d /tmp/generated-script.sql
     ```

     Anyone can run the script, but it is recommended that the system DBA run the script in dry mode. In either case, the system DBA should review the generated script, but should only make minor adjustments to the script, for example, change the location of a particular database file. No major changes should be made that substantially alter the script – otherwise the Ranger install may fail.

The system DBA must then run the generated script.

6. Log in to the host where KMS is to be installed. Run the following commands to back up files:

   ```
   cp /var/lib/ambari-agent/cache/common-services/RANGER_KMS/0.5.0.2.3/package/
   scripts/kms.py /var/lib/ambari-agent/cache/common-services/RANGER_KMS/0.5.0.
   2.3/package/scripts/kms.py.bak
   cp /var/lib/ambari-server/resources/common-services/RANGER_KMS/0.5.0.2.3/
   package/scripts/kms.py /var/lib/ambari-server/resources/common-services/
   RANGER_KMS/0.5.0.2.3/package/scripts/kms.py.bak
   ```

7. In both of the `kms.py` files copied in the previous step, find and comment out the following line (shown here commented out).

   ```
   #Execute(dba_setup, environment=env_dict, logoutput=True, user=params.
   kms_user)
   ```

8. Run the Ranger Ambari install procedure, but set **Setup Database and Database User** to **No** in the Ranger Admin section of the Customize Services screen.

2.1.2. Configure HDFS Encryption to use Ranger KMS Access

At this point, Ranger KMS should be installed and running. If you plan to use Ranger KMS for HDFS data at rest encryption, complete the following steps:

1. Create a link to `/etc/hadoop/conf/core-site.xml` under `/etc/ranger/kms/conf`:

   ```
   sudo ln -s /etc/hadoop/conf/core-site.xml /etc/ranger/kms/conf/
   core-site.xml
   ```

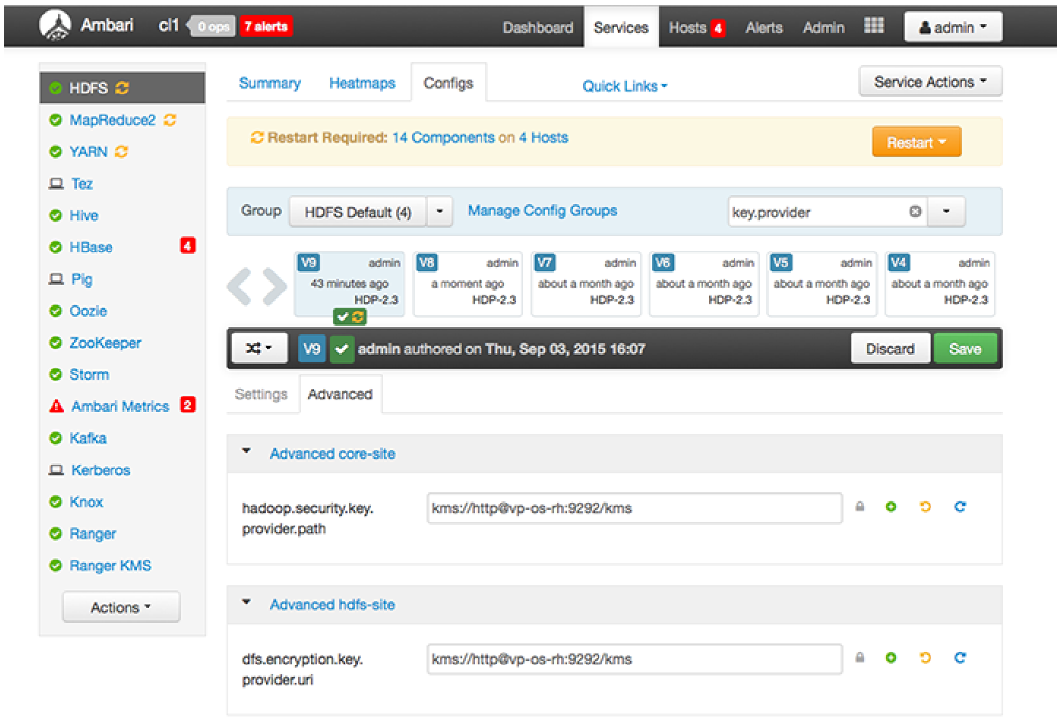
2. Configure HDFS to access Ranger KMS.

   a. In the left panel of the Ambari main menu, choose HDFS.

   b. Choose the **Configs** tab at the top of the page, and then choose the **Advanced** tab partway down the page.

   c. Specify the provider path (the URL where the Ranger KMS server is running) in the following two properties, if the path is not already specified:

      - In "Advanced core-site", specify `hadoop.security.key.provider.path`
      - In "Advanced hdfs-site", specify `dfs.encryption.key.provider.uri`



hadoop.security.key.provider.path kms://http@vp-os-rh:9292/kms

dfs.encryption.key.provider.uri kms://http@vp-os-rh:9292/kms

The Ranger KMS host is where Ranger KMS is installed. The Ranger KMS host name should have the following format:

```
kms://http@<kmshost>:9292/kms
```

3. Under Custom core-site.xml, set the value of the `hadoop.proxyuser.kms.groups` property to `*` or service user.

4. Restart the Ranger KMS service and the HDFS service.

2.1.3. Use a Kerberos Principal for the Ranger KMS Repository

In Ranger, all access policies are configured within a repository for each service. For more information, refer to the Ranger User Guide.

To manage access policies for Ranger KMS, a repository is needed with Ranger for the Ranger KMS service. Ambari creates the repository automatically using the repository config user and password provided.

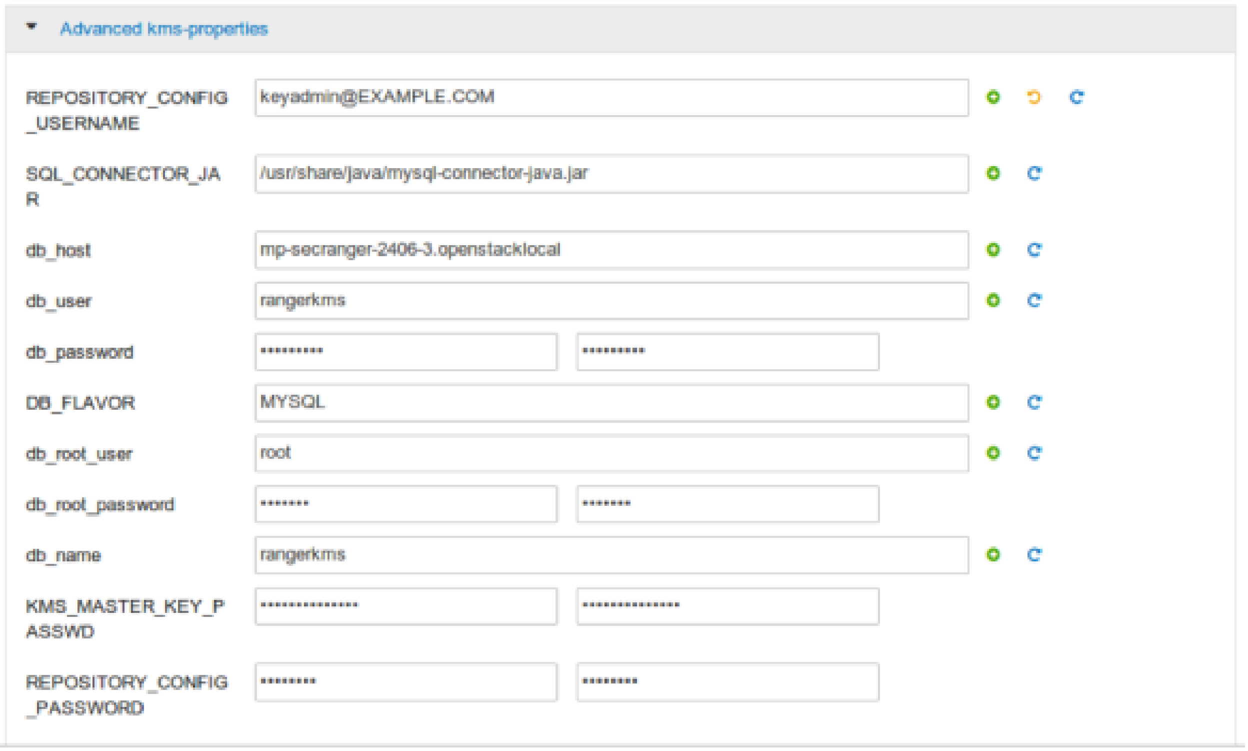
The repository config user also needs to be created as a principal in Kerberos with a password. Use the following steps to use a Kerberos principla for the Ranger KMS repository.

1. Create system user `keyadmin` which should be sync in User Tabs in Ranger Admin.

2. Create principal `keyadmin@EXAMPLE.COM` with password `keyadmin`:

```
kadmin.local -q 'addprinc -pw keyadmin keyadmin'
```

3. On the Add Service wizard Customize Services page, set the required values (marked in red).
4. Under ranger-kms-properties, set the principal and password in the REPOSITORY_CONFIG_USERNAME and REPOSITORY_CONFIG_PASSWORD fields.
5. To check logs, select **Audit to DB** under Advanced ranger-kms-audit.
6. Click **Next** to continue with the Ranger KMS Add Service wizard.



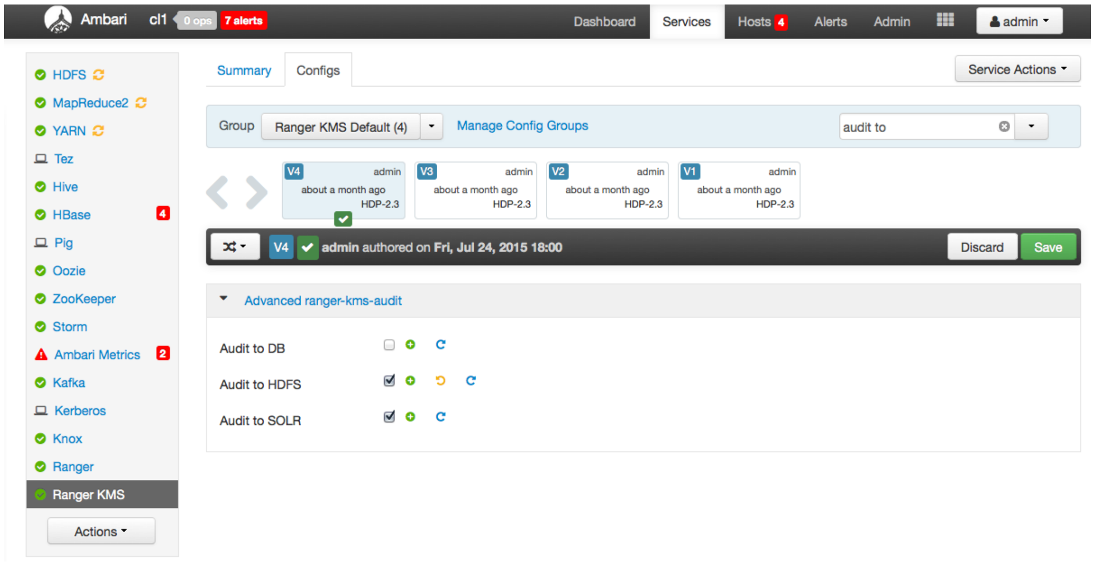
REPOSITORY_CONFIG_USERNAME	keyadmin@EXAMPLE.COM	
SQL_CONNECTOR_JAR	/usr/share/java/mysql-connector-java.jar	
db_host	mp-secranger-2406-3.openstacklocal	
db_user	rangerkms	
db_password	••••••••••	••••••••••
DB_FLAVOR	MYSQL	
db_root_user	root	
db_root_password	•••••••	•••••••
db_name	rangerkms	
KMS_MASTER_KEY_PASSWD	•••••••••••••••	•••••••••••••••
REPOSITORY_CONFIG_PASSWORD	•••••••••	•••••••••

3. Enable Ranger KMS Audit

Ranger KMS supports audit to DB, HDFS, and Solr. Solr is well-suited for short-term auditing and UI access (for example, one month of data accessible via quick queries in the Web UI). HDFS is typically used for archival auditing. They are not mutually exclusive; we recommend configuring audit to both Solr and HDFS.

First, make sure Ranger KMS logs are enabled:

1. Go to the Ambari UI: `http://<gateway>:8080`
2. Select `ranger-kms` from the service.
3. Click the Configs tab, and go to the accordion menu.
4. In the Advanced ranger-kms-audit list, set `xasecure.audit.is.enabled` to true.
5. Select "Audit to Solr" and/or "Audit to HDFS", depending on which database(s) you plan to use:



Audit to DB ☐

Audit to HDFS ☑

Audit to SOLR ☑

6. Save the configuration and restart the Ranger KMS service.

Next, check to see if the Ranger KMS Plugin is enabled:

1. Go to the Ranger UI: `http://<gateway>:6080`
2. Login with your keyadmin user ID and password (the defaults are `keyadmin`, `keyadmin`). The default repository will be added under KMS service.
3. Run a test connection for the service. You should see a ‘connected successfully’ popup message. If the connection is not successful, make sure that the configured user exists (in KDC for a secure cluster).
4. Choose the Audit > Plugin tab.

5. Check whether plugins are communicating. The UI should display `Http Response code 200` for the respective plugin.

The next two subsections describe how to save audit to Solr and HDFS.

3.1. Save Audit to Solr

Prerequisite: To save Ranger KMS audits to Solr, you must have Solr installed and running.

To save audits to Solr:

1. Edit the following Ranger properties in the Advanced ranger-admin.site list:

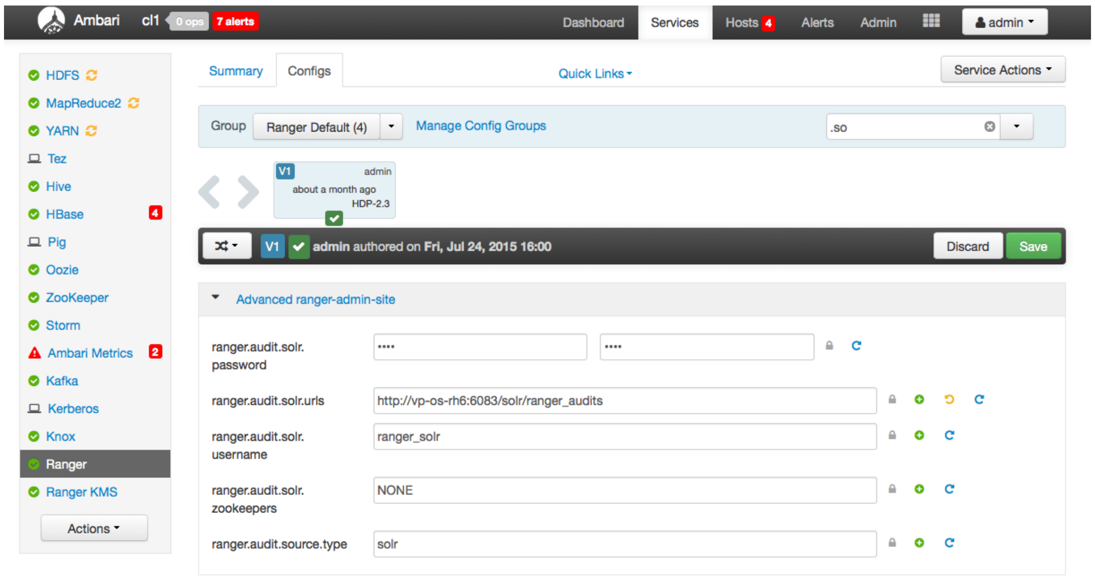
   ```
   ranger.audit.solr.password = NONE
   ```

   ```
   ranger.audit.solr.urls = http://solr_host:6083/solr/
   ranger_audits
   ```

   ```
   ranger.audit.solr.username = ranger_solr
   ```

   ```
   ranger.audit.source.type = solr
   ```

   For example:

   V1 admin authored on Fri, Jul 24, 2015 16:00

ranger.audit.solr.password	••••
ranger.audit.solr.urls	http://vp-os-rh6:6083/solr/ranger_audits
ranger.audit.solr.username	ranger_solr
ranger.audit.solr.zookeepers	NONE
ranger.audit.source.type	solr

2. Restart Ranger.

3. Next, to enable Ranger KMS auditing to Solr, set the following properties in the Advanced ranger-kms-audit list:

   a. Check the box next to `Enable audit to solr` in the Ranger KMS component.

   b. Check the `Audit provider summary enabled` box, and make sure that `xasecure.audit.is.enabled` is set to true.

   c. Restart Ranger KMS.

Note

Check audit logs on Ranger UI, to make sure that they are getting through Solr: `http://RANGER_HOST_NAME:6080/index.html#!/reports/audit/bigData` or `http://solr_host:6083/solr/ranger_audits`.

3.2. Save Audit to HDFS

There are no configuration changes needed for Ranger properties.

To save Ranger KMS audits to HDFS, set the following properties in the Advanced ranger-kms-audit list.

Note: the following configuration settings must be changed in each Plugin.

1. Check the box next to `Enable Audit to HDFS` in the Ranger KMS component.
2. Set the HDFS path to the path of the location in HDFS where you want to store audits:

   ```
   xasecure.audit.destination.hdfs.dir = hdfs://NAMENODE_FQDN:8020/
   ranger/audit
   ```
3. Check the `Audit provider summary enabled` box, and make sure that `xasecure.audit.is.enabled` is set to true.
4. Make sure that the plugin's root user (`kms`) has permission to access HDFS Path `hdfs://NAMENODE_FQDN:8020/ranger/audit`
5. Restart Ranger KMS.
6. Generate audit logs for the Ranger KMS.
7. (**Optional**) To verify audit to HDFS without waiting for the default sync delay (approximately 24 hours), restart Ranger KMS. Ranger KMS will start writing to HDFS after the changes are saved post-restart.

To check for audit data:

```
hdfs dfs -ls /ranger/audit/
```

To test Ranger KMS audit to HDFS, complete the following steps:

1. Under custom core-site.xml, set `hadoop.proxyuser.kms.groups` to “*” or to the service user.
2. In the custom kms-site file, add `hadoop.kms.proxyuser.keyadmin.users` and set its value to "*". (If you are not using keyadmin to access Ranger KMS Admin, replace “keyadmin” with the user account used for authentication.)
3. In the custom kms-site file, add `hadoop.kms.proxyuser.keyadmin.hosts` and set its value to "*". (If you are not using keyadmin to access Ranger KMS Admin, replace “keyadmin” with the user account used for authentication.)

4. Copy the core-site.xml to the component's class path (`/etc/ranger/kms/conf`)

   OR

   link to `/etc/hadoop/conf/core-site.xml` under `/etc/ranger/kms/conf` (`ln -s /etc/hadoop/conf/core-site.xml /etc/ranger/kms/conf/core-site.xml`)

5. Verify the service user principal. (For Ranger KMS it will be the `http` user.)

6. Make sure that the component user has permission to access HDFS. (For Ranger KMS the `http` user should also have permission.)

4. Enabling SSL for Ranger KMS

If you do not have access to Public CA-issued certificates, complete the following steps to create and configure self-signed certificates.

Note

The following examples contain sample values (folder locations, passwords, and filenames). Change these values according to your environment.

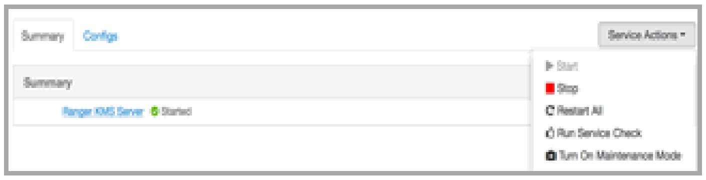
Considerations:

- Copy `keystore/truststore` files into a different location (e.g. `/etc/security/serverKeys`) than the `/etc/<component>/conf` folders.
- Make sure JKS file names are different from each other.
- Make sure correct permissions are applied.
- Make sure all passwords are secured.
- For the test connection to be successful after enabling SSL, self-signed certificates should be imported to the Ranger admin's trust store (typically JDK `cacerts`).
- Property `ranger.plugin.service.policy.rest.ssl.config.file` should be verified; for example:

  `ranger.plugin.kms.policy.rest.ssl.config.file` ==> `/etc/ranger/kms/conf/ranger-policymgr-ssl.xml`

To enable SSL:

1. Stop the Ranger KMS service:

2. Go to the Ranger KMS (and plugin) installation location, and create a self-signed certificate:

   ```
   cd /etc/ranger/kms/conf/
   ```

   ```
   keytool -genkey -keyalg RSA -alias rangerKMSAgent -keystore
   <ranger-kms-ks> -storepass myKeyFilePassword -validity 360 -
   keysize 2048
   ```

   ```
   chown kms:kms <ranger-kms-ks>
   ```

   ```
   chmod 400 <ranger-kms-ks>
   ```

   where

`<ranger-kms-ks>` is the name of the Ranger KMS keystore (for example, `ranger-plugin-keystore.jks`)

3. Provide an identifiable string in response to the question "What is your first and last name?"

   Important: In case multiple servers need to communicate with Ranger admin for downloading policies for the same service/repository, make sure to use the repo name or a common string across all nodes. Remember exactly what you entered, because this value will be required for the Common Name for Certificate field on the edit repository page in the policy manager UI.

   To create the keystore, provide answers to the subsequent questions. **Note:** Press enter when prompted for a password.

4. Create a truststore for the Ranger KMS plugin, and add the public key of admin as a trusted entry into the truststore:

   ```
   cd /etc/ranger/kms/conf/
   ```

   ```
   keytool -export -keystore <ranger-admin-ks> -alias rangeradmin -
   file <cert-filename>
   ```

   ```
   keytool -import -file <cert-filename> -alias rangeradmintrust -
   keystore <ranger-kms-ts> -storepass changeit
   ```

   ```
   chown kms:kms <ranger-kms-ts>
   ```

   ```
   chmod 400 <ranger-kms-ts>
   ```

   where

   `<ranger-admin-ks>` is the location of the Ranger Admin keystore (for example, `/etc/ranger/admin/conf/ranger-admin-keystore.jks`)

   `<ranger-kms-ts>` is the name of the Ranger KMS plugin trustore (for example, `ranger-plugin-truststore.jks`)

   `<cert-filename>` is the name of the Ranger Admin certificate file (for example, `ranger-admin-trust.cer`)

   Note: Press enter when prompted for a password.

5. Change the policy manager URL to point to HTTPS, and specify the keystore & truststore in `ews/webapp/WEB-INF/classes/conf/ranger-policymgr-ssl.xml`.

   a. In `xasecure.policymgr.clientssl.keystore`, provide the location for the keystore that you created in the previous step.

   b. In `xasecure.policymgr.clientssl.keystore.password`, provide the password for the keystore (myKeyFilePassword).

   c. In `xasecure.policymgr.clientssl.truststore`, provide the location for the truststore that you created in the previous step.

   d. In `xasecure.policymgr.clientssl.truststore.password`, provide the password for the truststore (changeit).

6. Add the plugin's self-signed cert into Admin's trustedCACerts:

   ```
   cd /etc/ranger/admin/conf
   ```

   ```
   keytool -export -keystore <ranger-kms-ks> -alias rangerKMSAgent
   -file <cert-filename> -storepass myKeyFilePassword
   ```

   ```
   keytool -import -file <cert-filename> -alias rangerkmsAgentTrust
   -keystore <ranger-admin-ts> -storepass changeit
   ```

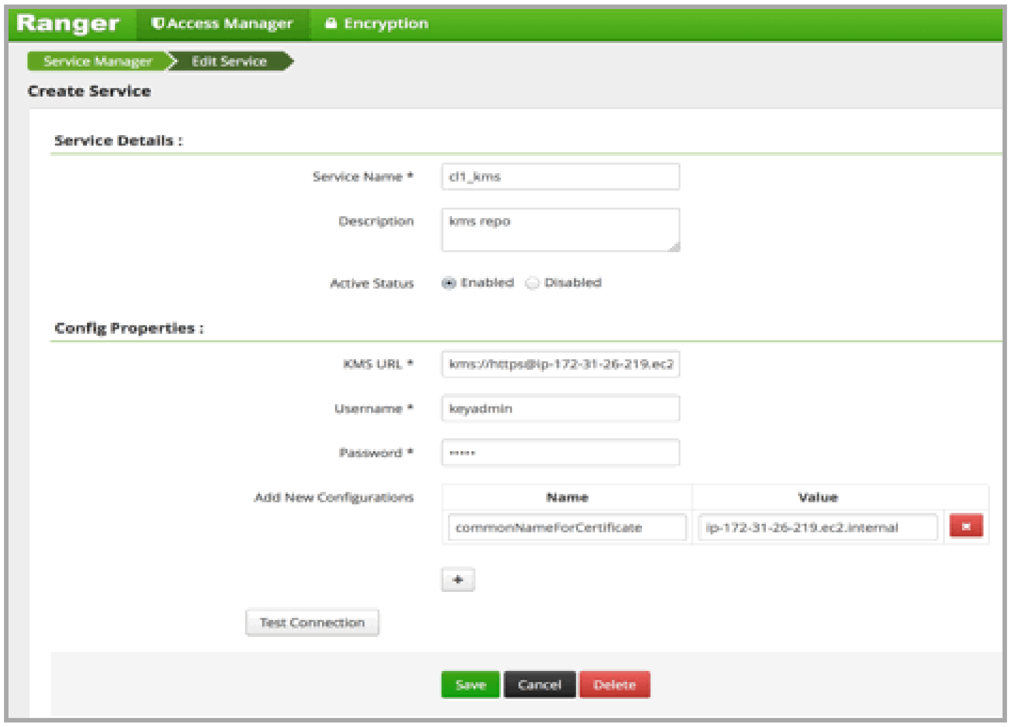
   where

   `<ranger-kms-ks>` is the path to the Ranger KMS keystore (for example, `/etc/ranger/kms/conf/ranger-plugin-keystore.jks`)

   `<cert-filename>` is the name of the certificate file (for example, `ranger-kmsAgent-trust.cer`)

   `<ranger-admin-ts>` is the name of the Ranger Admin truststore file (for example, the JDK cacerts file)

7. Log into the Policy Manager UI (as `keyadmin` user) and click on the Edit button of your KMS repository. Provide the CN name of the keystore for **Common Name For Certificate** (`commonNameForCertificate`), and save it. This property is not added by default.



Service Name *	cl1_kms
Description	kms repo
Active Status	[✓ Enabled] Disabled
KMS URL *	kms://https@ip-172-31-26-219.ec2
Username *	keyadmin
Password *	•••••

Add New Configurations

Name	Value
commonNameForCertificate	ip-172-31-26-219.ec2.internal

Configuring the Ranger KMS Server

1. Go to the Ranger KMS config location and create a self-signed certificate:

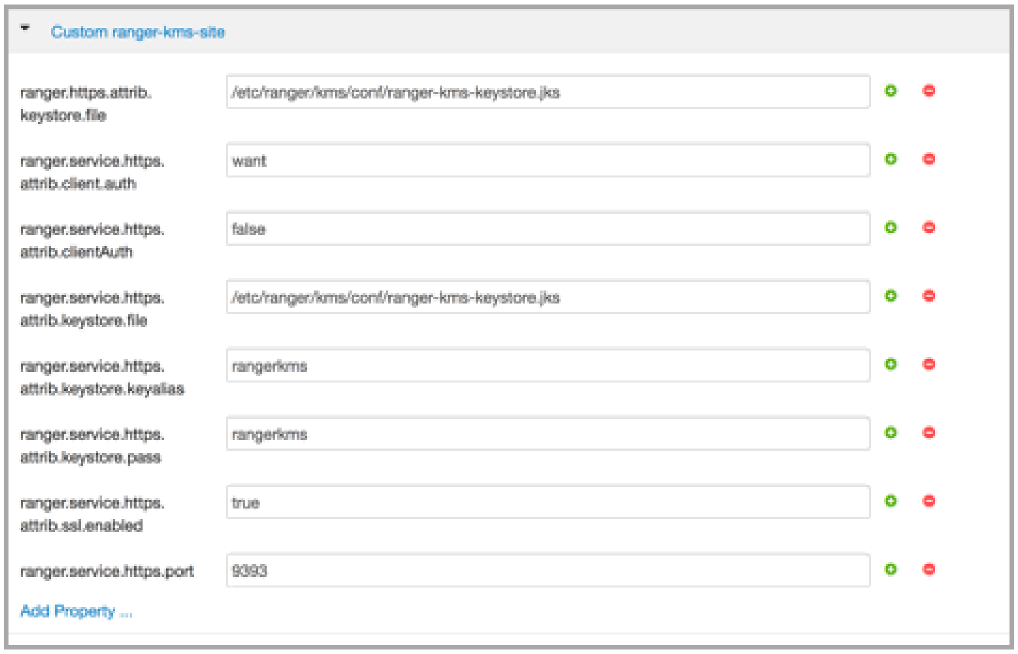
   ```
   cd /etc/ranger/kms/conf

   keytool -genkey -keyalg RSA -alias rangerkms -keystore <ranger-
   kms-ks> -storepass rangerkms -validity 360 -keysize 2048

   chown kms:kms ranger-kms-keystore.jks

   chmod 400 ranger-kms-keystore.jks
   ```

   where

   `<ranger-kms-ks>` is the name of the Ranger KMS keystore (for example, `ranger-plugin-keystore.jks`)

   Provide an identifiable string in response to the question "What is your first and last name?" To create the keystore, provide answers to all subsequent questions to create the keystore **Note**: Press enter when prompted for a password.

2. Add the following properties and values to the Custom ranger-kms-site list:



Custom ranger-kms-site

ranger.https.attrib.keystore.file	/etc/ranger/kms/conf/ranger-kms-keystore.jks
ranger.service.https.attrib.client.auth	want
ranger.service.https.attrib.clientAuth	false
ranger.service.https.attrib.keystore.file	/etc/ranger/kms/conf/ranger-kms-keystore.jks
ranger.service.https.attrib.keystore.keyalias	rangerkms
ranger.service.https.attrib.keystore.pass	rangerkms
ranger.service.https.attrib.ssl.enabled	true
ranger.service.https.port	9393

3. Update the value of `kms_port` (in Advanced kms_env) to the `ranger.service.https.port` value.

4. Save your changes and start Ranger KMS.

5. In your browser (or from Curl) when you access the Ranger KMS UI using the HTTPS protocol on the `ranger.service.https.port` listed in Ambari, the browser should respond that it does not trust the site. Proceed, and you should be able to access Ranger KMS on HTTPS with the self-signed cert that you just created.

6. Export the Ranger KMS certificate:

```
cd /usr/hdp/<version>/ranger-kms/conf
```

```
keytool -export -keystore <ranger-kms-ks> -alias rangerkms -file
<cert-filename>
```

where

`<ranger-kms-ks>` is the name of the Ranger KMS keystore (for example, `ranger-kms-keystore.jks`)

`<cert-filename>` is the name of the certificate file (for example, `ranger-kms-trust.cer`)

7. Import the Ranger KMS certificate into the Ranger admin truststore:

   ```
   keytool -import -file <cert-filename> -alias rangerkms -keystore
   <ranger-admin-ts> -storepass changeit
   ```

   where

   `<cert-filename>` is the name of the certificate file (for example, `ranger-kms-trust.cer`)

   `<ranger-admin-ts>` is the name of the Ranger Admin truststore file (for example, JDK cacerts)

8. Import the Ranger KMS certificate into the Hadoop client truststore:

   ```
   keytool -import -file <cert-filename> -alias rangerkms -keystore
   <ts-filename> -storepass bigdata
   ```

   where

   `<cert-filename>` is the name of the certificate file (for example, `ranger-kms-trust.cer`)

   `<ts-filename>` is the name of Hadoop client truststore file (for example, `/etc/security/clientKeys/all.jks`)

9. Restart Ranger Admin and Ranger KMS.

10. Now in the Policy Manager UI, Audit –> Plugin tab, you should see an entry for your service name with HTTP Response Code = 200.

5. Install Multiple Ranger KMS

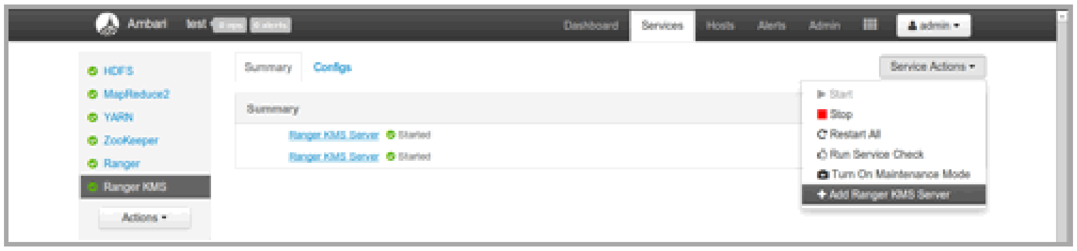
Multiple services can be set up for high availability of Ranger KMS. HDFS interacts with the active process.

Prerequisite: an instance with more than one node.

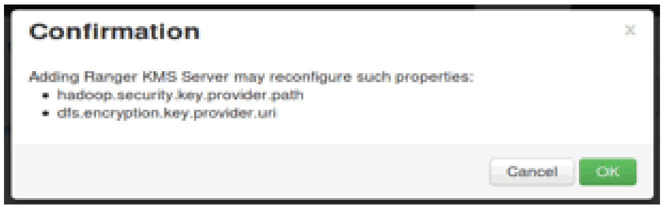
To install Ranger KMS on multiple nodes:

1. First install Ranger KMS on a single node (see Installing the Ranger Key Management Service).
2. Next, add the Ranger KMS service to another node.

   In the Ambari Web UI for the additional node, go to Ranger KMS service # Summary # Service Actions # Add Ranger KMS server.

3. After adding Ranger KMS server, Ambari will show a pop-up message.
4. Press OK. Ambari will modify two HDFS properties, `hadoop.security.key.provider.path` and `dfs.encryption.key.provider.uri`.
5. Restart the HDFS service:

   Confirmation

   Adding Ranger KMS Server may reconfigure such properties:
   - hadoop.security.key.provider.path
   - dfs.encryption.key.provider.uri

6. For the Ranger KMS service, go to the Advanced kms-site list and change the following property values:

   ```
   hadoop.kms.cache.enable=false
   ```

   ```
   hadoop.kms.cache.timeout.ms=0
   ```

   ```
   hadoop.kms.current.key.cache.timeout.ms=0
   ```

   ```
   hadoop.kms.authentication.signer.secret.provider=zookeeper
   ```

   ```
   hadoop.kms.authentication.signer.secret.provider.zookeeper.connection.string
   ip of first node}:2181,{internal ip of second node}:2181, ...
   ```

```
hadoop.kms.authentication.signer.secret.provider.zookeeper.auth.type=none
```

7. Save your configuration changes and restart the Ranger KMS service.

Next, check connectivity from Ranger admin for the newly-added Ranger KMS server:

1. Go to the Ranger UI: `http://<gateway>:6080`
2. Login with your keyadmin user ID and password (the defaults are `keyadmin`, `keyadmin`; these should be changed as soon as possible after installation). The default repository will be added under Ranger KMS service.
3. Under Config properties of the Ranger KMS URL, add the newly added Ranger KMS server FQDN. For example:

   Previous Ranger KMS URL = `kms://http@<internal host name>:9292/kms`

   New Ranger KMS URL = `kms://http@<internal host name1>;<internal host name2>;...:9292/kms`
4. Run a test connection for the service. You should see a 'connected successfully' message.
5. Choose the Audit > Plugin tab.
6. Check whether plugins are communicating. The UI should display HTTP Response Code = 200 for the respective plugin.

6. Using the Ranger Key Management Service

Ranger KMS can be accessed at the Ranger admin URL, `http://<hostname>:6080`. Note, however, that the login user for Ranger KMS is different than that for Ranger. Logging on as the Ranger KMS admin user leads to a different set of screens.

Role Separation

By default, Ranger admin uses a different admin user (`keyadmin`) to manage access policies and keys for Ranger KMS.

The person accessing Ranger KMS via the `keyadmin` user should be a different person than the administrator who works with regular Ranger access policies. This approach separates encryption work (encryption keys and policies) from Hadoop cluster management and access policy management.

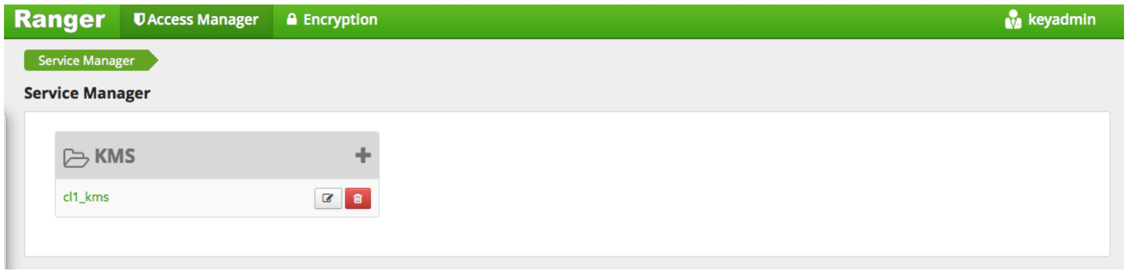
6.1. Accessing the Ranger KMS Web UI

To access Ranger KMS, log in as user `keyadmin`, password `keyadmin`.

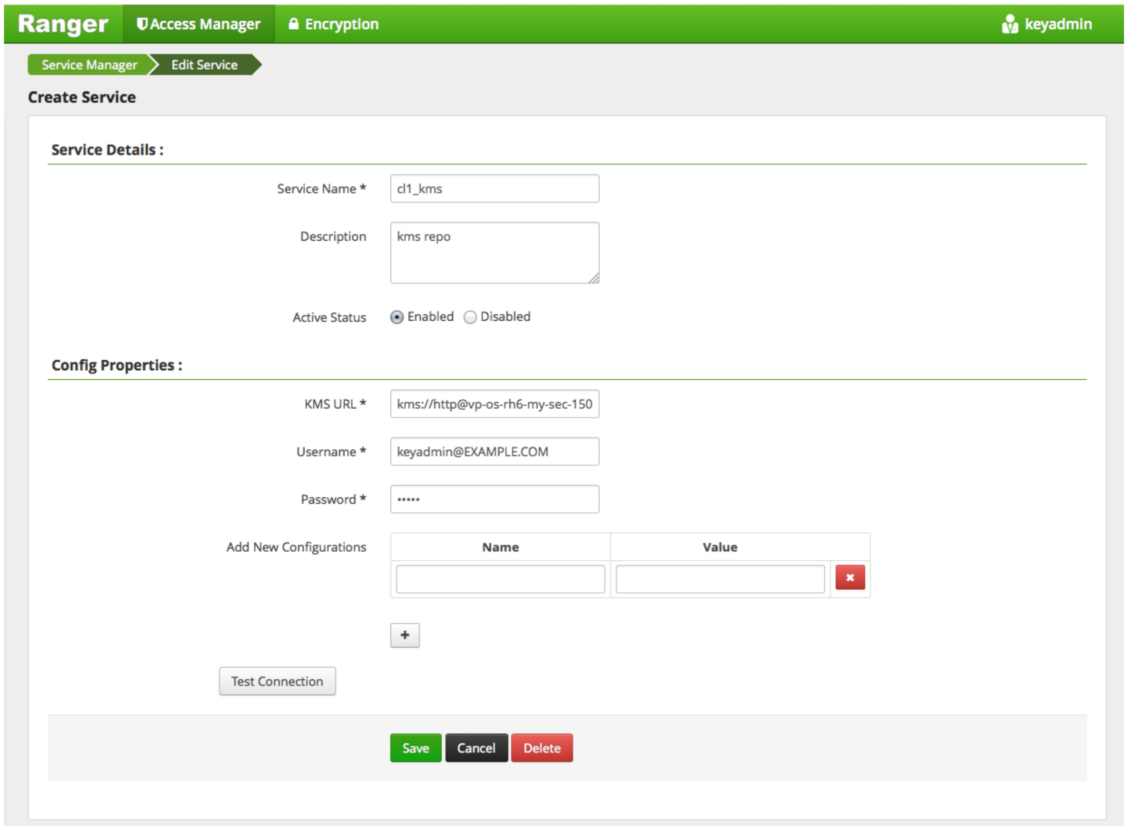
Important

Change the password after you log in.

After logging in, you will see the Service Manager screen. To view or edit Ranger KMS repository properties, click on the edit button next to the repository name:



You will see a list of service details and config properties for the repository:



Create Service

Service Details :

Service Name *	cl1_kms
Description	kms repo
Active Status	[✓ Enabled] Disabled

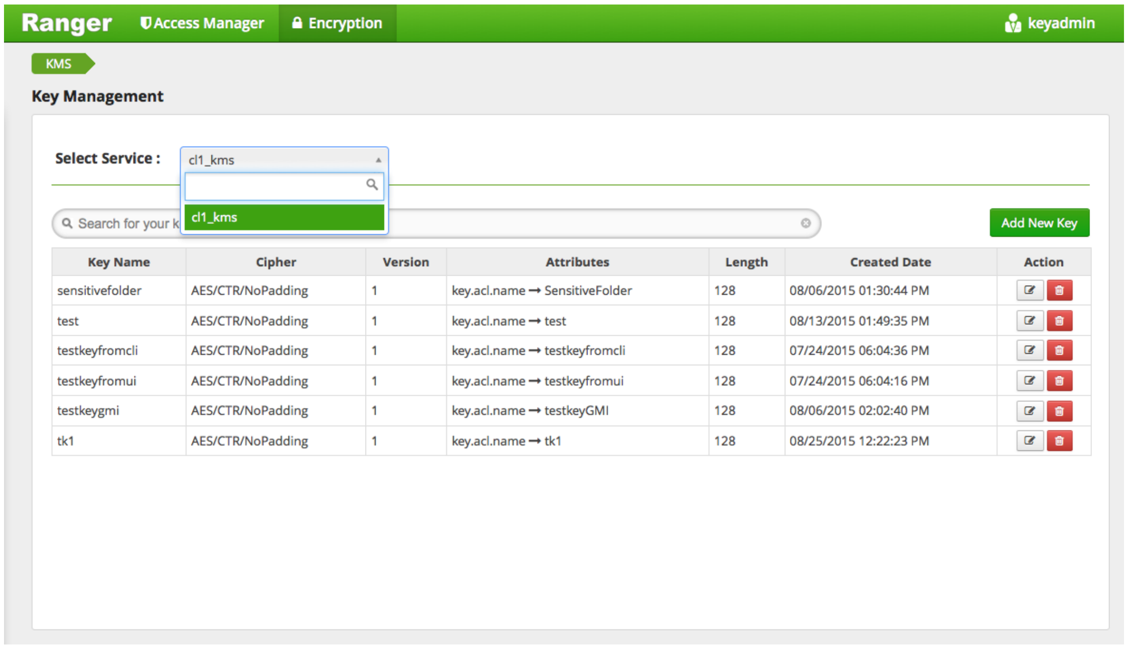
Config Properties :

KMS URL *	kms://http@vp-os-rh6-my-sec-150
Username *	keyadmin@EXAMPLE.COM
Password *	•••••
Add New Configurations	Name / Value

6.2. Listing and Creating Keys

To list existing keys:

1. Choose the Encryption tab at the top of the Ranger Web UI screen.
2. Select the Ranger KMS service from the drop-down list.



Key Management

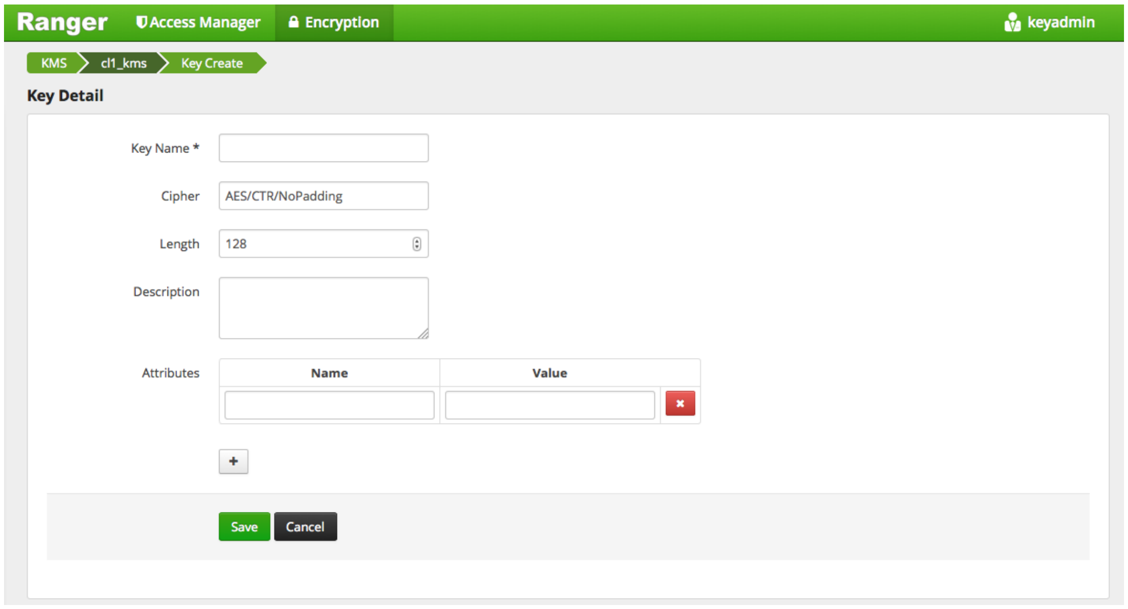
Select Service : cl1_kms

Search for your k

Key Name	Cipher	Version	Attributes	Length	Created Date	Action
sensitivefolder	AES/CTR/NoPadding	1	key.acl.name → SensitiveFolder	128	08/06/2015 01:30:44 PM	
test	AES/CTR/NoPadding	1	key.acl.name → test	128	08/13/2015 01:49:35 PM	
testkeyfromcli	AES/CTR/NoPadding	1	key.acl.name → testkeyfromcli	128	07/24/2015 06:04:36 PM	
testkeyfromui	AES/CTR/NoPadding	1	key.acl.name → testkeyfromui	128	07/24/2015 06:04:16 PM	
testkeygmi	AES/CTR/NoPadding	1	key.acl.name → testkeyGMI	128	08/06/2015 02:02:40 PM	
tk1	AES/CTR/NoPadding	1	key.acl.name → tk1	128	08/25/2015 12:22:23 PM	

To create a new key:

1. Click on "Add New Key".
2. Add a valid key name.
3. Select the cipher name. Ranger supports AES/CTR/NoPadding as the cipher suite.
4. Specify the key length, 128 or 256 bits.
5. Add other attributes as needed, and then save the key.



Key Detail

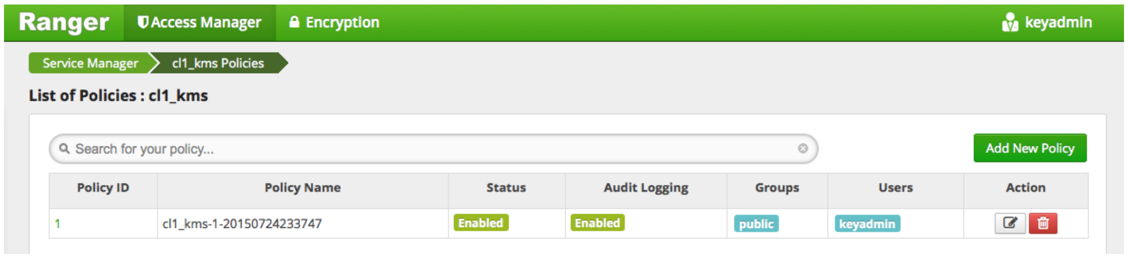
Key Name *	
Cipher	AES/CTR/NoPadding
Length	128
Description	
Attributes	Name / Value

6.3. Rolling Over an Existing Key

Rolling over (or "rotating") a key retains the same key name, but the key will have a different version. This operation re-encrypts existing file keys, but does not re-encrypt the actual file. Keys can be rolled over at any time.

After a key is rotated in Ranger KMS, new files will have the file key encrypted by the new master key for the encryption zone.

To rotate a key, click the edit button next to the key name in the list of keys, as shown in the following screenshot:

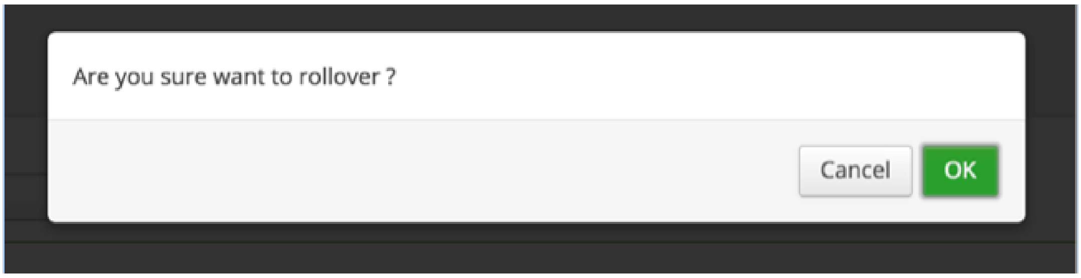


List of Policies : cl1_kms

Policy ID	Policy Name	Status	Audit Logging	Groups	Users	Action
1	cl1_kms-1-20150724233747	Enabled	Enabled	public	keyadmin	

Edit the key information, and then press Save.

When asked to confirm the rollover, click "OK":


Are you sure want to rollover ?

6.4. Deleting a Key

Warning

Deleting a key associated with an existing encryption zone will result in data loss.

To delete an existing key:

1. Choose the Encryption tab at the top of the Ranger Web UI screen.
2. Select Ranger KMS service from the drop-down list.
3. Click on the delete symbol next to the key.
4. You will see a confirmation popup window; confirm or cancel.

7. Ranger KMS Properties

This chapter describes configuration properties for the Ranger Key Management Service (KMS).

Table 7.1. Properties in Advanced dbks-site Menu (dbks-site.xml)

Property Name	Default Value	Description
ranger.ks.masterkey.credential.alias	ranger.ks.masterkey.password	Credential alias used for masterkey.
ranger.ks.jpa.jdbc.user	rangerkms	Database username used for operation.
ranger.ks.jpa.jdbc.url	jdbc:log4jdbc:mysql://localhost:3306/rangerkms	JDBC connection URL for database.
ranger.ks.jpa.jdbc.password	_ (default it's encrypted)	Database user's password.
ranger.ks.jpa.jdbc.driver	net.sf.log4jdbc.DriverSpy	Driver used for database.
ranger.ks.jpa.jdbc.dialect	org.eclipse.persistence.platform.database.MySQLPlatform	Dialect used for database.
ranger.ks.jpa.jdbc.credential.provider.path	/etc/ranger/kms/rangerkms.jceks	Credential provider path.
ranger.ks.jpa.jdbc.credential.alias	ranger.ks.jdbc.password	Credential alias used for password.
ranger.ks.jdbc.sqlconnectorjar	/usr/share/java/mysql-connector-java.jar	Driver jar used for database.
ranger.db.encrypt.key.password	_ (Default; it's encrypted)	Password used for encrypting the Master Key.
hadoop.kms.blacklist.DECRYPT_EEK	hdfs	Blacklist for decrypt EncryptedKey CryptoExtension operations. This can have multiple user IDs in a comma separated list. e.g `stormuser,yarn,hdfs`.

Table 7.2. Properties in Advanced kms-env

Property Name	Default Value	Description
Kms User	kms	Ranger KMS process will be started using this user.
Kms Group	kms	Ranger KMS process will be started using this group.
LD library path		LD library path (basically used when the db flavor is SQLA). Example: `/opt/sqlanywhere17/lib64`
kms_port	9292	Port used by Ranger KMS.
kms_log_dir	/var/log/ranger/kms	Directory where the Ranger KMS log will be generated.

Table 7.3. Properties in Advanced kms-properties (install.properties)

Property Name	Default Value	Description
db_user	rangerkms	Database username used for the operation.
db_root_user		Database root username. Default is blank. Specify the root user.
db_root_password		Database root user's password. Default is blank. Specify the root user password.

Property Name	Default Value	Description
db_password		Database user's password for the operation. Default is blank. Specify the Ranger KMS database password.
db_name	rangerkms	Database name for Ranger KMS.
db_host	<FQDN of instance where the Ranger KMS is installed>	Hostname where the database is installed. **Note**: Check the hostname for DB and change it accordingly.
SQL_CONNECTOR_JAR	/usr/share/java/mysql-connector.jar	Location of DB client library.
REPOSITORY_CONFIG_USERNAME	keyadmin	User used in default repo for Ranger KMS.
REPOSITORY_CONFIG_PASSWORD	keyadmin	Password for user used in default repo for Ranger KMS.
KMS_MASTER_KEY_PASSWD		Password used for encrypting the Master Key. Default value is blank. Set the master key to any string.
DB_FLAVOR	MYSQL	Database flavor used for Ranger KMS. Supported values: MYSQL, SQLA, ORACLE, POSTGRES, MSSQL

Table 7.4. Properties in Advanced kms-site (kms-site.xml)

Property Name	Default Value	Description
hadoop.security.keystore. JavaKeyStoreProvider.password	none	If using the JavaKeyStoreProvide, the password for the keystore file.
hadoop.kms.security. authorization.manager	`org.apache.ranger. authorization.kms. authorizer.RangerKmsAuthorizer`	Ranger KMS security authorizer.
hadoop.kms.key.provider.uri	dbks://http@localhost:9292/kms	URI of the backing KeyProvider for the KMS.
hadoop.kms.current.key. cache.timeout.ms	30000	Expiry time for the KMS current key cache, in milliseconds. This affects getCurrentKey operations.
hadoop.kms.cache.timeout.ms	600000	Expiry time for the KMS key version and key metadata cache, in milliseconds. This affects getKeyVersion and getMetadata.
hadoop.kms.cache.enable	true	Whether the KMS will act as a cache for the backing KeyProvider. When the cache is enabled, operations like getKeyVersion, getMetadata, and getCurrentKey will sometimes return cached data without consulting the backing KeyProvider. Cached values are flushed when keys are deleted or modified. **Note**: This setting is beneficial if Single KMS and single mode are used. If this is set to true when multiple KMSs are used, or when the key operations are from different modes (Ranger UI, CURL, or `hadoop` command), it might cause inconsistency.
hadoop.kms.authentication.type	simple	Authentication type for the Ranger KMS. Can be either "simple" or "kerberos".

Property Name	Default Value	Description
hadoop.kms.authentication.signer. secret.provider.zookeeper.path	/hadoop-kms/hadoop-auth-signature-secret	The ZooKeeper ZNode path where the Ranger KMS instances will store and retrieve the secret from.
hadoop.kms.authentication. signer.secret.provider. zookeeper.kerberos.principal	kms/#HOSTNAME#	The Kerberos service principal used to connect to ZooKeeper
hadoop.kms.authentication. signer.secret.provider. zookeeper.kerberos.keytab	/etc/hadoop/conf/kms.keytab	The absolute path for the Kerberos keytab with the credentials to connect to ZooKeeper.
hadoop.kms.authentication. signer.secret.provider. zookeeper.connection.string	#HOSTNAME#:#PORT#,...	The ZooKeeper connection string, a list of hostnames and port comma separated. For example: `<FQDN for first instance>:2181,<FQDN for second instance>:2181`
hadoop.kms.authentication. signer.secret.provider. zookeeper.auth.type	kerberos	ZooKeeper authentication type: 'none' or 'sasl' (Kerberos)
hadoop.kms.authentication. signer. secret.provider	random	Indicates how the secret to sign authentication cookies will be stored. Options are 'random' (default), 'string', and zookeeper'. If you have multiple Ranger KMS instances, specify 'zookeeper'.
hadoop.kms.authentication. kerberos.principal	HTTP/localhost	The Kerberos principal to use for the HTTP endpoint. The principal must start with 'HTTP/' as per the Kerberos HTTP SPNEGO specification.
hadoop.kms.authentication. kerberos.name.rules	DEFAULT	Rules used to resolve Kerberos principal names.
hadoop.kms.authentication. kerberos.keytab	${user.home}/kms.keytab	Path to the keytab with credentials for the configured Kerberos principal.
hadoop.kms.audit. aggregation.window.ms	10000	Specified in ms. Duplicate audit log events within this aggregation window are quashed to reduce log traffic. A single message for aggregated events is printed at the end of the window, along with a count of the number of aggregated events.

Table 7.5. Properties in Advanced ranger-kms-audit (ranger-kms-audit.xml)

Property Name	Default Value	Description
Audit provider summary enabled		Enable audit provider summary.
xasecure.audit.is.enabled	true	Enable audit.
xasecure.audit.destination. solr.zookeepers	none	Specify solr zookeeper string.
xasecure.audit.destination.solr.urls	{{ranger_audit_solr_urls}}	Specify solr URL. **Note**: In Ambari this value is populated from the Ranger Admin by default.
xasecure.audit.destination. solr.batch.filespool.dir	/var/log/ranger/kms/audit/solr/spool	Directory for solr audit spool.
Audit to SOLR		Enable audit to solr.
xasecure.audit.destination.hdfs.dir	hdfs://NAMENODE_HOST:8020/ ranger/audit	HDFS directory to write audit.

Property Name	Default Value	Description
		Note: Make sure the service user has required permissions.
xasecure.audit.destination. hdfs.batch.filespool.dir	/var/log/ranger/kms/audit/hdfs/ spool	Directory for HDFS audit spool.
Audit to HDFS		Enable hdfs audit.
xasecure.audit.destination.db.user	{{xa_audit_db_user}}	xa audit db user **Note**: In Ambari this value is populated from the Ranger Admin by default.
xasecure.audit.destination. db.password	encrypted (it's in encrypted format)	xa audit db user password **Note**: In Ambari this value is populated from the Ranger Admin by default.
xasecure.audit.destination.db.jdbc.url	{{audit_jdbc_url}}	Database JDBC URL for xa audit. **Note**: In Ambari the value for this is populated from the Ranger Admin by default.
xasecure.audit.destination. db.jdbc.driver	{{jdbc_driver}}	Database JDBC driver. **Note**: In Ambari this value is populated from the Ranger Admin by default.
xasecure.audit.destination. db.batch.filespool.dir	/var/log/ranger/kms/audit/db/spool	Directory for database audit spool.
Audit to DB		Enable audit to database.
xasecure.audit.credential.provider.file	jceks://file{{credential_file}}	Credential provider file.

Table 7.6. Properties in Advanced ranger-kms-policymgr-ssl

Property Name	Default Value	Description
xasecure.policymgr.clientssl. truststore.password	changeit	Password for the truststore.
xasecure.policymgr.clientssl. truststore	/usr/hdp/current/ranger-kms/conf/ ranger-plugin-truststore.jks	jks file for truststore
xasecure.policymgr.clientssl. keystore.password	myKeyFilePassword	Password for keystore.
xasecure.policymgr.clientssl. keystore.credential.file	jceks://file{{credential_file}}	Java keystore credential file.
xasecure.policymgr.clientssl. keystore	/usr/hdp/current/ranger-kms/conf/ ranger-plugin-keystore.jks	Java keystore file.
xasecure.policymgr.clientssl. truststore.credential.file	jceks://file{{credential_file}}	Java truststore file.

Table 7.7. Properties in Advanced ranger-kms-security

Property Name	Default Value	Description
ranger.plugin.kms.service.name	<default name for Ranger KMS Repo>	Name of the Ranger service containing policies for the KMS instance. **Note**: In Ambari the default value is `<clusterName>_kms`.
ranger.plugin.kms.policy.source.impl	org.apache.ranger.admin.client. RangerAdminRESTClient	Class to reterive policies from the source.
ranger.plugin.kms.policy.rest.url	{{policymgr_mgr_url}}	URL for Ranger Admin.

Property Name	Default Value	Description
ranger.plugin.kms.policy.rest. ssl.config.file	/etc/ranger/kms/conf/ranger-policymgr-ssl.xml	Path to the file containing SSL details for contacting the Ranger Admin.
ranger.plugin.kms.policy. pollIntervalMs	30000	Time interval to poll for changes in policies.
ranger.plugin.kms.policy.cache.dir	/etc/ranger/{{repo_name}}/ policycache	Directory where Ranger policies are cached after successful retrieval from the source.

8. Troubleshooting Ranger KMS

Table 8.1. Troubleshooting Suggestions

Issue	Action
Not able to install Ranger KMS	Check to see if ranger admin is running, verify DB.
Not able to start Ranger KMS	Check the Ranger KMS log. If there is a message about illegal key size, make sure unlimited strength JCE is available.
Hadoop key commands fail	Make sure Ranger KMS client properties are updated in hdfs config.
Not able to create keys from Ranger UI	Make sure that the `keyadmin` user (or any custom user) configured in the KMS repository is added to proxy properties in the custom `kms-site.xml` file.