How to copy encrypted data between two HDP clusters when Ranger KMS is utilized

(/users/98/emaxwell.html)

emaxwell (/users/98/emaxwell.html) created · Aug 18, 2016 at 04:51 AM · **edited (/revisions/51909.html)** · Aug 16, 2016 at 10:35 PM

10

Short Description:

This article describes the various ways to copy encrypted data between two clusters when Ranger KMS is used to manage the encryption keys.

Article

Since version 2.6, Apache Hadoop has had the ability to encrypt files that are written to special directories called encryption zones. In order for this at-rest encryption to work, encryption keys need to be managed by a Key Management Service (KMS). Apache Ranger 0.5 provided a scalable, open source KMS to provide key management for the Hadoop ecosystem. These features have made it easier to implement business and mission critical applications on Hadoop where security is a concern. These business/mission critical applications have also brought with them the need for fault tolerance and disaster recovery. Using Apache Falcon, it is easy to configure the copying of data from the Production Hadoop cluster to an off-site Disaster Recover (DR) cluster. But what is the best way to handle the encrypted data? Decrypting/encrypting the data to transfer it can hinder performance, but how do you decrypt data on the DR site without the proper keys from the KMS?

In this article, we will investigate 3 different scenarios for managing the encryption keys between two clusters when Ranger KMS is used as the key management infrastructure.

Scenario 1 - Completely Separate KMS Instances

The first scenario is the case where the Prod cluster has a Ranger KMS instance, and the KR cluster has a Ranger KMS instance. Each is completely separate with no copying of keys. This configuration has some advantage from a security perspective. Since there are two distinct KMS instances, the keys generated for encryption will be different even for the same directory within HDFS. This can provide a certain level of protection should the Production KMS instance be compromised, however, the tradeoff is in the performance of the data copy.

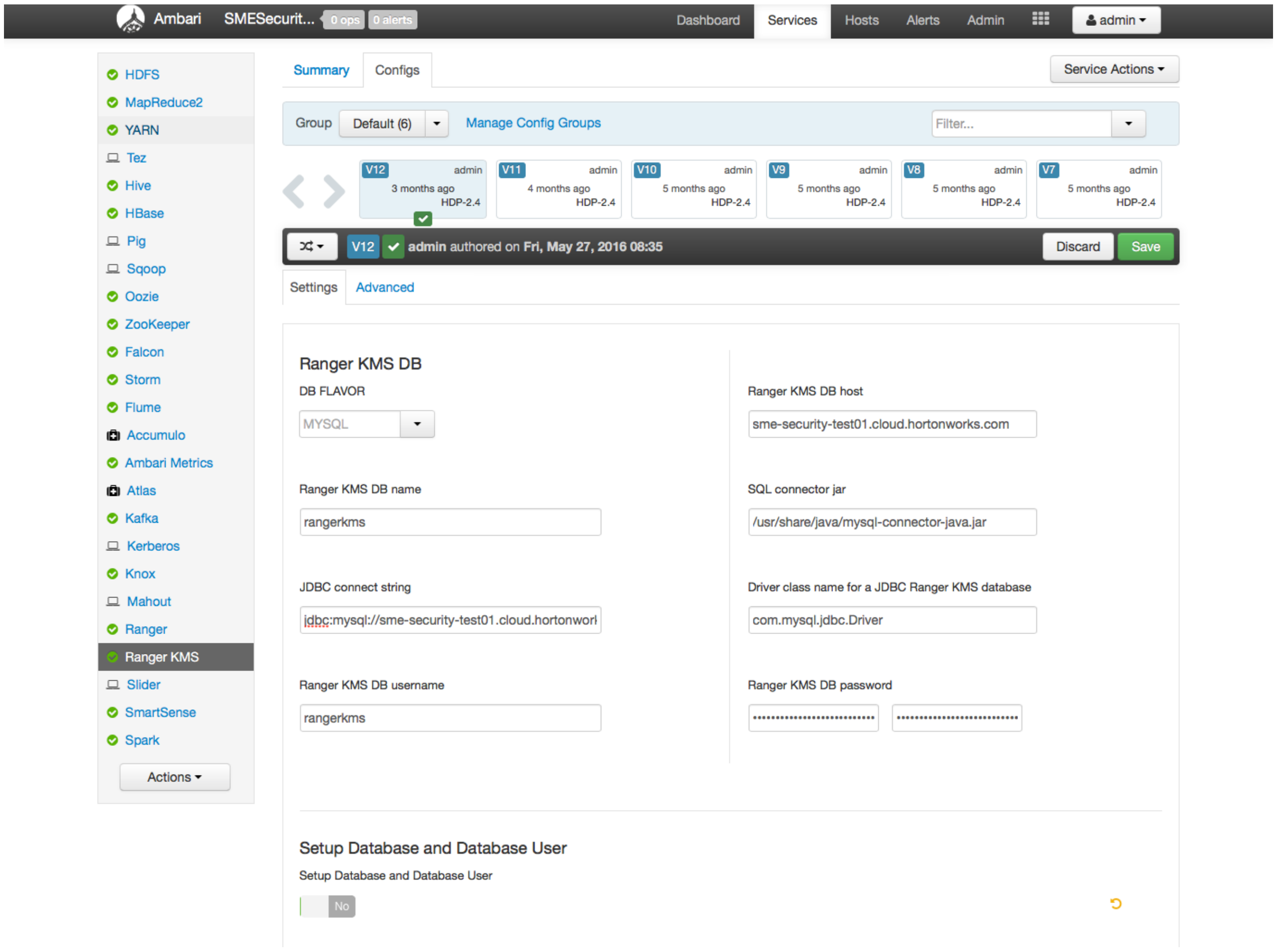
To copy the data in this type of environment, use the DistCp command similarly to how you would in a non-encrypted environment. DistCp will take care of the decrypt/encrypt functions automatically:

```
    ProdCluster:~$ hadoop distcp -update hdfs://ProdCluster:8020/data/encrypted/file1.txt \
      hdfs://DRCluster:8020/data/encrypted/
```

Scenario 2 - Two KMS instances, one database

In this configuration, the Prod and DR clusters each have a separate KMS Server, but the KMS Servers are both configured to use the same database to store the keys.

On the Prod cluster, configure the Ranger KMS per the Hadoop Security Guide (http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.4.2/bk_Security_Guide/content/install_ranger_kms.html). Once the KMS database is set up, copy the database configuration to the DR cluster's Ambari config tab. Make sure to turn off the "Setup Database and Database User" option at the bottom of the config page:



Ranger KMS DB

DB FLAVOR: MYSQL

Ranger KMS DB host: sme-security-test01.cloud.hortonworks.com

Ranger KMS DB name: rangerkms

SQL connector jar: /usr/share/java/mysql-connector-java.jar

JDBC connect string: jdbc:mysql://sme-security-test01.cloud.hortonworl

Driver class name for a JDBC Ranger KMS database: com.mysql.jdbc.Driver

Ranger KMS DB username: rangerkms

Ranger KMS DB password: •••••••••••••••••••••••••• ••••••••••••••••••••••••••

Setup Database and Database User

Setup Database and Database User: No

Once the KMS instances are both set up and working, creation of the encryption keys in this environment is simpler. Create the encryption key on the Prod cluster using either the Ranger KMS UI (login to Ranger as keyadmin), or via the CLI:

```
    ProdCluster:~$ hadoop key create ProdKey1
```

Specify which key to use to encrypt the data directory.

On the Prod cluster:

```
ProdCluster:~$ hdfs crypto -createZone -keyName ProdKey1 -path /data/encrypted
```

On the DR cluster, use the exact same command (even though it is for the DR cluster):

```
DRCluster:~$ hdfs crypto -createZone -keyName ProdKey1 -path /data/encrypted
```

Since both KMS instances use the same keys, the data can be copied using the /.reserved/raw virtual path to avoid decrypting/encrypting the data in transit. Note that it is important to use the -px flag on distcp to ensure that the EDEK (which is saved as an extended attribute) are transferred intact:

```
ProdCluster~$ hadoop distcp -px \
   hdfs://ProdCluster:8020/.reserved/raw/data/encrypted/file1.txt \
   hdfs://DRCluster:8020/.reserved/raw/data/encrypted/
```

Scenario 3 - Two KMS instances, two databases

In this configuration, the Prod and DR clusters each have a separate KMS Server, and each has it's own database store. In this scenario it is necessary to copy the keys from the Prod KMS database to the DR KMS database.

The Prod and DR KMS instances are setup separately per the Hadoop Security Guide (http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.4.2/bk_Security_Guide/content/install_ranger_kms.html). The keys for the encryption zones are created on the Prod cluster (the same as Scenario 2):

```
ProdCluster:~$ hadoop key create ProdKey1
```

Specify which key to use to encrypt the data directory on the Prod cluster:

```
ProdCluster:~$ hdfs crypto -createZone -keyName ProdKey1 -path /data/encrypted
```

Once the keys are created on the Prod cluster, a script is used to export the keys so they can be copied to the DR cluster. On the node where the KMS Server runs, execute the following:

```
ProdCluster:~# cd /usr/hdp/current/ranger-kms
ProdCluster:~# ./exportKeysToJCEKS.sh ProdCluster.keystore
Enter Password for the keystore FILE :
Enter Password for the KEY(s) stored in the keystore:
Keys from Ranger KMS Database has been successfully exported into ProdCluster.keystore
```

Now, the password protected keystore can be securely copied to the DR cluster node where the KMS Server runs:

```
ProdCluster:~# scp ProdCluster.keystore DRCluster:/usr/hdp/current/ranger-kms/
```

Next, import the keys into the Ranger KMS database on the DR cluster. On the Ranger KMS node in the DR cluster, execute the following:

```
DRCluster:~# cd /usr/hdp/current/ranger-kms
DRCluster:~# ./importJCEKSKeys.sh ProdCluster.keystore jceks
Enter Password for the keystore FILE :
Enter Password for the KEY(s) stored in the keystore:
Keys from ProdCluster.keystore has been successfully exported into RangerDB
```

The last step is to create the encryption zone on the DR cluster and specify which key to use for encryption:

```
DRCluster:~$ hdfs crypto -createZone -keyName ProdKey1 -path /data/encrypted
```

Now date can be copied using the /.reserved/raw/ virtual path to avoid the decryption/encryption steps between the clusters:

```
ProdCluster~$ hadoop distcp -px \
   hdfs://ProdCluster:8020/.reserved/raw/data/encrypted/file1.txt \
   hdfs://DRCluster:8020/.reserved/raw/data/encrypted/
```

Please note that the key copy procedure will need to be repeated when new keys are created or when keys are rotated within the KMS.

How-To/Tutorial (/topics/How-To%2FTutorial.html) **distcp (/topics/distcp.html)** **encryption (/topics/encryption.html)** **ranger-kms (/topics/ranger-kms.html)** **security (/topics/security.html)**

screen-shot-2016-08-16-at-41532-pm.png (/storage/attachments/6707-screen-shot-2016-08-16-at-41532-pm.png) (205.3 kB)

Add comment · Featured ✓

(https://dataworkssummit.com/san-jose-2018/?utm_campaign=dwssan-jose&utm_medium=banner&utm_source=website#packages-amp-passes)

ARTICLE

Contributors

(/users/98/emaxwell.html)

FOLLOW

+ FOLLOW (/FOLLOW/51909/KBENTRY.HTML)

(/user[illegible]alli.html)

(/user[illegible]ml)

NAVIGATION

How to copy encrypted data between two HDP clusters when Ranger KMS is utilized (/articles/51909/how-to-copy-encrypted-data-between-two-hdp-cluster.html)

RELATED ARTICLES

Kerberos cross realm trust for distcp (/articles/18686/kerberos-cross-realm-trust-for-distcp.html)

Setup cross realm trust between two MIT KDC (/articles/50935/setup-cross-realm-trust-between-two-mit-kdc.html)

Using Hadoop Credential API to store AWS secrets (/articles/59161/using-hadoop-credential-api-to-store-aws-secrets.html)

Transparent Data Encryption Explained: High-Level to In-Depth (and all those acronyms!) (/articles/176799/transparent-data-encryption-explained-high-level-t.html)

Choosing Kerberos approach for Hadoop cluster in an enterprise environment (/articles/17336/choosing-kerberos-approach-for-hadoop-cluster-in-a.html)

SSL configuration for Distcp accross the cluster in wire encrypted Multicluster Envrionment (/articles/198718/ssl-configuration-for-distcp-accross-the-cluster-i.html)

Using Transparent Data Encryption in HDFS (Non-Kerberized cluster) (/articles/42227/using-transparent-data-encryption-in-hdfs.html)

Enable HTTPS/SSL for HBASE Master UI (/articles/51165/enable-httpsssl-for-hbase-master-ui.html)

Demystifying Delegation Token (/articles/50069/demystifying-delegation-token.html)

Solr Rule-Based Authorization Plugin With External SolrCloud (/articles/105251/solr-rule-based-authorization-plugin-with-external.html)
