

Ranger KMS

Ranger KMS (Key Management Service)

- It's an 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.
- Multiple versions of Ranger KMS can be run using a load balancer

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

Ref: [Link](#)

HDFS Encryption Overview

- HDFS data at rest encryption implements end-to-end encryption of data read from and written to HDFS.
- End-to-end encryption means that data is encrypted and decrypted only by the client.
- HDFS does not have access to unencrypted data or keys.

HDFS encryption involves several elements:

- **Encryption key:**

A new level of permission-based access protection, in addition to standard HDFS permissions.

- **HDFS encryption zone:**

HDFS directory where all data is encrypted upon write, and decrypted upon read.

- Each encryption zone is associated with an encryption key that is specified when the zone is created.
- Each file within an encryption zone has a unique encryption key, called the "**data encryption key**" (DEK).
- HDFS does not have access to DEKs (HDFS DataNodes only see a stream of encrypted bytes). HDFS stores "encrypted data encryption keys" (EDEKs) as part of the file's metadata on the NameNode.
- Clients decrypt an EDEK and use the associated DEK to encrypt and decrypt data during write and read operations.

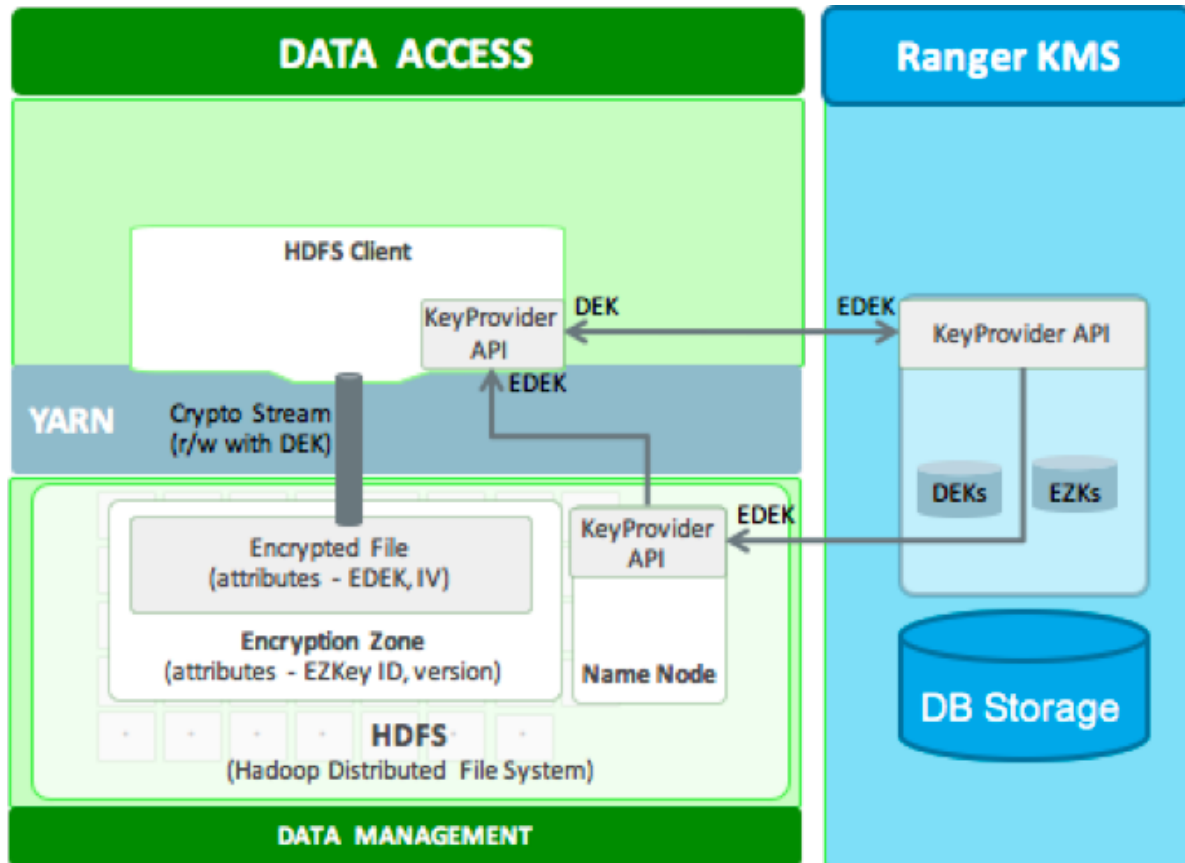
- **Ranger Key Management Service (Ranger KMS):**

An open source key management service based on Hadoop's *KeyProvider* API.

For HDFS encryption, the Ranger KMS has three basic responsibilities:

- Provide access to stored encryption zone keys
- Generate and manage encryption zone keys, and create encrypted data keys to be stored in Hadoop
- Audit all access events in Ranger KMS

HDFS Encryption Components

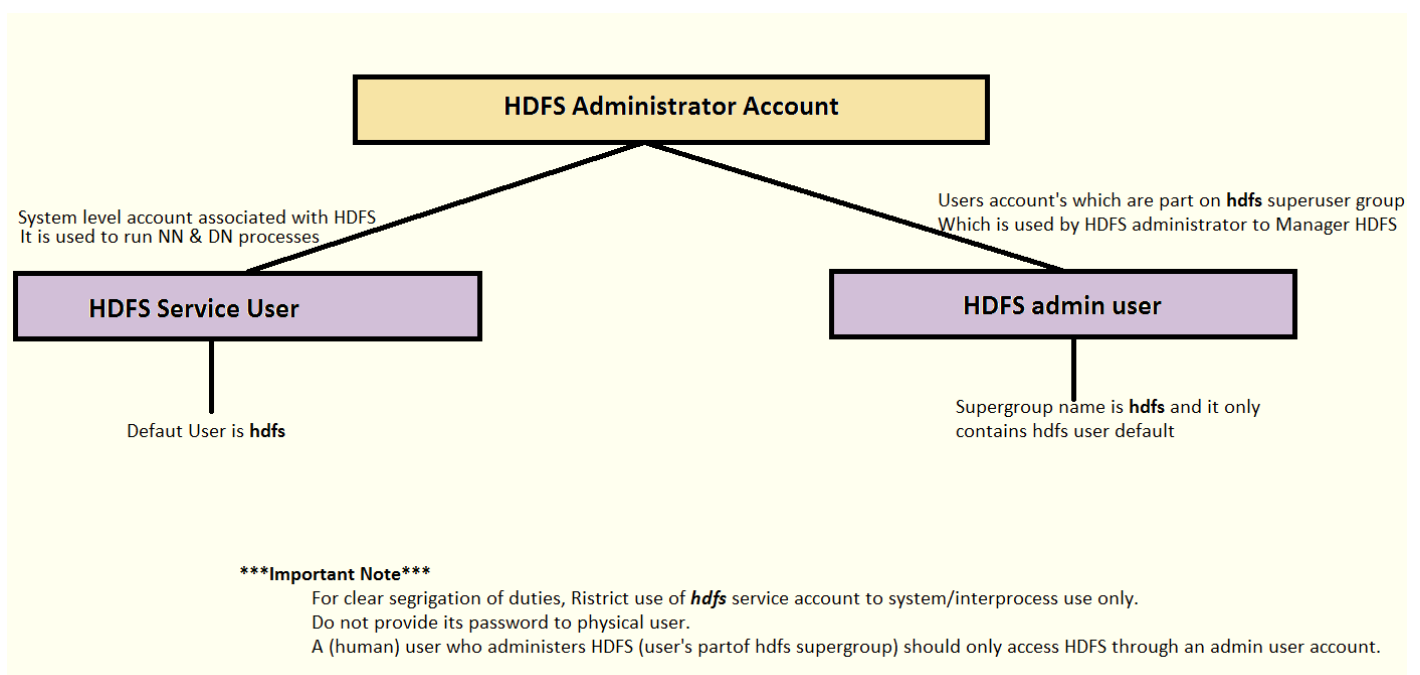


Role Separation

Access to the key encryption/decryption process is typically restricted to end users. This means that encrypted keys can be safely stored and handled by HDFS, because the HDFS admin user does not have access to them.

This role separation requires two types of HDFS administrator accounts:

- **HDFS service user:** the system-level account associated with HDFS (*hdfs* by default).
- **HDFS admin user:** an account in the *hdfs* supergroup, which is used by HDFS administrators to configure and manage HDFS.



[Ref Link](#) to create an HDFS admin account

Installation Prerequisites

- Ranger KMS requires HDFS and Ranger to be installed and running on the cluster.
- Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File should be installed on all hosts. For more information see [Install the JCE](#)

Install Ranger KMS using Ambari

- Goto Ambari dashboard
- **Actions** menu. Choose **Add Service**.
- Check the box next to **Ranger KMS**:
- Then, choose **Next**.
- **Assign Masters**: Choose host to install Ranger KMS
- **Customize Services**
 - Set required values (marked in red)
 - `KMS_MASTER_KEY_PASSWD`
 - `db_password`
 - `db_root_password`
 - Verify below properties
 - `DB_FLAVOR`
 - `db_host`
 - `db_root_user`
 - `db_root_password`
 - You can also customise the username for `REPOSITORY_CONFIG_USERNAME` (default set to **keyadmin**). This user will need to be set to proxy into Ranger KMS in a Kerberos mode (steps included below), so that Ranger will be able to connect to the Ranger KMS Server and look up keys for creating access policies.
 - Add the following properties to the Custom KMS-site section of the configuration.

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

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

- Add following properties to the Custom KMS-site section of the configuration. 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=*
```

- Verify below settings **advanced kms-site**

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

- Then, choose **Next**.

- Review the default values on the **Configure Identities** screen.
- In **Review**, make sure the configuration values are correct. Ranger KMS will be listed under **Services**.
- **Deploy**
- Restart stale services.

Configure HDFS Encryption to use Ranger KMS Access

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. In HDFS Config, 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`

The screenshot shows two configuration sections in a web interface. The first section, 'Advanced core-site', has a text input field for 'hadoop.security.key.provider.path' containing the value 'kms://http@node1.hadoop.com:9292/kms'. The second section, 'Advanced hdfs-site', has a text input field for 'dfs.encryption.key.provider.uri' also containing the same value. Both fields have icons for lock, refresh, and copy to the right.

3. In HDFS **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.

Use a Kerberos Principal for the Ranger KMS Repository

- In Ranger, all access policies are configured within a repository for each service.
- 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 principal for the Ranger KMS repository.
- Create system user `keyadmin` which should be sync in User Tabs in Ranger Admin

```
[root@node1 ~]# adduser keyadmin
```

- Create principal `keyadmin@EXAMPLE.COM` with password `keyadmin`

```
[root@node1 ~]# kadmin.local -q 'addprinc -pw keyadmin keyadmin'
```

- Under ranger-kms-properties, set the principal and password in the `REPOSITORY_CONFIG_USERNAME` and `REPOSITORY_CONFIG_PASSWORD` fields.

Advanced kms-properties

DB_FLAVOR	MYSQL	+	C
KMS_MASTER_KEY_PASSWORD			
REPOSITORY_CONFIG_PASSWORD			
REPOSITORY_CONFIG_USERNAME	keyadmin@TERADATA.LAB.COM	+	C
SQL_CONNECTOR_JAR	/usr/share/java/mysql-connector-java.jar	+	C
db_host	localhost	+	C
db_name	rangerkms	+	C
db_password			
db_root_password			
db_root_user	rangerdba	+	C
db_user	rangerkms	+	C

- Save and restart Stale configuration

Creating an HDFS Admin User(Recommended)

These steps use sample values for group (*operator*) and user account (*opt1*).

1. Create a new group called *operator*.

```
[root@node1 ~]# groupadd operator
```

2. Add a new user (*opt1*) to the group.

```
[root@node1 ~]# adduser -G operator opt1

[root@node1 ~]# id opt1
uid=1008(opt1) gid=1009(opt1) groups=1009(opt1),1008(operator)
```

3. Add principal *opt1@EXAMPLE.COM* and create a keytab.

```
[root@node1 ~]# kadmin.local -q "addprinc -pw hadoop opt1"

[root@node1 ~]# kadmin.local -q "ktadd -kt /home/opt1/opt1.keytab opt1"
```

4. Login as *opt1*, and do a *kinit* operation

```
[root@node1 ~]# chown opt1 /home/opt1/opt1.keytab

[root@node1 ~]# su - opt1

[opt1@node1 ~]$ kinit -kt opt1.keytab opt1
```

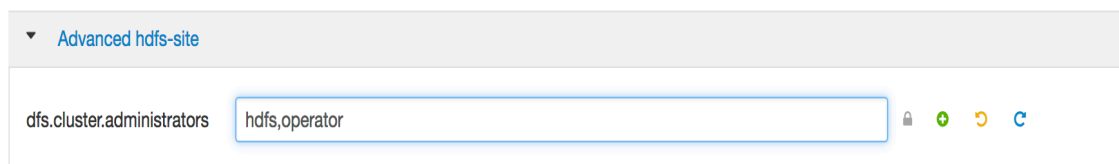
5. In Ambari, replace the current value of *dfs.permissions.superusergroup* with the group name “operator”.

Note:
You can add only one administrator group to the *dfs.permissions.superusergroup* parameter.

Advanced hdfs-site

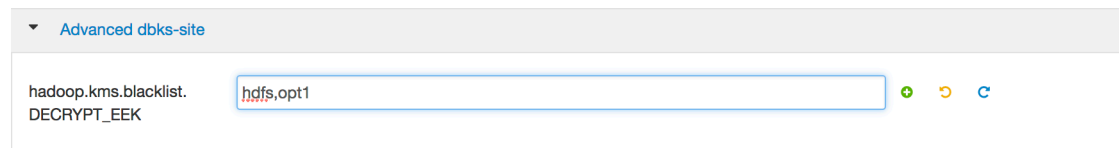
dfs.permissions.superusergroup	operator	🔒	+	↺	C
--------------------------------	----------	---	---	---	---

6. In Ambari, add `hdfs,operator` to `dfs.cluster.administrators`



A screenshot of the Ambari configuration interface. At the top, there is a dropdown menu labeled 'Advanced hdfs-site'. Below it, the configuration property 'dfs.cluster.administrators' is shown on the left. To its right is a text input field containing the value 'hdfs,operator'. To the right of the input field are three icons: a lock, a green plus sign, and a blue 'C' icon.

7. Add `opt1` to the KMS blacklist. Set the corresponding property in Ranger KMS configuration



A screenshot of the Ambari configuration interface. At the top, there is a dropdown menu labeled 'Advanced dbks-site'. Below it, the configuration property 'hadoop.kms.blacklist.DECRYPT_EEK' is shown on the left. To its right is a text input field containing the value 'hdfs,opt1'. To the right of the input field are three icons: a green plus sign, an orange arrow, and a blue 'C' icon.

8. Restart HDFS and Ranger KMS

Validation

- Make sure the `opt1` account has HDFS administrative access:

```
hdfs dfsadmin -report
```

- Make sure the `opt1` account cannot access encrypted files. For example, if `/data/test/file.txt` is in an encryption zone, the following command should return an error:

```
hdfs dfs -cat /data/test/file.txt
```

Using the Ranger Key Management Service

- Ranger KMS can be accessed at the Ranger admin URL, <http://<hostname>:6080>
- 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/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.

Configure HDFS encryption using Ranger KMS:

Goal: To create HDFS encryption zone for an end-user **gulshad**.

Pre-requisite:

1. As user keyadmin, create an encryption key via Ranger KMS UI using Ambari with a key called **ez-gulshad-key**
2. Connect as HDFS superuser **opt1** and create a directory via HDFS CLI and provide read/write permission to intended end-user **gulshad**

```
[opt1@node1 ~]$ hdfs dfs -mkdir /project
[opt1@node1 ~]$ hdfs dfs -chown gulshad:hdfs /project
[opt1@node1 ~]$ hdfs dfs -ls /
drwxr-xr-x  - gulshad hdfs          0 2018-04-26 18:36 /project
```

3. Logon to Ranger KMS UI and add user **hdfs**, **opt1** to the default KMS policy with permissions **GET_METADATA** and **Generate_EEK**.
4. Create an encryption zone with this directory via HDFS CLI

```
[opt1@node1 ~]$ hdfs crypto -createZone -keyName ez-gulshad-key -path /project
```

5. List the encryption zones as the **opt1**:

```
[opt1@node1 ~]$ hdfs crypto -listZones
```

6. Logon to Ranger KMS UI and create a new policy to allow **only** the Decrypt_EEK operation for **gulshad** user
7. Verify that user **gulshad** can write/read to/from the encryption zone /project

```
[gulshad@node1 ~]$ hdfs dfs -copyFromLocal /etc/hosts /project/
[gulshad@node1 ~]$ hdfs dfs -ls
[gulshad@node1 ~]$ hdfs dfs -cat /project/hosts
```

8. With the above steps, at this point, even admin users like **opt1** and **hdfs** cannot access this encryption zone and this is the expected & correct behaviour:

```
[gulshad@node1 ~]$ exit
[root@node1 ~]$ su - opt1

[opt1@node1 ~]$ hdfs dfs -copyFromLocal /etc/hosts /project/hosts2
copyFromLocal: User:opt1 not allowed to do 'DECRYPT_EEK' on 'ez-gulshad-key'

[opt1@node1 ~]$ hdfs dfs -cat /project/hosts
cat: User:opt1 not allowed to do 'DECRYPT_EEK' on 'ez-gulshad-key'
```