Enable DB2 native encryption in an HADR environment

Overview

The purpose of this tech note is to provide a simplified set of working steps, with examples, for the enablement of native encryption in an HADR environment. These steps are designed to minimize the downtime at the database service.

Prerequisites

In Version 10.5 Fix Pack 5 IBM introduced Native Encryption. It encrypts your DB2 database but requires no additional hardware, software, application(s), or schema changes, and provides transparent and secure key management.

Please note that between steps 9 to 17 HADR will be stopped and will not be replicating to a standby. To minimize the risk a 2nd standby could be added.

The following is a set of tested steps to enable native encryption in an HADR environment. It makes the following assumptions:

- The primary and the standby must be at version 10.5 fix pack 5 or later.
- There is a pre-existing un-encrypted database on both the primary and standby servers.

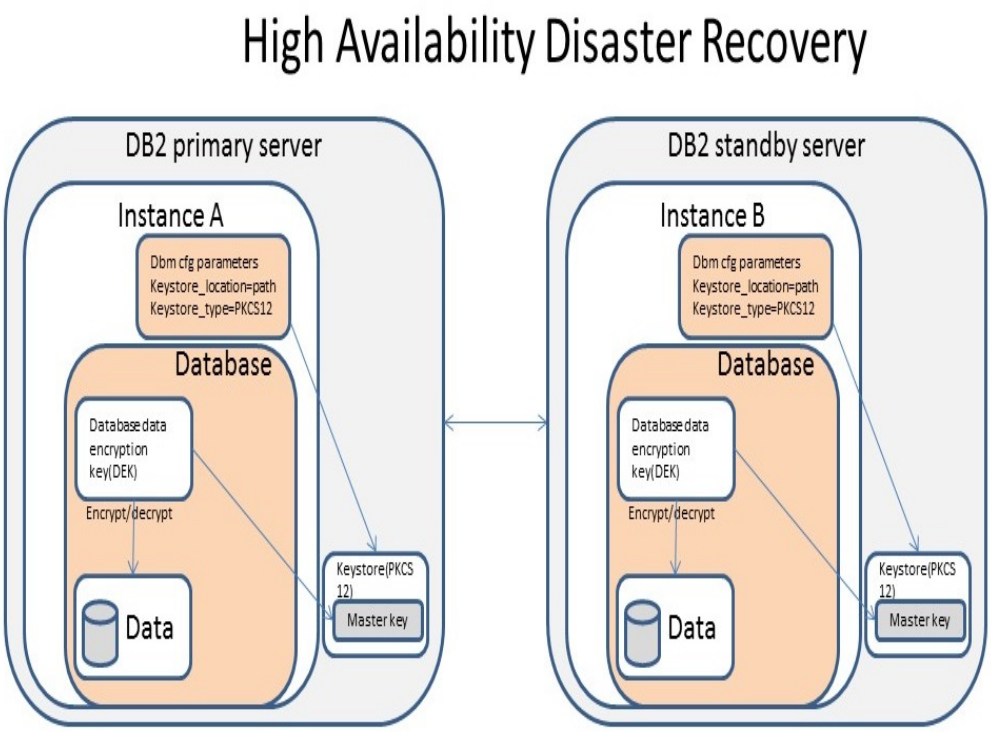

Figure 1: Native encryption in an HADR system

We will use Instance A and Instance B to differentiate between the HADR primary and HADR standby. The instructions will begin with Instance A as the primary and Instance B as the standby, as illustrated in the above figure. When a takeover operation is described we will explicitly specify what the new primary and standby are after the operation.

Step 1: Instance B: create a keystore and configure the instance for native encryption.

If instance B is already configured for native encryption skip this step.

```
gsk8capicmd_64 -keydb -create -db keystore_standby.p12 -pw
Str0ngPassw0rd -strong -type pkcs12 -stash

db2 update dbm cfg using keystore_type pkcs12 keystore_location
<keystore path>/keystore_standby.p12
```

Step 2: Instance B: create a master key.

Create a master key that will reside in the keystore to be used to encrypt the database.

```
gsk8capicmd_64 -secretkey -create -db keystore_standby.p12 -stashed -
label <label_name> -size 32
```

Step 3: Instance B: transfer master key to the other system.

Transfer the master key to be used in the restore command at step 9 from the keystore on instance B to instance A.

Export the master key to a temporary keystore that is to be transferred.

```
gsk8capicmd_64 -cert -export -db keystore_standby.p12 -stashed -label
<label-name> -target temp_keystore.p12 -target_pw temp0RaryPassw0rd -
target_type
pkcs12
```

Securely transfer the temporary keystore from instance B to instance A.

Step 4: Instance A: create a keystore and configure the instance for native encryption.

If instance A is already configured for native encryption skip this step.

```
gsk8capicmd_64 -keydb -create -db keystore_primary.p12 -pw
Str0ngPassw0rd -strong -type pkcs12 -stash

db2 update dbm cfg using keystore_type pkcs12 keystore_location
<keystore path>/keystore_primary.p12
```

Step 5: Instance A: import master key.

Import the master key from the temporary keystore that was transferred at step 3.

```
gsk8capicmd_64 -cert -import -db temp_keystore.p12 -pw
temp0RaryPassw0rd -label <label-name> -target keystore_primary.p12 -
stashed
```

Step 6: Instance A and B: remove temporary keystore.

On both instances, remove the temporary keystore file.

```
rm temp_keystore.p12
```

Step 7: Instance A: backup database and transfer backup image to the other system.

Backup the database on instance A with online option.

```
db2 backup db test online to <backup directory>
```

Securely transfer the backup image from instance A to instance B.

Step 8: Instance B: deactivate and drop database.

```
db2 deactivate db test
db2 drop db test
```

Step 9: Instance B: restore database.

Restore the database with the backup image transferred at step 7 using the 'encrypt' clause. Use the same master key label as the one that was used at step 2.

```
db2 restore db test from <restore directory> encrypt master key label
<label-name>
```

Step 10: Instance B: configure HADR local host/service name.

Configure the HADR local host/service name to HADR remote host/service name in database configuration, since the backup image is from Instance A.

```
db2 update db cfg for test using HADR_LOCAL_HOST <Instance B Host name>
HADR_LOCAL_SVC <Instance B Service name> HADR_TARGET_LIST <Instance A
Host name:Instance A Service Name>
```

Step 11: Instance B: start HADR, monitor.

Start HADR as the standby.

```
db2 start hadr on db test as standby
```

After the standby starts, it enters local catchup state in which locally available log files are read and replayed. After it has replayed all local logs, it enters remote catchup pending state.

Use db2pd to ensure that the standby is caught up with the primary side.

```
db2pd -db test -hadr
```

Step 12: Instance B: takeover.

Takeover hadr. After command completes instance B becomes primary and instance A becomes standby.

```
db2 takeover hadr on database test
```

Step 13: Instance B: backup database and transfer backup image to the other system.

Optionally, take a new backup to reduce the catchup time and transfer it to instance A.

```
db2 backup db test online to <backup directory>
```

Securely transfer the backup image from instance B to instance A.

Step 14: Instance A: deactivate and drop database.

```
db2 deactivate db test
db2 drop db test
```

Step 15: Instance A: restore database

Restore the most recent database backup, taken either at step 7 or 13, using the 'encrypt' clause. Use the same master key label as the one that was used at step 2.

```
db2 restore db test from <restore directory> taken at <timestamp>
encrypt master key label <label-name>
```

Step 16: Instance A: configure HADR local host/service name.

Optionally, if most recent backup image is restored at the previous step, configure the HADR local host/service name to HADR remote host/service name in database configuration, since the backup image is from Instance B.

```
db2 update db cfg for test using HADR_LOCAL_HOST <Instance A Host name>
HADR_LOCAL_SVC <Instance A Service name> HADR_TARGET_LIST <Instance B
Host name:Instance B service name>
```

Step 17: Instance A: start HADR, monitor, and takeover.

Start HADR as the standby.

```
db2 START HADR ON DB TEST AS STANDBY
```

Use db2pd to ensure that the standby is caught up with the primary side.

```
db2pd -d test -hadr
```

Optionally, once HADR has started successfully and has caught up, you may wish to takeover back to instance A.

```
db2 takeover hadr on database test
```

Step 18: Instance A and B: Verify database is encrypted.

On both instances, verify the database is encrypted.

```
db2 get db cfg for test | grep Encrypted
```

Step 19: Encrypt the database in auxiliary standbys.

Optionally, if you wish to encrypt the database in auxiliary standbys repeat steps 3 to 11. Step 7 is optional. You can use the backup previously taken at step 7 or 13 instead.

Resources

To learn how the DB2 native encryption feature works: http://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.admin.sec.doc/doc/c0061758.html

Enabling native encryption in a standalone DB2 instance and database is documented here and is beyond the scope of this tech note, although some of the steps are common: http://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.admin.sec.doc/doc/t0062041.html

For the DB2 native encryption feature to work properly, IBM Global Security Kit (GSKit) must be installed, and the environment must be configured correctly, as described here: https://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.admin.sec.doc/doc/t0062024.html

To learn more about keystore management best practices: https://www.ibm.com/developerworks/data/library/techarticle/dm-1504-master-encrypted-keys/

Authors

Stephen Levett - DB2 Level 2 Advanced Support

Mihai Nicolae - Software Developer - DB2 for LUW Security

Terry Lee - Software Developer - DB2 for LUW Security

Mihai Iacob - Software Developer - Watson Data Platform Security

Roger Zheng - DB2 for LUW HADR

Greg Stager - Software Developer - DB2 for LUW Security